NEURONAL NETWORK FOR MODELING OF SYSTEMS

Ing. Daniel ZELENKA, Doctoral Degree Programme (2)
Dept. of Software Engineering, FM, TUL
E-mail: daniel.zelenka@vslib.cz

Supervised by: Dr. Václav Záda

ABSTRACT

For location linear or non-linear model of dynamic system, we can use many methods. We can look for a model on the basis of physical properties of the real dynamics system, or only from knowledge of the system response to the input signal. Usually we try to describe the system by differential equations. Location of relevant equation is often very difficult if we know only input and output signals. If we do not need know differential equations of the dynamic system, we can use another access. For dynamic system modeling we can use, for example, neuronal network. We use backpropagation algorithm for learning of neuronal network.

1 INTRODUCTION

This paper is continuation of papers [1] and [2], and describes, how to use the neuronal network as model of the dynamics system. When we look for differential equations of linear and non-linear dynamic systems [3] and we know only the response of the system on its input signal, difficult problems can come. When we need to know only the response of the system on its input signal, we do not have to look for the description by means of the differential equation but we can use neuronal network as model of the dynamics system.

Our program for personal computer, written in Delphi 5 by Borland makes it possible to propose feedforward neuronal network on the basis of knowledge of the system response to the existing input signal. Measured values of the real system may be stored in files. For demonstrational exercise our program has the equipment with simulator of linear systems. After teaching the algorithm we can draw the graph of modeling and the real response of the dynamic system and compare the accuracy of the model.

2 SIMULATION OF LINEAR SYSTEM

If we use the integrated simulator of the linear system we use only the file with the input signal of the dynamic system. We obtain the output signal by simulation. During simulation we use the system of linear differential equations in this form:

$$\bar{x}' = \overline{A} \cdot \bar{x} + \overline{B} \cdot u \tag{1}$$

$$y = \overline{C} \cdot \bar{x} \tag{2}$$

where $\overline{A}$ is system matrix, $\overline{B}$ is input matrix and $\overline{C}$ is output matrix. Vector $\bar{x}$ is state vector of linear dynamic system, y is output signal and u is input signal. Initial condition are zeros. We set coefficients of matrix $\overline{A}$, $\overline{B}$, $\overline{C}$ directly in our program.

For solving by means of numerical computer we transform differential equation to difference equation [4]:

$$\bar{x}\left((k+1)\cdot T\right) = e^{\overline{A}\cdot T}\cdot\bar{x}(k\cdot T) + \overline{A}^{-1}\cdot\left(e^{\overline{A}\cdot T} - \overline{E}\right)\cdot\overline{B}\cdot u(k\cdot T) \tag{3}$$

$$y\left((k+1)\cdot T\right) = \overline{C}\cdot\bar{x}(k\cdot T) \tag{4}$$

where T is sampling period, which must be set by the user, k is a step of simulation calculation and $\overline{E}$ is unitary matrix.

For successful simulation the set linear dynamic system must be stable. We perform the verification of stability of the linear dynamic system by means of Routh - Schur's stability theorem. Coefficients of characteristic polynomial:

$$a(s) = \det\left(s\cdot\overline{E} - \overline{A}\right) = s^n + a_{(n-1)}\cdot s^{(n-1)} + a_{(n-2)}\cdot s^{(n-2)} + \ldots + a_1\cdot s + a_0 \tag{5}$$

are determined by means of Bocher's theorem [4].

3 NEURONAL NETWORK

3.1 STRUCTURE OF NEURONAL NETWORK

In our program we use feedforward neuronal network [5] with one output neuron, with one or two hidden layers. The user can set the number of layers according to how complicated the dynamic system is as well as the number of neurons in layers. The input layer is split into two parts. Samples of input signal of the system enter the first part. Their number depends on the user sets. Previous samples of the output enter the second part and form the inner dynamics of the dynamic system. The structure of the neuronal network is shown schematically in Fig. 1.

