International Journal
Information Technology Applications (ITA)

Volume 9, Number 1, September 2020

MODELING OF NONLINEAR DYNAMIC SYSTEMS USING SOFT COMPUTING METHODS

Ján Cigánek, Filip Žemla

Abstract:

The presented paper deals with the modeling principles of nonlinear dynamic systems and with the design of optimal model structures created by artificial neural networks and fuzzy theory. Four models were created for modeling of selected system: a neural network model, an adaptive neuro-fuzzy model and two neural networks with optimal structure obtained using Optimal Brain Damage and Optimal Brain Surgeon algorithms. The quality of proposed solutions were compared and verified on a real system of steam turbine in thermal power plant and in nuclear power plant.

Keywords:

Modeling, artificial neural networks, Optimal brain damage, Optimal brain surgeon, pruning methods, dynamic system.

ACM Computing Classification System:

Computing methodologies,
modeling and simulation,
model development and analysis,
modeling methodologies.

Introduction

Presently, several methods for modeling of dynamic systems are known.

The latest techniques use artificial intelligence, which is able to approximate the behavior of complex nonlinear processes with a high degree of accuracy. One of the options for processing such information is represented by neural networks using fuzzy logic. The main advantage of this combination is their ability to learn.

Modern methods for the structure optimization of neural networks can more effectively exploit their properties by identification and modeling of dynamic systems.

Models have an important position by simulation, analysis and system design. The first step in modeling is to create a mathematical model.

Models typically arise from the understanding of physical patterns and interactions in the studied system. The complexity of the model depends on the given requirements of the model accuracy.

There is a lot of model information that can be difficult to write in the form of algebraic or differential equations, but there are several progressive methods that can introduce this information into the model.

1 Neural Networks

Artificial Neural Networks (ANNs) are widely used to development of empirical models for a broad class of scientific, engineering and business activities. In the area of decision-making and control, the ANNs are used to modeling linear and non-linear processes [6]. The basis of systems modeling using ANNs are their very good approximation properties.

Principly, the neural network consists of 3 base layers (input, hidden and output) with a different numbers of neurons, whereby only the neurons from different layers can be connected together (Fig.1).

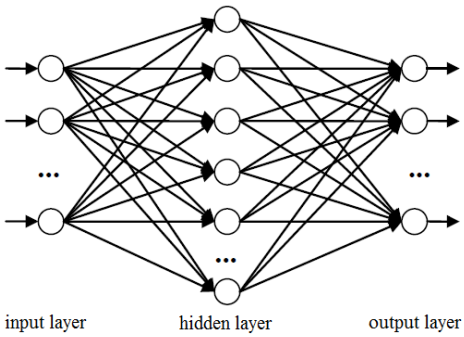

Fig.1. Neural network with 3 layers.

Neurons and neural network connections are rated by real numbers. Each neuron v_i is evaluated by the threshold x_i. Similarly, each joint between two neurons (v_j, v_i) is evaluated by the weighting factor w_{ji} (weight).

The target of the adaptive process in ANN is to find such thresholds and weights to minimalize the difference between required real output vector and simulated output vector from ANN.

According to the direction of transmitted signals, the ANNs are divided into two main groups: feed-forward and recurrent ANNs.

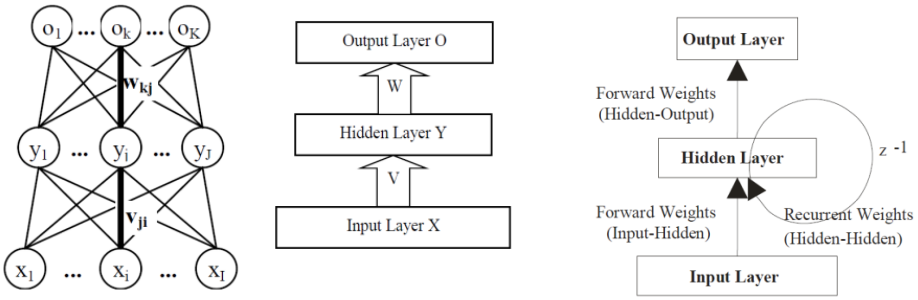

Fig.2. a) Feed-forward ANN, b) Recurrent ANN.

