004.032.26
O.Tyshchenko, I. Pliss
Kharkiv National University of Radioelectronics**,**
Control System Research Laboratory

AN EVOLVING RESERVOIR NEO-FUZZY NETWORK FOR TIME SERIES PREDICTION

Reservoir Computing is a paradigm of training Recurrent Neural Networks based on treating the recurrent part (the so-called "reservoir") differently from the readouts. This paradigm has become so popular recently due to its computational efficiency and the fact that it's enough to train only a supervised readout. Meanwhile Evolving Systems define a new approach which focuses on learning fuzzy systems that have both their parameters and their structure adapting on-line. In this paper an evolving reservoir neo-fuzzy network is built using time delay elements and nonlinear neo-fuzzy synapses which means that Reservoir Computing, Evolving Systems and Soft Computing are combined in a new computational system.

Keywords: reservoir computing, evolving systems, hybrid systems, neo-fuzzy neuron, online learning procedure, prediction.

Introduction

Reservoir Neural Networks (RNN) have been widely used in emulation and prediction tasks due to training a single linear output layer. Reservoirs are usually created randomly. So when the results don't have a required accuracy we have to start over training our network.

Evolving Systems (ES) are usually associated with streaming data and on-line (often real-time) modes of operation. They can be seen as adaptive intelligent systems with low-computational complexity. Evolving Systems assume on-line adaptation of system structure in addition to the parameter adaptation which is usually associated with the term "incremental".

Due to the implementation of a wide variety of adaptive, evolving and dynamic methodologies, they represent an important cornerstone within the field of "data-driven learning in non-stationary environments"[1,2]. Combining the favorable properties of (data-driven) fuzzy systems (especially universal approximation capabilities in connection with comprehensability and understandability aspects) with the concept of evolving modelling approaches may widen the applicability of fuzzy systems to on-line processing and modelling tasks. Under this scope, fuzzy systems inherit all the benefits and merits that evolving models have over batch modelling approaches [3-8].

Reservoir neural networks were basically designed to have the same computational power as traditional recurrent neural networks except the fact that there's no need to train internal weights [9-11]. The reservoir is a recurrent neural network which has n inputs, h internal units of the hidden layer and m output units. Input elements at a discrete time point k form a vector $\mathrm{x(k)} = \left(x_1(\mathrm{k})\ldots x_n(\mathrm{k})\right)^T$, internal units form a vector $\tilde{\mathrm{s}}(\mathrm{k}) = \left(\tilde{\mathrm{s}}_1(k)\ldots\tilde{\mathrm{s}}_h(k)\right)^T$ and finally output units form a vector $\tilde{x}(\mathrm{k}) = \left(\tilde{x}_1(\mathrm{k})\ldots\tilde{x}_m(\mathrm{k})\right)^T$. We take into consideration the simplest case when m equals to 1. So in this case the output signal is a scalar value.

Real-valued connection weights are collected in a $\left(h\times n\right)$ weight matrix W_{in} for the input weights, in an $\left(h\times h\right)$ matrix W_{res} for the internal connections, in an $m\times\left(h+n+m\right)$- matrix W_{out} for the connections to the output units, and in a $\left(n\times m\right)$-matrix W_{back} for the connections that feed

back from the output to the internal units. The weight matrix W_{in} is usually created randomly and can be either full or sparsed. The weight matrix W_{res} is usually randomly created according to Gaussian distribution. It can be easily explained by the fact that a network creates a reservoir full of different nonlinear current and preceding values (the so-called "echo"). Connections directly from the input to the output units and connections between output units are allowed. A reservoir should have a suitable perceptibility. The simplest way to make it is to tune a spectral radius (the biggest eigenvalue) of weight matrix W_{res}.

The activation of internal units is updated according to expression

$$\tilde{s}(k+1) = f\left(W_{in}x(k+1) + W_{res}\tilde{s}(k) + W_{back}\tilde{x}(k)\right), \quad (1)$$

where $\tilde{s}(k+1)$ is a vector of reservoir states at a time point k, where $f = (f_1 \ldots f_P)$ – the internal unit's output functions (typically sigmoid functions).

