УДК 004.032.26:519.174.2

PRACTICE METHOD OF NEURAL NETWORK PROCESSING

[1] *B.Ya. Bakay, Ph.D., Assoc. Prof.;*
[1] *S.V. Horzov, stud.*
[2]*P.V. Didenko, postgrad. stud.*

[1]***Ukrainian National Forestry University***
[2]***Zhytomyr National Agroecological University***

One of the effective and accurate modern modeling methods is neural networks [1]. In this paper, improved algorithm for constructing an interpreted factor structure is proposed. Also, topic of this paper carries a discussion about solving the factor rotation problem, associated with the use of interferometry in various fields of science, by using neural network.

A neural network allows to restore any unambiguous regularity between the source and target indicators. In this case, the restored regularities may have a nonlinear character [2-4]. For restoring a relation between the parameters, the error back propagation algorithm is used, which, from mathematical point of view, is a gradient optimization method.

The state of a neuron in an artificial neural network (fig. 1) is described by a set of variables: the weights of the input signals (w_1, w_2, ..., w_m), where m – number of input signals x_i; w_0 – free term computing the output signal. Individual neurons are combined into layers.

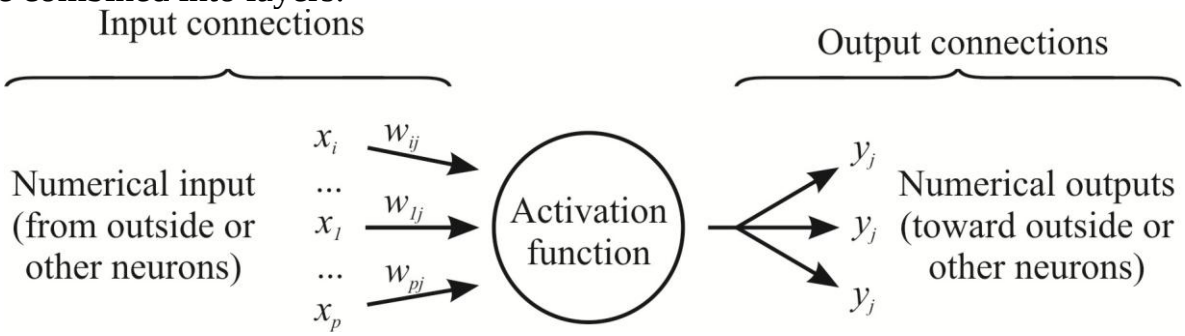

Fig. 1. Schematic structure of the artificial neuron

A neuron has several input signals x and one output signal y. The neuron parameters that determine its operation are: the vector of weights w, the threshold level θ, and the form of the activation function.

The signal at the output of the neuron is calculated by the formula $y_j = \sigma(v)$, where v – weighted sum of signals at the inputs of a neuron, σ – neuron transfer function (activation function), for example, sigmoidal (logistic) function $\sigma(v)=1/(1+\exp(-v))$ [1, 2]. The output signals of neurons from one layer are sent to the neurons of the next layer. Mentioned model is called multilayer perceptron.

To represent single layer perceptron (SLP) and multilayer perceptron (MLP), following expressions are used:

$$S = \sum_i w_i x_i , \qquad S_{jl} = \sum_i w_{ijl} x_{ijl} - \theta_{jl} , \tag{1}$$

where: w_i – weighting coefficients, x_i – input data, i – number of input signal, j – number of neuron in the layer, l – number of layer, θ_{jl} – threshold level of neuron j in layer l.

The most important and time-consuming process of model work is training. In neural network models, there is one or two hidden layers (excluding the input and output layers). The obtained result signal of model is compared with the actual value of dependent variable and from sample of the source data. After, the network error is calculated and its acceptability is evaluated.

Most often, the type of nonlinearity does not fundamentally affect the solution of the problem. However, a good choice can reduce training time. The choice of the activation function is determined by:

1. The specifics of the task;
2. Difficulty of implementation on a computer;
3. Learning algorithm: some algorithms can impose restrictions on the type of activation function.

The training procedure takes place in several iterations and after its completion, the network is ready to make a forecast. The above algorithm for the operation of a neural network is valid for solving many poorly structured problems that cannot be solved by direct analytical methods.

The prediction error is a key indicator of a mathematical model quality. Slight decrease of error can provide a significant advantage and good results for research.