Feed-forward ANNs (Fig.2a) are those in which the signal spreads only from the input neurons through hidden neurons (any types of ANNs do not have hidden layer) to the output neurons. Input and output patterns do not depend on time.

The most widely used learning algorithm is the so-called Backpropagation algorithm – consists of calculating and redistributing of the error signal. While the target output on the output layer of the network is known, there can be easily calculated deviations of the actual output of the network from the required one. This difference is an error signal in the output layer. Then the error signal proceeds in the opposite direction, and the error signal propagates with a force adequate to the weight of the individual connections. Thus, at the output of the neuron in the hidden layer arrives the sum of the error signals from all neurons of output layer (transmitted by the weight of the individual connections).

In **recurrent ANNs** (Fig.2b) can also the signal be moved from the output layer to the hidden parts, or even to the input layer. Recurrent ANNs are an extension of feed-forward multilayer perceptron networks. The memory is realized by weight-delayed interconnections.

Both types of ANNs are the universal approximator, theoretically they can approximate arbitrary nonlinear projection with required accuracy.

2 Fuzzy Models

The mathematical model, created using fuzzy sets, is called a fuzzy model. In such a model, relations between variables are represented by "IF-THEN" rules. The aim of fuzzy modeling is to model and to control processes based on empirical knowledge. The basic advantage of fuzzy logic is the ability to mathematically represent the information expressed verbally.

The design of the shape, type and number of membership functions (MFs) is usually the most complex part of the design of each fuzzy system. Usually a triangular or trapezoidal MFs is used. The number of linguistic values for one fuzzy variable is different. Usually it is used 3 to 5 MFs for input variable and up to 7 MFs for output variable [6].

The basic principle of fuzzy modeling is the combination of input and output values and their formalization by means of rules [6]. The basic scheme of fuzzy system is shown in (Fig.3).

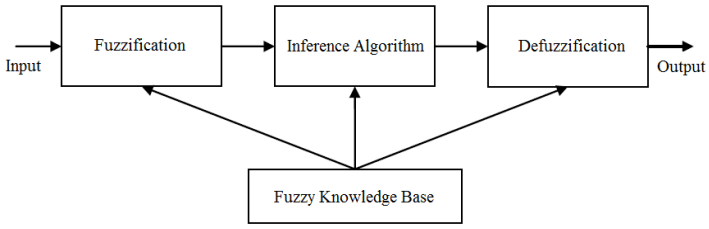

Fig.3. Basic scheme of fuzzy system.

Fuzzification - is the first step in the fuzzy inferencing process. This involves a domain transformation where crisp inputs are transformed into fuzzy inputs. Crisp inputs are exact inputs measured by sensors and passed into the fuzzy system for processing.

Fuzzy inference algorithm is the process of formulating the mapping from a given input to an output using fuzzy logic. The output values of all enabled rules for each linguistic variable are merged into one fuzzy set.

Defuzzification is the process of producing a quantifiable result of output value in Crisp logic, given fuzzy sets and corresponding membership degrees.

Fuzzy Knowledge Base - represents the facts of the rules and linguistic variables based on the fuzzy set theory so that the knowledge base systems will allow approximate reasoning.

3 Hybrid Models

It is possible to upgrade the properties of ANNs and fuzzy systems by their combination in hybrid systems [1]. (Tab.1) summarizes all the advantages and disadvantages of these two technologies. Hybrid systems are trying to combine these features so that the new system has maximum of advantages and minimum of disadvantages.

Table 1. Properties of ANNs and fuzzy systems.

	Advantages	**Disadvantages**
ANNs	- function approximation of various data - learning ANNs and their adaptability - parallelism of ANNs - hardware implementation	- learning time is relatively long - we can not precisely determine the topology - impossible implementation of a priori knowledge
Fuzzy systems	- possible implementation of a priori knowledge - manipulation of fuzzy sets	- no learning - knowledge base has to be designed by an expert

A typical example of hybrid models is ANFIS (Adaptive Neuro-Fuzzy Inference System). ANFIS realizes a fuzzy system of Takagi-Sugeno type using ANN. The methodology includes controlled learning of nodes in ANN by input data. In (Fig.4), adaptive nodes are marked with a square, while non-adaptive nodes are marked with a circle.