The output is computed according to equation

$$\tilde{x}(k+1) = f_{out}\left(W_{out}\begin{bmatrix}\tilde{s}(k+1)\\ x(k+1)\end{bmatrix}\right), \quad (2)$$

where $f_{out} = (f_{out1} \ldots f_{outT})$ – the output unit's output functions.

The most interesting thing about reservoirs is that all weight matrices to the reservoir are initialized randomly, while all connections to the output are trained. When using the system after training with the "reservoir-output" connections, the computed output by is fed back into the reservoir.

Reservoir construction is largely driven by a series of randomized model-building stages, which rely on a series of trials and errors. Typical model construction decision of an echo state network involves setting the reservoir size, the sparsity of the reservoir and input connections, the ranges for random input and reservoir weights, and the reservoir matrix scaling parameter. A simple, deterministically constructed cycle reservoir is comparable to the standard echo state network methodology [12]. The short-term memory capacity of linear cyclic reservoirs can be made arbitrarily close to the proved optimal value.

In this case the reservoir is created with the help of time delay elements and bell membership functions instead of traditional sigmoid functions. The reservoir output depends linearly on the inputs. So we propose to use neo-fuzzy neurons [13-15] in the output layer of the reservoir. We also propose a rather simple and quick learning algorithm.

A disadvantage of the NFN model is that it assumes that the nonlinearities of the inputs are separable. So the NFN model is not a universal approximator, in contrast to the neural nets or to the conventional fuzzy systems. But it possesses better approximation properties comparing to linear models, as it has more degrees of freedom allowing good piecewise-linear approximation for many processes and imposing very low requirements on the computational resources. The most important advantage of the NFN model is its extremely fast learning which can be performed in an online mode with very simple weight update rules, or in just a single operation with the linear least-squares formula.

An evolving network architecture

The proposed evolving reservoir neo-fuzzy architecture sets up both network parameters and network architecture.

Neo-fuzzy neurons were proposed by T. Yamakawa and co-authors [15]. These constructions are neuronal models with nonlinear synapses. The output of the nonlinear neuron is obtained by sum of the synapses outputs represented by nonlinear functions. They can approximate a nonlinear input-output relationship by one neuron, and there is no local minimum problem in learning [13, 14]. Fig.1 shows a structure of the conventional NFN. An input signal $x(k)$ is fed into the NFN layer. It should

be mentioned that this construction has back connections which come through the layer of delay elements from NFN outputs back to their inputs [16-18].

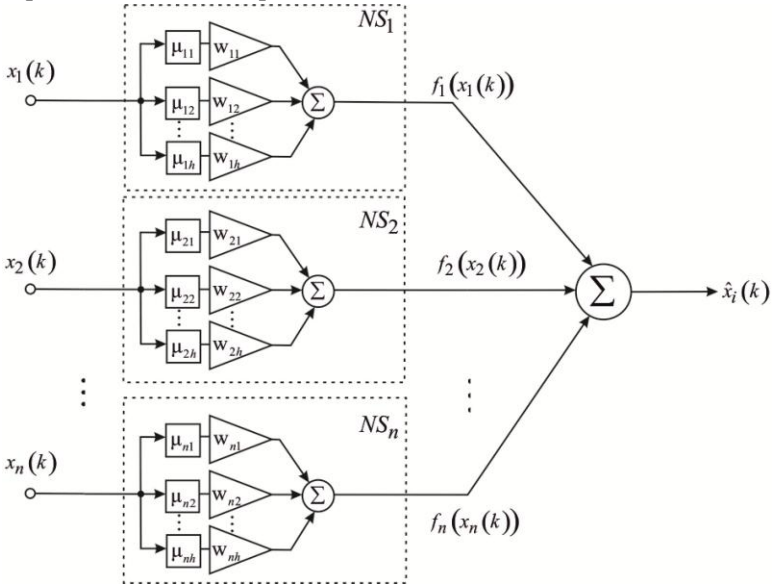

Fig.1 – The structure of a conventional neo-fuzzy neuron which consists of nonlinear neo-fuzzy synapses

