UDC 629.423.1

MODELING OF ARTIFICIAL NEURAL NETWORKS IN THE MATLAB ENVIRONMENT

Mirabror Vakhidov
PhD student,
Tashkent State Transport University,
Republic of Uzbekistan, Tashkent,
E-mail: mvx1995jd@gmail.com
Otabek Khamidov
doctor of technical sciences, professor
Tashkent State Transport University,
Republic of Uzbekistan, Tashkent,

Annotation: *This article solves the problem of developing a template that allows you to get acquainted with the process of creating and training, as well as predicting the results of neural networks in the Matlab system. The end result of the work is designed to help students of technical specialties and master the initial stage of working with neural networks. To implement the task, the MatlabR2013b system is used*

Key words: *neural network, modeling, weights, computing, optimization, input, hidden layer, output.*

Introduction

The theory and apparatus of artificial neural networks (ANN) and neural network models (NSM) are an actively developing area of science and technology. The main prospects for their use are related to the solution of complex practical problems. In telecommunication systems, they are used to solve the following important problems [1, 2]: switching control, adaptive routing, traffic control in telecommunication networks, planning of mobile radio networks, optimal distribution of channels in cellular radio networks.

The solution of any problem using ANN and NSM includes the following steps:

- development of a neural network model;
- formation of the input and desired output signals of the NSM;
- generation of error signal and optimization functionality;
- formation of the structure of the NSM, adequate to the problem being solved;
- development of an NSM tuning algorithm equivalent to the process of solving a problem in a neural network logical basis;
- solution of the problem using the developed NSM.

Implementation of ANN in the mathematical package MATLAB

Artificial neural networks are widely used to solve various problems. Among the developing areas of application of ANNs are the processing of analog and digital signals, the synthesis and identification of telecommunication systems. The

fundamentals of the theory and technology of ANN application are widely presented in the MATLAB package. In this regard, the latest version of the package, MATLAB 6.0, should be especially noted, which for the first time presents a graphical user interface GUI (Graphical User Interface) for ANN - NNTool.

This article describes NNTool and shows the technique of its application in solving a number of problems in the synthesis of circuits and communication networks [3].

Application in telecommunication systems is carried out using either ANN without memory or ANN with memory. In both cases, the key element is an ANN without memory, a similar role of which is determined by the fact that when neurons with certain activation functions (transfer characteristics) are used, the ANN is a universal approximator. The latter means that in a given range of input variables, the ANN can reproduce and model an arbitrary continuous function of these variables with a given accuracy. Below we discuss issues related to the so-called forward-propagation ANNs, i.e., without feedback. A remarkable property of such ANNs is their stability.

After the INS structure is selected, its parameters must be set. The choice of ANN structure and types of neurons is an independent and rather difficult issue, which we will not discuss here. As for the values of the parameters, as a rule, they are determined in the process of solving some optimization problem. This procedure in ANN theory is called learning.

The NNTool graphical user interface allows you to select ANN structures from an extensive list and provides a variety of learning algorithms for each type of network.

We consider the following issues related to working with NNTool:

- assigning graphical controls;
- data preparation;
- creation of a neural network;
- network training;
- network run.

An example of creating a neural network

Let it be required to create a neural network that performs the logical function "AND".

We choose a network consisting of one perceptron with two inputs. In the process of network training, input data are fed to its inputs and the value obtained at the output is compared with the target (desired). Based on the result of the comparison (deviation of the obtained value from the desired one), the weight changes and biases are calculated to reduce this deviation.

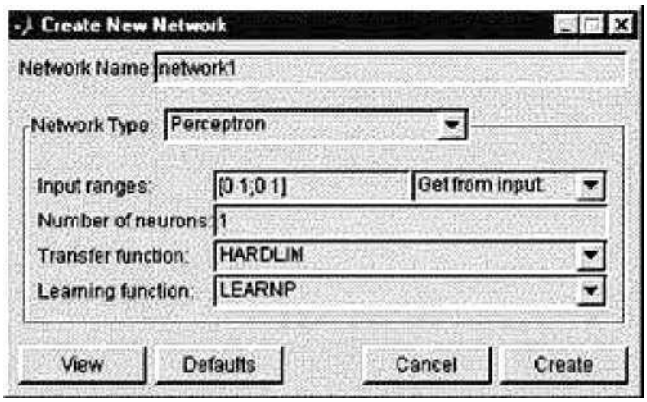
So, before creating a network, it is necessary to prepare a set of training and target data. Let's make a truth table for the logical function "AND", where P1 and P2 are inputs, and A is the desired output (table).

Truth table of the logical function "AND"

P	P	A
0	0	0
0	1	0
1	0	0
1	1	1

The data can be represented by any expression that MATLAB understands. For example, the previous definition of a target vector could be equivalently replaced by a string of the form bitand([0 0 1 1], [0 1 0 1]).

Now you should start creating a neural network (Fig. 1).