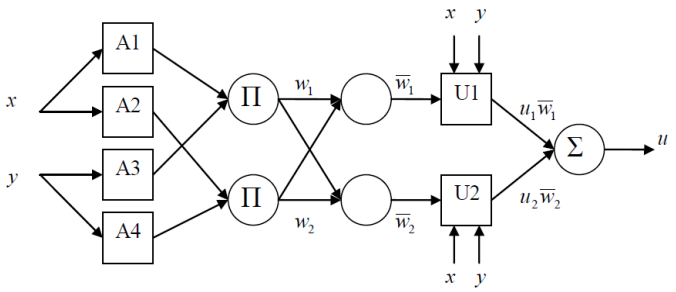

Fig.4. Example of ANFIS model.

The presented ANFIS model (Fig.4) contains two inputs x and y and one output u. The knowledge base consists of two IF - THEN fuzzy rules. The crisp value of the output is given by:

$$u = \frac{u_1 w_1 + u_2 w_2}{w_1 + w_2} = \overline{w_1} u_1 + \overline{w_2} u_2 \qquad (1)$$

The network consists of five layers. The nodes of each layer have the same nodal functions.

Layer 1: Each node of this layer is adaptive and its output is the membership value of the input x to the level A_i of the linguistic variable. Usually, we select the function μ_{Ai} with values in interval <0; 1>, i.e. in form (2):

$$\mu_{A_i} = \frac{1}{1 + \left(\frac{x - c_i}{a_i}\right)^{b_i}}, \qquad i = 1, \dots, 4 \tag{2}$$

where a_i, b_i, c_i are premise parameters changing the shape and placement of membership functions.

Layer 2: It represents the power of the rule. Each node of this layer is non-adaptive, labeled with a sign Π that multiplies input signals and transmits their product to next layer.

Layer 3: Each node of this layer is non-adaptive and calculates the power of the i-th rule to the power of all rules.

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}, \qquad i = 1, 2 \tag{3}$$

Layer 4: Each node of this layer is adaptive, with consequent parameters p_i, q_i, r_i:

$$\overline{w_i}\,\overline{u_i} = \overline{w_i}(p_i x + q_i y + r_i) \tag{4}$$

Layer 5: Non-adaptive node filling the suming function of input signals (1).

By the feed-forward, the signals propagate up to layer 4, the consequent parameters are identified by the least squares method. By back-propagation, the error signal is moving in reverse direction with updating the premise parameters.

4 Pruning Methods

One of the most important problems encountered in the practical application of ANNs is to find a suitable or ideally minimal ANN topology. Some of the main reasons are that an unsuitable topology increases the training time or even causes nonconvergence, thus it usually decreases the generalization capability of ANN [7].

The principle of pruning algorithms starts with a large ANN that is gradually decreasing [3]. The following methods represent various ways of removing connections and neurons. The pruning algorithm attempts to reduce ANN by eliminating unnecessary connections and neurons.

Sensitivity-based algorithms (requiring ANN convergence status) work according to the following procedure:

1. appropriate ANN is selected,
2. learning with back-propagation or similar algorithm is performed until ANN achieves a minimal error,
3. the importance of each element is calculated (connection or neuron) in terms of network performance,
4. the element of least importance shall be deleted,
5. re-learning of ANN (until a minimum of error function),
6. if the error is too high, return to point 3,
7. (optional) reinsert the last cut off element to obtain a lower error.

The simplest heuristic variant of pruning - after each learning step, the neural connection with the smallest weight is removed. The absolute value of the connection weight is therefore the only measure for the usefulness of the connection. Although this method is very simple, it rarely achieves worse results than more sophisticated algorithms.

Optimal Brain Damage (OBD) algorithm reduces the number of connections to prevent learning. The algorithm is designed to remove connections that will have the least impact on the network's error function *E(w)*, where *w* is the weight vector.

Under assumption that the original network is too large, after removing these connections new learning can improve the generalization capability of ANN. The OBD approximates the error function at the local minimum by Taylor's polynomial:

$$\delta E = \left(\frac{\delta E(w)}{\delta w}\right)^T \delta w + \frac{1}{2}\delta w^T H \delta w + F(\|\delta w\|^3) \tag{5}$$

where *H* is Hessian matrix of second partial derivatives.