We could get a MISO-object (multi-input single-output) schema by joining advantages of the above-mentioned reservoir and evolving computing approaches and using a neo-fuzzy synapse as a basic element of our new system. The reservoir is formed with the help of time delay elements and nonlinear synapses. If the required accuracy is not high enough while processing data one more nonlinear synapse is added to the network which turns a system into a NARMA(1,1)-object. We could also get in a simple way a NARMA(2,1)-object.

The simplest NAR first-order architecture [15] based on neo-fuzzy synapses is in fig.1. $x(k)$ is an input value of the network, $\tilde{x}(k)$ is a network output, $k = 1,2,\ldots,N,\ldots$ - current discrete time.

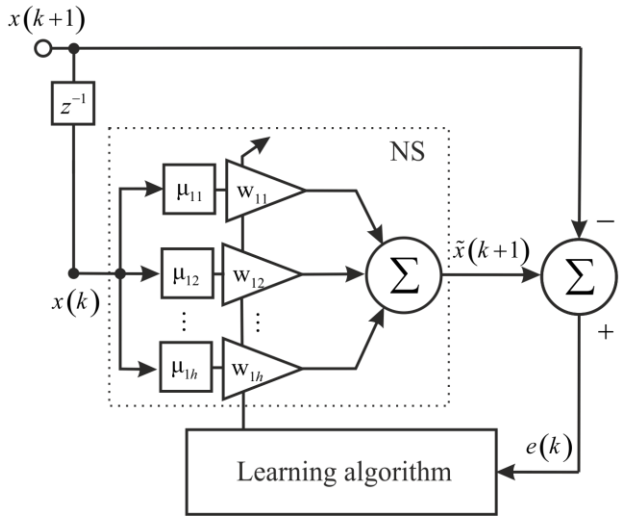

Fig.2 – The simplest “reservoir” architecture

$$\tilde{x}(k+1) = f\left(x(k)\right) = \sum_{j=1}^{h} \mu_{1j}\left(x(k)\right) w_{1j}(k). \qquad (3)$$

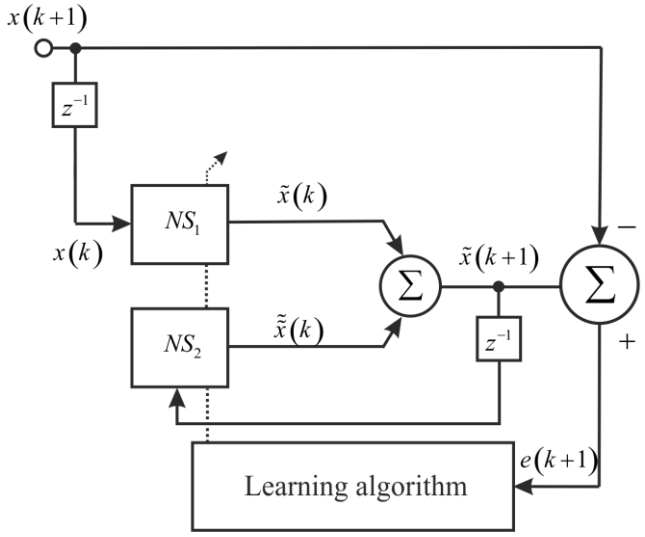

Fig. 3 – NARMA(1,1)-object

$$\tilde{x}(k+1) = f\left(\tilde{x}(k)\right) + f\left(\tilde{\tilde{x}}(k)\right), \tag{4}$$

$$f\left(\tilde{x}(k)\right) = \sum_{j=1}^{h} \mu_{1j}\left(x(k)\right) w_{1j}(k), \tag{5}$$

$$f\left(\tilde{\tilde{x}}(k)\right) = \sum_{j=1}^{h} \mu_{2j}\left(\tilde{x}(k)\right) w_{2j}(k). \tag{6}$$