In hidden and output layers we use neurons with log-sigmoid transfer function:

$$y = \frac{1}{1+e^{-v}} \tag{6}$$

$$v = \sum_{i=1}^{n} w_i\cdot x_i \tag{7}$$

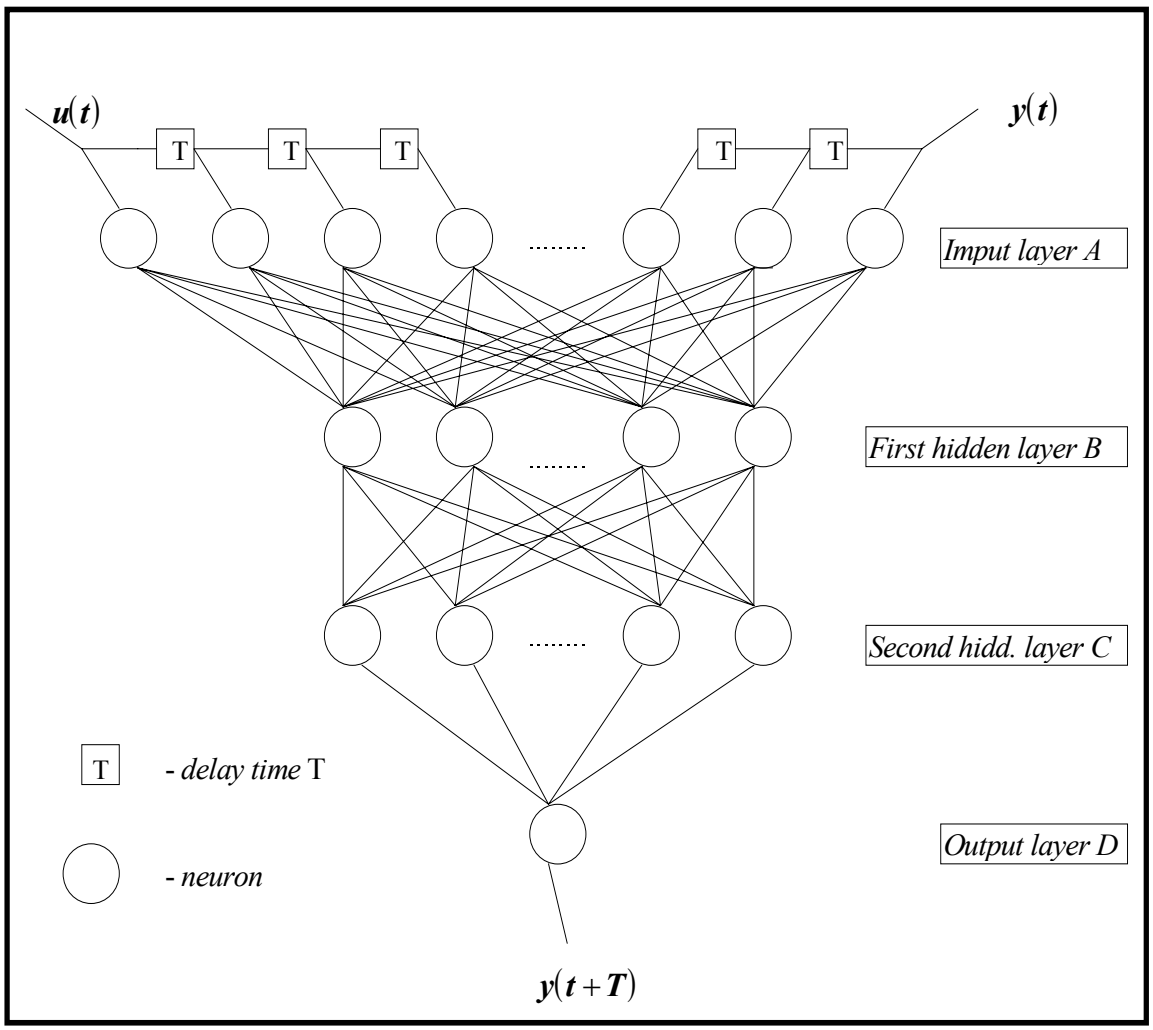

Fig. 1: *Structure of neuronal network*

where y is output value of neuron, x_i is input value with index number i and w_i is synaptic weight for input i. We use neurons with zero treshold and learning algorithm modify only synaptic weights of neurons w. The output signal of used neurons has the value from interval $[0;1]$. Before start of algorithm of learning our program must always perform the standardization of input and output neuronal network signals gained from the file or by simulation. Our program uses this formula:

$$y_{n(i)} = \frac{y_{(i)} - \min}{\max - \min} \tag{8}$$

where $y_{n(i)}$ is standardized sample, $y_{(i)}$ is original sample of number i and $\min, \max$ are maximum and minimum values from standardized signal.

3.2 BACKPROPAGATION - LEARNING METHOD

We have used the modification of their synaptic weights w in our program for learning of neuronal network - basic backpropagation algorithm [6]. We quantify the output neuron

error by means of this formula:

$$\boldsymbol{d}_{\boldsymbol{D}} = \boldsymbol{y}_{\boldsymbol{D}} \cdot \left(1 - \boldsymbol{y}_{\boldsymbol{D}}\right) \cdot \left(\boldsymbol{y}_{\boldsymbol{n}} - \boldsymbol{y}_{\boldsymbol{D}}\right) \tag{9}$$

where $\boldsymbol{y}_{\boldsymbol{D}}$ is output signal of neuron and $\boldsymbol{y}_{\boldsymbol{n}}$ is desired value of the output. Errors $\boldsymbol{d}_{\boldsymbol{B}}$ of single neurons number $\boldsymbol{i}$ in hidden layer $\boldsymbol{B}$ we quantified by means of this formula:

$$\boldsymbol{d}_{\boldsymbol{B}(i)} = \boldsymbol{b}_{\boldsymbol{i}} \cdot \left(1 - \boldsymbol{b}_{\boldsymbol{i}}\right) \cdot \left(\sum_{j} \boldsymbol{w}_{\boldsymbol{BC}(i,j)} \cdot \boldsymbol{d}_{\boldsymbol{C}(j)} \right) \tag{10}$$

where $\boldsymbol{b}_{\boldsymbol{i}}$ is output signal of neuron number $\boldsymbol{i}$ in layer $\boldsymbol{B}$, $\boldsymbol{d}_{\boldsymbol{C}(j)}$ is error of neuron number $\boldsymbol{j}$ in layer $\boldsymbol{C}$ and $\boldsymbol{w}_{\boldsymbol{BC}(i,j)}$ is synaptic weight between neurons $\boldsymbol{i}$ in layer $\boldsymbol{B}$ and $\boldsymbol{j}$ in layer $\boldsymbol{C}$.

New synaptic weight $\boldsymbol{w}_{\boldsymbol{ABnew}(i,j)}$ between layers $\boldsymbol{A}$ and $\boldsymbol{B}$ we quantify:

$$\Delta \boldsymbol{w}_{\boldsymbol{AB}(i,j)} = \alpha_{\boldsymbol{AB}} \cdot \boldsymbol{d}_{\boldsymbol{B}(ji)} \cdot \boldsymbol{y}_{\boldsymbol{A}(i)} \tag{11}$$

$$\boldsymbol{w}_{\boldsymbol{ABnew}(i,j)} = \boldsymbol{w}_{\boldsymbol{ABold}(i,j)} + \Delta \boldsymbol{w}_{\boldsymbol{AB}(i,j)} \tag{12}$$

where $\boldsymbol{y}_{\boldsymbol{A}(i)}$ is output signal of input neuron number $\boldsymbol{i}$, $\alpha_{\boldsymbol{AB}}$ is learning rate for synaptic weights between layer $\boldsymbol{A}$ and $\boldsymbol{B}$.