To simplify the calculation, it is assumed that ANN has come the local minimum of error function *E* so that the first equation member of the right side can be ignored. Also third order and higher derivatives can be dismissed. By the Hessian approximation, only the diagonal elements are counted. After the neural connection removing, it is assumed that the effect on the error function will not depend on the settings of the other connections. The result of these simplifications is the equation:

$$\delta E = \frac{1}{2} \sum_{1 \le i,j \le n_k} h_{(i,j)}\, \delta w(ij)^2 \tag{6}$$

The importance of the line is inversely proportional to δE. The weight elimination procedure organizes the neural connections by importance. One or more of the least important connections is removed, ANN is re-learned and this cycle is repeated for the required number of times or till required accuracy of ANN model is reached.

Procedure steps of OBD algorithm:

1. The classical "reasonably large" feed-forward neural network is created, all neurons is neighbour layers are connected each with other.
2. Learning of ANN by selected algorithm until the minimum of error function is reached.
3. Evaluation of ANN on test data, storage of actual results (mean quadratic error for test data, network structure, values of weights and thresholds).
4. Elimination of weights that have the least impact on the error function (elimination of more weights are recommended, in our case study 5%). If there is nothing to be removed, then step 5 follows, otherwise go back to step 2.
5. The ANN structure with the smallest error was found (resulting in the optimal NN structure).

Optimal Brain Surgeon (OBS) algorithm is an improvement of the OBD algorithm. The benefits of OBS over OBD include that it does not require slow re-learning of ANN after connection withdrawal.

The approximation of the error function uses the same equation (5). The meaning of symbols is the same as for OBD. The first and third members of the Taylor's polynomial are ignored for the same reasons.

The first step of algorithm is to set one of the weights to zero (w_q) to minimize the increase in error function (5). The elimination of the weight w_q can be expressed by equation (7):

$$\delta w_q + w_q = 0 \quad or \quad e_q^T . \delta w + w_q = 0 \tag{7}$$

where e_q^T is the unit vector in the weight space, which also includes (scalar) weight w_q.

The task is to solve:

$$Min_q \left\{ Min_{\delta w} \left(\frac{1}{2} \delta w^T \, H \, \delta w \right) \right\} \tag{8}$$

under restriction (7). This leads to Lagrangian equation (9):

$$L = \frac{1}{2} \delta w^T \, H \, \delta w + \lambda \left(e_q^T . \delta w + w_q \right) \tag{9}$$

Thus, the resulting optimal weight change δw and resulting change in error L are:

$$\delta w = -\frac{w_q}{[H^{-1}]_{qq}} H^{-1} . e_q, \quad L_q = \frac{1}{2} \frac{w_q^2}{[H^{-1}]_{qq}} \tag{10}$$

Note: neither Hessian matrix H nor H^{-1} need to be diagonal [8]. Moreover, this method recalculates the magnitude of all the weights in ANN. L_q is called "saliency" of weight q – the increase in error after the weight elimination.

Procedure steps of OBS algorithm:

1. The classical "reasonably large" feed-forward neural network is created, all neurons in neighbour layers are connected each with other.
2. Learning of ANN by selected algorithm until the minimum of error function is reached.
3. Computation of inverted Hessian matrix H^{-1}.
4. Find such q that gives the minimal saliency L_q. If this candidate error increase is much smaller than E, then the qth weight w_q should be deleted and we proceed to step 5, othewise go to step 6. (Other stopping criteria can be used too.)
5. Use q from step 4 to update all weights δw (10). Go to step 3.
6. No more weights can be deleted without large increase of E. The ANN structure with the smallest error was found (resulting in the optimal NN structure).

5 Case Study

The dynamic system represents the steam turbine in the second block of the thermal power plant in Nováky, Slovak republic. Data for modeling were obtained by two measurements (Fig.5). The first measurement contains 4501 samples and the second measurement 3601 samples. The following variables were observed by the measurements: *y(t)* - steam pressure, *u(t)* - fuel supply, *ff(t)* - required active power.

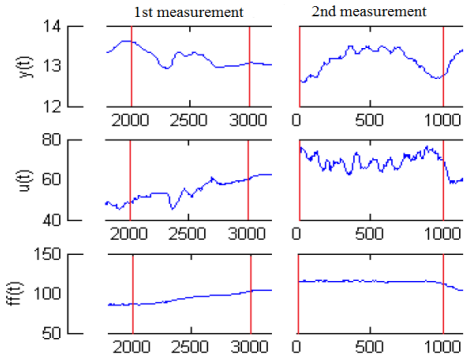

Fig.5. Training data with 2000 samples.