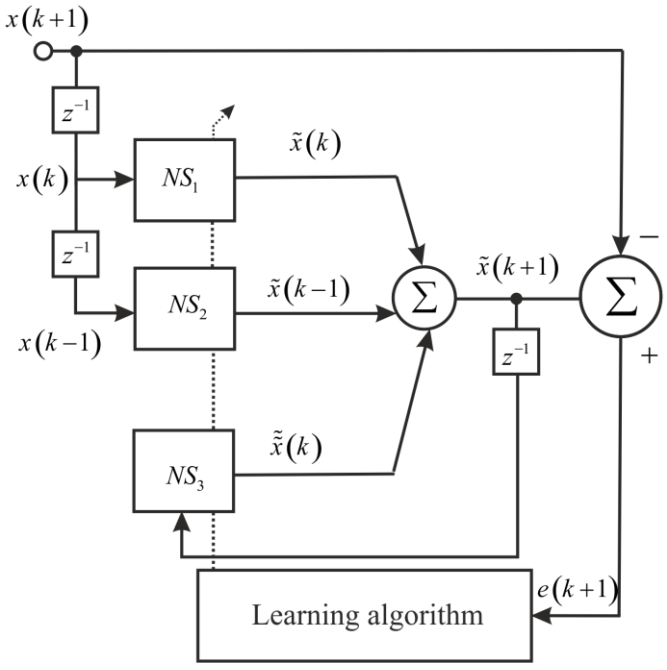

Fig. 4 – NARMA(2,1)-object

$$\tilde{x}(k+1) = f\left(\tilde{x}(k)\right) + f\left(\tilde{x}(k-1)\right) + f\left(\tilde{\tilde{x}}(k)\right), \tag{7}$$

$$f\left(\tilde{x}(k-1)\right) = \sum_{j=1}^{h} \mu_{2j}\left(x(k-1)\right) w_{2j}(k). \tag{8}$$

Taking into consideration a NARMA(n,1)-object the output can be written

$$\tilde{x}(k+1) = f\left(\tilde{x}(k)\right) + f\left(\tilde{x}(k-1)\right) + \ldots + f\left(\tilde{x}(k-n+1)\right) + f\left(\tilde{\tilde{x}}(k)\right). \tag{9}$$

Membership functions usually form a set of functions similar to the function array shown in fig. 5. In our case membership functions are evenly distributed in the range $[0,1]$.

It should be noticed that triangular membership functions provide piecewise-linear approximation which can lead to the results accuracy deterioration. To minimize this effect one may increase the amount of membership functions but this will increase the amount of synaptic weights and make a system architecture much more complicated as well as its learning algorithm. Cubic splines should be used as membership functions to get rid of the above-mentioned situations which can be written in the form:

$$\mu_{ij}\left(x_i\right) = \begin{cases} 0.25\left(2 + 3\dfrac{2x_i - c_{ij} - c_{i,j-1}}{c_{ij} - c_{i,j-1}} - \left(\dfrac{2x_i - c_{ij} - c_{i,j-1}}{c_{ij} - c_{i,j-1}}\right)^3\right), & x \in \left[c_{i,j-1}, c_{ij}\right], \\ 0.25\left(2 - 3\dfrac{2x_i - c_{i,j+1} - c_{ij}}{c_{i,j+1} - c_{ij}} + \left(\dfrac{2x_i - c_{i,j+1} - c_{ij}}{c_{i,j+1} - c_{ij}}\right)^3\right), & x \in \left(c_{ij}, c_{i,j+1}\right], \\ 0, \quad \textit{otherwise.} \end{cases} \tag{10}$$

Cubic splines satisfy Ruspini partition too and improve approximation characteristics of the fuzzy inference process. On the other hand, the usage of cubic splines provides smooth polynomial approximation and allows to model nonstationary signals with high accuracy results (fig. 5).