We calculate the quadratic error Δ of neuronal network response to the input signal from desired reaction which can be see during learning of neuronal network together with the actual step of learning as follows:

$$\Delta = \frac{\sum_{k=1}^{l} \left(\boldsymbol{y}_{\boldsymbol{D}(k)} - \boldsymbol{y}_{\boldsymbol{n}(k)} \right)^2}{\boldsymbol{l}} \tag{13}$$

where $\boldsymbol{l}$ is number of measured samples of input and output signal of dynamic system.

3.3 LEARNING OF NEURONAL NETWORK

When the program starts, first we must select the names of input and output files. The user must enter the file with measured input signal of dynamic system, the file with measured output signal of dynamic system (When we use simulator of linear dynamic system, we can not set name of this file), the file for storage of synaptic weights of neuronal network and file for storage of results of simulation of the dynamic system. First, second and fourth files are textfiles and they are used by, for example, mathematical software MatLab by MathWorks.

When the user sets generating of the new neuronal network, neuronal network initialization is used (Detailed description of neuronal network initialization we can find, for example, in [5]), otherwise program uses the neuronal network from the select file. Then the user selects number of neuronal network hidden layers and sets the number of their neurons. The user must set number of input neurons for signal $\boldsymbol{u}(\boldsymbol{t})$, for signal $\boldsymbol{y}(\boldsymbol{t})$ and number of measured samples in file.

Before start of learning algorithm neuronal network user sets, whether program uses both learning period (In first period program uses input signal $\boldsymbol{y}(\boldsymbol{t})$ of neuronal network signal from measured samples on real dynamic system, witch storage in file, in second period program uses directly the signal from the neuronal network output.), or only one. For each

period the user can set number of optimization steps and for each layer of neuronal network can set learning rate α. When learning ends, resulting setting of synaptic weights of neuronal network is stored into the relevant file. It is possible to perform the storage of the actual state into this file during optimization, when user this storage sets and sets the relevant number of steps too. During optimization the actual optimization step and the actual square error between neuronal network and real dynamic system are displayed on the panel.

The last step is storage of calculated response to the input signal of learned neuronal network. User chooses, whether program will use the input signal $y(t)$ of the neuronal network directly from the output of neuronal network or signal from measured samples of the real dynamic system.

4 CONCLUSION

This paper has introduced readers to our programm for location of model of dynamic system by means of feedforward neuronal network, with backpropagation algorithm. Its functionality has been experimentally attested, for the time being for simple trust dynamics systems.

Created program serves as an instrument for teaching of the theory and for using of the neuronal network and it can be certainly used in technical practice.

In the future we think about using the modificated algorithms for learning of the neuronal network. Next purpose is using of complicated paradigm of neuronal network in this model and facilitation on-line response of model on input signal.

Our program has been created in terms of study how to use the neuronal network for time signal processing.

REFERENCES

[1] V. Záda, D. Zelenka: „Parametrical optimization of matrix controllers“, Proceedings the 4th International Scientific-Technical Conference: PROCESS CONTROLL, ŘÍP 2000, Kouty nad Desnou 2000, Czech republic, ISBN 80-7194-271-5

[2] V. Záda, D. Zelenka: „Optimální nastavení parametrů maticových regulátorů, PROCEEDINGS, VIII. International Conference on the Theory of Machines and Mechanisms, TU Liberec 2000, Czech republic, ISBN 80-7083-418-8

[3] M. Razím, J. Štecha: „Nelineární systémy“, script ČVUT Praha, fakulta elektrotechnická 1997, Czech republic, ISBN 80-01-01663-3

[4] S.Kubík, Z. Kotek, V. Strejc, J. Štecha.: „Teorie automatického řízení I. – Lineární a nelineární systémy“, SNTL Praha 1982, Czech republic

[5] M. Novák: „Umělé neuronové sítě: teorie a aplikace“, C. H. BECK Praha 1998, Czech republic, ISBN 80-7179-132-6

[6] J. Šíma, R. Neruda: „Teoretické otázky neuronových sítí“, MATFYZPRESS Praha 1996, Czech republic, ISBN 80-85863-18-9