The target of the modeling will be to approximate the output variable - steam pressure *y(t)*. We consider that a nonlinear dynamic system is described by a differential equation in form (11):

$$y(t) = f[y(t-1), \dots, y(t-P), u(t), \dots, u(t-Q), \quad ff(t), \dots, ff(t-R)] \tag{11}$$

where *P*, *Q*, *R* are numbers of previous values. In order not to have large computational demands, the maximum number of past values was determined:

$$1 \le P \le 5; \;\; 0 \le Q \le 5; \;\; 0 \le R \le 5 \tag{12}$$

where 180 different combinations of inputs into ANN model is possible to create from the *P*, *Q*, *R* values. The simpliest input combination is *P=1*, *Q=0*, *R=0* with differential equation *y(t)=f[y(t-1),u(t),ff(t)]*, and the largest input combination is *P=5*, *Q=5*, *R=5*.

Usually, the more inputs into ANN are connected, the more accurate the model is. After few experiments, the combination with *P=5*, *Q=3*, *R=5* was selected.

As training data were selected two subsets of samples from both measurements (Fig.6) so that the complete training set contains 2000 samples to best characterize the dynamics of the system. The first set of data (from the first measurement) contains all samples between 2001 and 3000 and the second set of data (the second measurement) between 1 and 1000. These two subsets were merged into one training set.

For testing of each model, a whole set of data (both measurements separately) will be used to evaluate the model accuracy by the mean square error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(X_i - Y_i)^2 \tag{13}$$

where *N* is number of samples, X_i is required model value, Y_i is output model value.

Based on (12), the ANN (Fig.6) was created using Neural Network Toolbox in Matlab-Simulink. It is a feed-forward ANN with back-propagation error. Created ANN consists of 15 inputs, where the input layer only distributes the input signals, the hidden layer contains 10 neurons, and the output layer contains one output neuron. Each neuron is connected to all other neurons in neighbor layers and each neuron has a threshold. The transfer function on the hidden layer is sigmoid (tansig) and the transfer function on the output layer is linear (purelin). The Nguyen-Widrow algorithm [2] is used to initialize weights and thresholds.

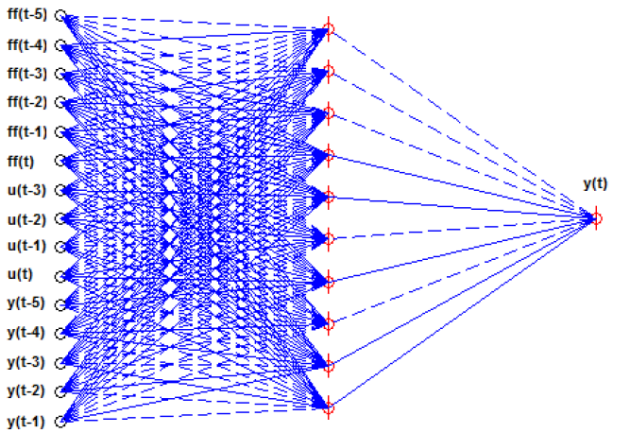

Fig.6. ANN structure.

Four learning algorithms [5] (all implemented in MATLAB-Simulink) were used to train:

1. Gradient descent with momentum backpropagation (traingdm).
2. Gradient descent with adaptive learning rate backpropagation (traingda).
3. BFGS quasi-Newton backpropagation (trainbfg).
4. Levenberg-Marquardt backpropagation (trainlm).

For each method, 10 training sessions were run, with the maximum number of epochs set to 1000. The average MSE for 10 sets of test data was used to evaluate (Tab.2) with the best rating for Levenberg-Marquardt backpropagation algorithm.

Table 2. Comparison of learning algorithms.

Algorithm	**Average MSE for test data 1**	**Average MSE for test data 2**
Traingdm	0.01395	0.00791
Traingda	0.00947	0.00582
Trainbf	0.00161	0.00066
Trainlm	0.00035	0.00011

The results obtained by ANN are acceptable, created models are able to approximate the current output value of the steam pressure for both measurements.