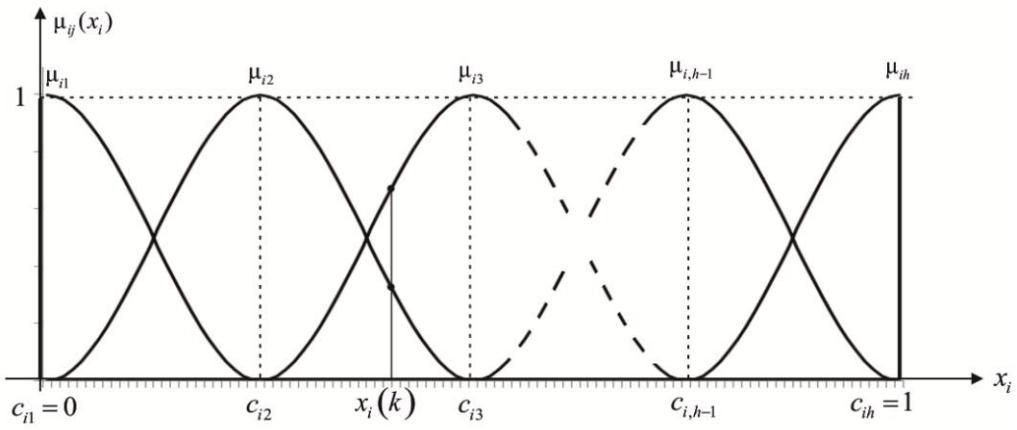

Fig. 5 – Cubic spline membership functions

Taking into consideration a $\left((n+1)h\right)\times 1$-vector of membership functions values

$$\mu(k)=\Big(\Big(\mu_{11}^{[q-1]}\big(x(k)\big);\ldots;\mu_{21}^{[q-1]}\big(x(k-1)\big);\ldots;\mu_{31}^{[q-1]}\big(x(k-2)\big);$$

$$\Big(\mu_{n1}^{[q-1]}\big(x(k-n+1)\big);\ldots;\mu_{n+1,h}^{[q]}\big(\tilde{x}(k)\big)\Big)^T,$$

where q stands for an amount of neo-fuzzy synapses in the evolving reservoir network architecture ($q-1$ says that the architecture hasn't been changed, q says that the architecture has been updated which means that new synaptic weights and membership functions have been added to the network architecture), and a synaptic weights vector of the same dimensionality

$$w(k)=\Big(\Big(w_{11}^{[q-1]}\big(x(k)\big);\ldots;w_{21}^{[q-1]}\big(x(k-1)\big);\ldots;w_{31}^{[q-1]}\big(x(k-2)\big);$$

$$\Big(w_{n1}^{[q-1]}\big(x(k-n+1)\big);\ldots;w_{n+1,h}^{[q]}\big(\tilde{x}(k)\big)\Big)^T,$$

the network output can be presented in a vector form

$$\tilde{x}=w^T\mu(k). \tag{11}$$

A Learning Procedure

To find optimal values of synaptic weights the conventional learning criterion

$$E=\frac{1}{2}\sum_{k=1}^{N}\left\|y(k)-\hat{y}(k)\right\|^2\alpha^{N-k} \tag{12}$$

and an adaptive procedure are used:

$$\begin{cases} w^q(k+1)=\begin{pmatrix} w_1^{q-1}(k)\\ \cdots\\ w_{n+1}^{q}(k)\end{pmatrix}+\eta^q(k)e(k-1)\begin{pmatrix} \mu_1^{q-1}(k+1)\\ \cdots\\ \mu_{n+1}^{q}(k+1)\end{pmatrix},\\ \eta^q(k)=\left(r^q(k)\right)^{-1},\\ r^q(k+1)=\alpha r^{q-1}(k)+\left\|\mu_1^{q-1}(k+1)\right\|^2+\left\|\mu_{n+1}^{q}(k+1)\right\|^2,\\ 0<\alpha\le 1. \end{cases} \tag{13}$$

Conclusion

The proposed architecture forms an evolving "reservoir" which is built using neo-fuzzy synapses. This network is designed to deal with nonstationary time series. The network has such

advantages as numerical simplicity and high processing speed while comparing it to traditional forecasting neural networks and neuro-fuzzy systems.