The error back propagation algorithm for training a neural network corresponds to minimizing the error function $E(w_{ij})$. As an error function, the sum of the squared deviations of the network output signals from the required signals was used:

$$E = \sum_{i=1}^{m} (y_{ij} - y'_{ij})^2 , \tag{2}$$

where y_{ij} and y'_{ij} – output and required values of the i-th neuron of the j-th layer.

In this algorithm, learning iteration consists of three procedures:

1. Definition of signal and its distribution;
2. Calculation of signals at the output of each neuron;
3. Calculation of errors for each neuron;
4. Change of weights ties.

Performing repeated cyclic substitution of sets of signals at the input and output, as well as the back propagation of the error, the neural network is being trained.

To simulate, research, develop, and apply artificial Neural Networks we can use the following programs software Stuttgart Neural Network Simulator (SNNS),

NEST, Emergent, Neuron, Neural Lab, GENESIS, Brian etc. These are adapted from biological neural networks, and in some cases, a wider array of adaptive systems such as artificial intelligence and machine learning.

Modeling the work of neural networks with this software is relatively simple, but the researcher may have difficulty interpreting the obtained graphical results. Fig. 2 provides a model example network (fire Poisson spikes) for a group of *N* source neurons connected via synapses to a target neuron.

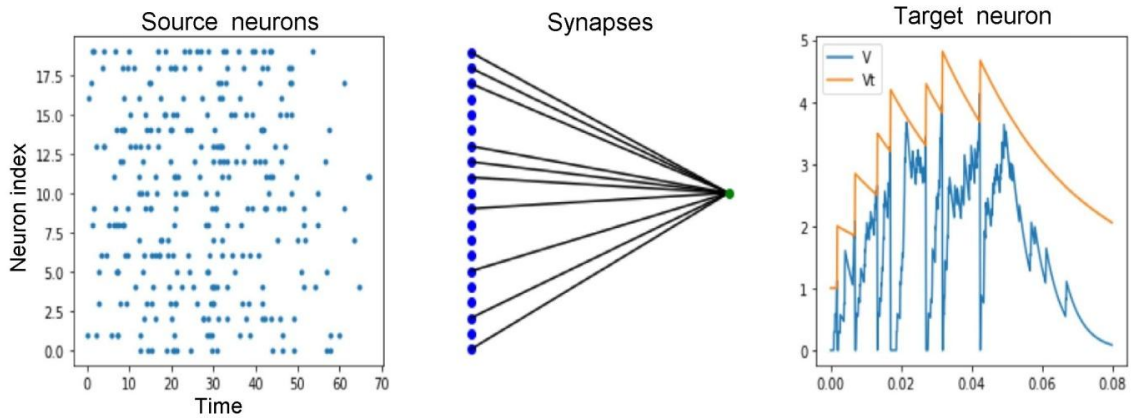

Fig. 2. An example of the neural network simulation results: source neurons; synapses; target neuron

Conclusion. An alternative approach to the construction of a factor model based on a neural network and an algorithm for the back propagation of errors is proposed. The advantages of this method are that it combines all the stages of classical factor analysis: the search for factor mapping and factor values. This method carries out oblique factor analysis, thereby having the maximum degree of generality for a linear model. In the course of a numerical experiment, it was found that good learning of the neural network is achieved when the number of neurons in the hidden layer is not less than the upper limit of the number of eigenvalues of the source variables.

References

1. Andrej Krenker, Janez Bešter and Andrej Kos (April 11th 2011). Introduction to the Artificial Neural Networks, Artificial Neural Networks – Methodological Advances and Biomedical Applications, Prof. Kenji Suzuki, IntechOpen. 362 p. ISBN 978-953-307-243-2. DOI: 10.5772/15751.
2. Ahmad Jobran Al-Mahasneh, Sreenatha G. Anavatti, Matthew A. Garratt (2018). Review of Applications of Generalized Regression Neural Networks in Identification and Control of Dynamic Systems. CoRR abs/1805.11236.
3. Ferreira, C. (2006). Designing Neural Networks Using Gene Expression Programming. In A. Abraham, B. de Baets, M. Köppen, and B. Nickolay, eds., Applied Soft Computing Technologies: The Challenge of Complexity, pages 517–536, Springer-Verlag.
4. Kruse, Rudolf; Borgelt, Christian; Klawonn, F.; Moewes, Christian; Steinbrecher, Matthias; Held, Pascal (2013). Computational intelligence : a methodological introduction. Springer. ISBN 9781447150121. OCLC 837524179.