A. **ANFIS Model**

This model was created using Fuzzy logic Toolbox [4]. For network training is used ANFIS hybrid learning method. Parameters in the antecedent of a rule (determining the input variability attributes) are optimized by a gradient method, the subsequence parameters (constants or linear function coefficients) are counted using the least squares (backpropagation) method. The number of inputs for this model has been reduced to five, due to the simplification of the ANFIS model (by experimenting, it was found to be unsuitable to have many inputs for this model).

Selected dynamic system is identified by function $y(t) = f\,[y(t-1), u(t), u(t-1), ff(t), ff(t-1)]$, that will be approximed by ANFIS structure.

The fuzzy system has to be initialized; Takagi-Sugeno method was chosen. The genfis3 method [4] was used to create the fuzzy model using the clustering technique. The number of clusters is adjustable and determines the number of rules and the number of membership functions. The best results were obtained by 3 clusters.

The next step is to define the learning parameters. The number of iteration is 100 and the tolerance (error) is 10^{-6}. The last step is the learning process and the evaluation of the model accuracy on test data.

ANFIS model was tested for both test sets (first and second measurements). The red color is the ANFIS output and the blue color actual steam pressure value of the steam turbine (Fig.7).

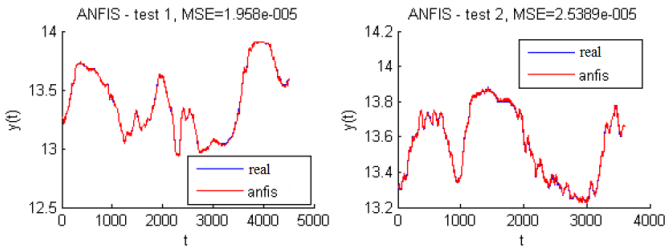

Fig.7. Comparison of ANFIS model with test data.

B. **Optimal Brain Damage**

The Levenberg-Marquardt backpropagation method was used to train the ANN. After successfully performing of OBD algorithm (by removing the connection), the simplified network structure was obtained (Tab.3) as it is shown in (Fig.8).

The blue dotted line represents the negative weight value, the plain blue line means the positive value, the red circle is the neuron (if it is checked, its threshold has a nonzero value) and the individual inputs are represented by black circles.

The optimal ANN structure was obtained after 56 iterations.

Table 3. Comparison of ANN with OBD algorithm.

	ANN before optimization	**ANN after optimization**
Number of connections	160	17
Number of inputs	15	11
Number of hidden neurons	10	4
Number of nonzero thresholds	11	4

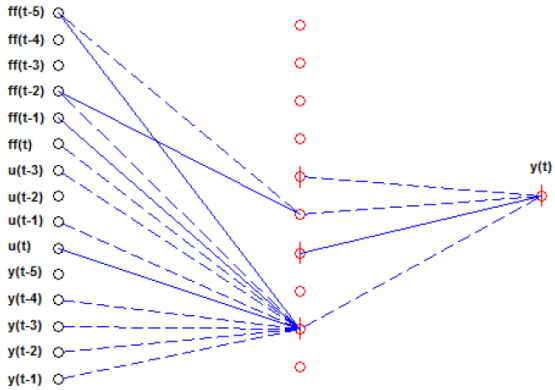

Fig.8. ANN structure after OBD algorithm.

C. **Optimal Brain Surgeon**

After successfully performing of OBS algorithm (by removing the connection), the simplified network structure was obtained (Tab.4) as it is shown in (Fig.9).

The optimal ANN structure was obtained after 60 iterations and it is even simpler than the above-mentioned ANN, optimized by OBD algorithm.

Table 4. Comparison of ANN with OBS algorithm.

	ANN before optimization	**ANN after optimization**
Number of connections	160	17
Number of inputs	15	7
Number of hidden neurons	10	2
Number of nonzero thresholds	11	1

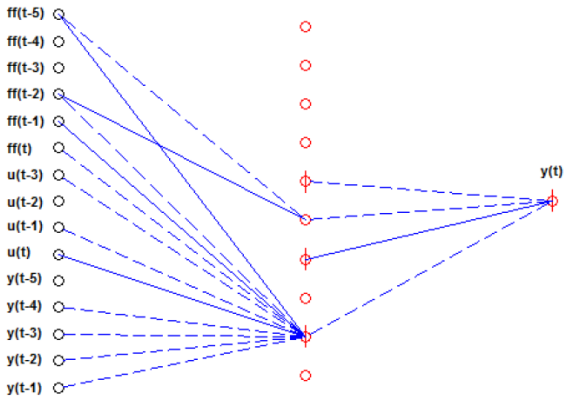

Fig.9. ANN structure after OBS algorithm.