1. Lughofer E. On-line incremental feature weighting in evolving fuzzy classifiers // Fuzzy Sets and Systems. – 163(1). – 2011. – P. 1-23. 2. Lughofer E. Evolving Fuzzy Systems – Methodologies, Advanced Concepts and Applications // Studies in Fuzziness and Soft Computing. – Springer, 2011. – 410p. 3. Angelov P., Lughofer E., Zhou X. Evolving fuzzy classifiers using different model architectures // Fuzzy Sets and Systems. – 159(23). – 2008. – P. 3160-3182. 4. Baruah R.D., Angelov P. Evolving fuzzy systems for data streams: a survey // Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery. – 1(6). – 2011. – P. 461-476. 5. Angelov P., Zhou X. Evolving Fuzzy-Rule-Based Classifiers From Data Streams // IEEE Fuzzy Systems. – 16(6). – 2008. – P. 1462-1475. 6. Angelov P., Zhou X. Evolving Fuzzy Classifier for Novelty Detection and Landmark Recognition by Mobile Robots // Mobile Robots. – 2007. – P. 89-118. 7. Angelov P. A fuzzy controller with evolving structure // Information Science. – 161(1-2). – 2004. – P. 21-35. 8. Kasabov N. Evolving Connectionist Systems: The Knowledge Engineering Approach. – Springer London, 2007. – 451p. 9. Alexandre L.A., Embrechts M.J. Reservoir size, spectral radius and connectivity in static classification problems // ICANN 2009, Part I, LNCS 5768. – Springer-Verlag, Berlin Heidelberg, 2009. – P. 1015-1024. 10. Jaeger H. The echo-state approach to analysing and training recurrent neural networks // Technical report, German National Research Centre for Information Technology. – 2001. – 45p. 11. Jaeger H. Short-term memory in echo state networks // Technical report, German National esearch Centre for Information Technology. – 2001. – 36p. 12. Rodan A., Tino P. Minimum complexity echo state network // IEEE Transactions on Neural Networks. – 22(1). – 2011. – P. 131–144. 13. Miki T., Yamakawa T. Analog implementation of neo-fuzzy neuron and its on-board learning // Computational Intelligence and Applications. – Piraeus: WSES Press, 1999. – P. 144-149. 14. Uchino E., Yamakawa T. Soft computing based signal prediction, restoration and filtering // Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks and Genetic Algorithms. – Boston: Kluwer Academic Publisher, 1997. – P. 331-349. 15. Yamakawa T., Uchino E., Miki T., H. Kusanagi. A neo fuzzy neuron and its applications to system identification and prediction of the system behavior // Proc. 2-nd Int.Conf. on Fuzzy Logic and Neural Networks "IIZUKA-92". – Iizuka, Japan, 1992. –P. 477-483. 16. Bodyanskiy Ye., Tyshchenko O. Neo-fuzzy forecasting echo state network // Proc. 4th Int. Conf. ACSN-2009. – Lviv: NVF «Ukrainski Tehnologii», 2009. – P. 95-96. 17. Bodyanskiy Ye., Tyshchenko O. The reservoir predictive neuro-fuzzy network // Proc. Int. Conf. on Intellectual Systems for Decision Making and Problems of Computational Intelligence. – Vol.1. – Kherson: KhNTU, 2010. – P. 279-282. 18. Tyshchenko O., Pliss I. The forecasting neuro-neo-fuzzy network based on reservoir computing // Control Systems, Navigation and Connection Systems. – No.1(21), Vol.2. – Kyiv, 2012. – P. 123-126.