D. **Results**

For a particular dynamic system, the steam turbine of the thermal power plant, four models were created. Each model is based on ANN.

The first model is a classical feed-forward ANN, the second one is an adaptive neuro-fuzzy model, and optimal structures of ANN were designed using two pruning methods (Optimal Brain Damage and Optimal Brain Surgeon). It has been confirmed that the proposed models are able to approximate the dynamic system, in our case study the output variable of steam presuure.

Obtained results are compared in (Tab.5). It is shown that the model accuracy is almost the same for all methods. However, the ANN model after OBS algorithm counts the lowest number of neurons and connections, this model obtained the best values of the model accuracy. These numbers are proof that ANN with optimal and incomplete structure can learn on a given set of data.

Table 5. Comparison of model accuracy.

Model	**MSE 1st test set**	**MSE 2nd test set**
ANN	$3.524*10^{-4}$	$1.0698*10^{-4}$
ANFIS	$1.958*10^{-5}$	$2.5389*10^{-5}$
ANN + OBD	$2.256*10^{-5}$	$2.227*10^{-5}$
ANN + OBS	$1.7896*10^{-5}$	$1.7013*10^{-5}$

Conclusion

Problems with a large number of neurons and connections in ANNs are huge; computational demands are high and a lot of time is needed to train the network. Conversely, if the number of connections in the network is too low, it does not have the power to provide the correct results for test examples. Therefore, the task is to find the correct relationship between network error and its complexity. Presented pruning methods for ANN optimization were used for approximation of any real dynamic system in energetics, e.g. for modeling a nuclear power plant in Jaslovské Bohunice. The result of these procedures is the exact model of a real dynamic system.

Acknowledgement

This paper was supported by the Slovak Grant Agency VEGA 1/0819/17, and by the Scientific Grants APVV-17-0190 and SEMOD-79-2/2019.

References

[1] Babjak, J. (2003). *Hybrid inteligency* (In Czech) http://www.root.cz/clanky/hybridna-inteligencia/

[2] Demuth, H., Beale, M. and Hagan, M. (2009). *Neural Network Toolbox ™ User's Guide*. The MathWorks, Inc..

[3] Bundzel, M. (2001). *Neural networks with adaptive topology*. University of technology, Kosice, Slovak republic.

[4] Jang, R., Gulley, N. (2009). *Fuzzy Logic Toolbox ™ User's Guide*. The MathWorks, Inc.

[5] Norgaard, M. (2000). *Neural Network Based System Identification Toolbox*. Technical University of Denmark

[6] Cigánek, J., Noge, F. (2014). *Fuzzy Logic Control of Mechatronic Systems*. In IEEE Int. conference on Process Control, Strbske Pleso, Slovak Republic.

[7] Ghosh, J., Tumer, K. (1994). *Structural adaptation and generalization in supervised feed-forward networks*. In Journal of Artificial Neural Networks, vol. 1, no. 4, pp. 431-458.

[8] Le Cun, Y., Denker, J. S. and Solla, S. A. (1990). *Optimal Brain Damage*. In Proceedings of Neural Information Processing Systems, pp. 598-605.

Authors

Ing. Ján Cigánek, PhD.
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
jan.ciganek@stuba.sk
He was born in 1981 in Malacky, Slovakia. He received the diploma and PhD. degree in Automatic Control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in Bratislava, in 2005 and 2010, respectively. He is now Assistant Professor at Institute of Automotive Mechatronics FEI STU in Bratislava. His research interests include optimization, robust control design, computational tools, SCADA systems, big data, and hybrid systems.

Ing. Filip Žemla
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
filip.zemla@icloud.com
He is currently a student of doctoral studies at Slovak University of Technology in Bratislava. The main focus of his studies is oriented to virtualization and optimization of modern manufacture processes. His main skills are SCADA systems, database systems and front-end programming.