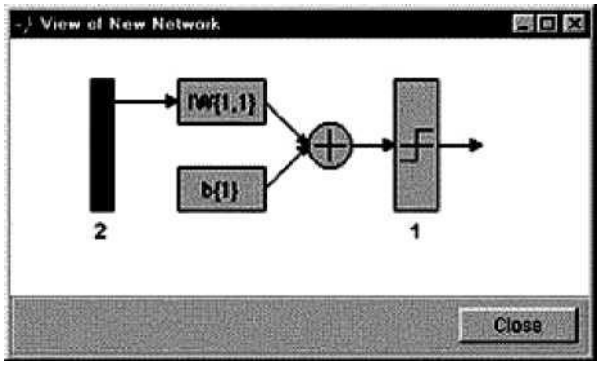
Rice. 1. Window "Create a network"

The fields carry the following semantic loads:

- network name - this is the name of the object of the network being created;
- network type (Network Type) - defines the type of network and, in the context of the selected type, presents various parameters for input in the part of the window located below this item. Thus, for different types of networks, the window changes its content;
- input ranges - a matrix with the number of rows equal to the number of network inputs. Each row is a vector with two elements: the first is the minimum value of the signal that will be applied to the corresponding input of the network during training, the second is the maximum. To simplify the input of these values, a drop-down list “Get from input” is provided, which allows you to automatically generate the necessary data by specifying the name of the input variable;
- number of neurons - the number of neurons in the layer;
- transfer function (Transfer function) - in this paragraph, the transfer function (activation function) of neurons is selected;
- learning function - the function responsible for updating the weights and biases of the network during the learning process.

You can see the architecture of the network being created, i.e. we have the opportunity to make sure that all the actions were performed correctly. On fig. 2 shows a perceptron network with an output block that implements a hard-limited transfer

function. The number of neurons in the layer is one, which is symbolically represented by the dimension of the column vector at the output of the layer and indicated by the number directly below the transfer function block. The network under consideration has two inputs, since the dimension of the input column vector is equal to two.



Rice. 2. Preview of the created network

So, the structure of the network corresponds to the task.

As a result of the performed operations, an object with the name network1 will appear in the "Networks" section of the main window of NNTool.

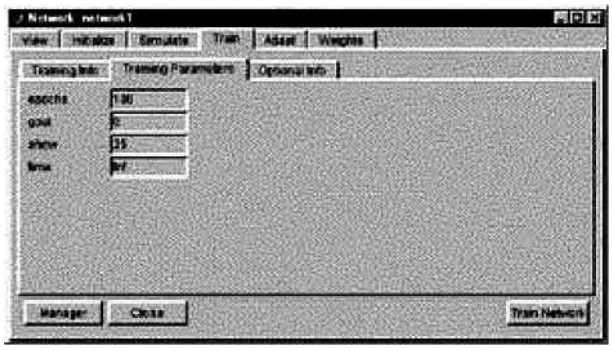
Our goal is to build a neural network that performs the function of a logical "AND". Obviously, one cannot count on the fact that immediately after the network creation stage, the latter will provide the correct result (the correct input / output ratio). To achieve the goal, the network must be properly trained, i.e., it is necessary to select the appropriate parameter values. MATLAB implements most of the well-known algorithms for training neural networks, among which there are two for perceptron networks of the type under consideration. Let's return to the main window of NNTool. At this stage, the bottom panel "Networks only" is of interest. Pressing any of the keys on this panel will bring up a window with many tabs presenting the network parameters necessary for its training and running, as well as reflecting the current state of the network.

You can complete the learning process, guided by different criteria. There are situations when it is preferable to stop training, assuming that a certain time interval is sufficient. On the other hand, the level of error is an objective criterion.

On the Training parameters tab for our network (Fig. 3), you can set the following fields:

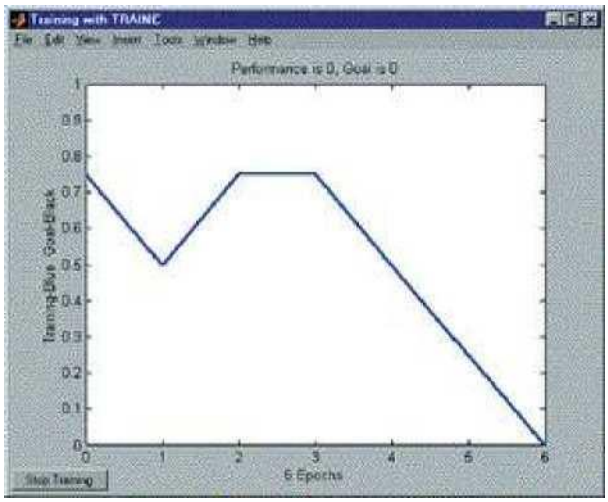
- the number of epochs (epochs) - determines the number of epochs (time interval) after which training will be terminated (an epoch is a single presentation of all training input data to the network inputs);
- achievement of the goal or hit (goal) - here the absolute value of the error function is set, at which the goal will be considered achieved;
- update period (show) - the update period of the learning curve graph, expressed as the number of epochs;
- training time (time) - after the time interval specified here, expressed in

seconds, training stops.



Rice. 3. Training Options Tab

Taking into account the fact that for problems with linearly separable sets (and our problem belongs to this class) there always exists an exact solution, we set the goal achievement threshold equal to zero. Leave the rest of the parameters at their default values. We only note that the training time field contains the record Inf, which defines an infinite time interval (from the English Infinite - infinite) (Fig. 4).



Rice. 4. Learning curve

It should be noted that for perceptrons that have a hard-limited activation function, the error is calculated as the difference between the target and the resulting output.

So, the learning algorithm has found the exact solution to the problem. For methodological purposes, we will verify the correctness of the solution of the problem by running the trained network. In this task, it is natural to use the same data set as in training data1. It should be noted that the network is created initialized, i.e. the values of weights and biases are set in a certain way. Before each next learning experience, the initial conditions are usually updated.

When choosing a neural network for solving a particular problem, it is difficult to predict its order. If you choose an unreasonably large order, the network may be too flexible and may represent a simple dependency in a complex way. This phenomenon is called overfitting. In the case of a network with an insufficient number of neurons,

on the contrary, the required level of error will never be reached. There is an over-generalization here.

The following technique is used to prevent overfitting. At the beginning of the operation, the network errors on the training and control sets will be the same. As the network is trained, the learning error decreases, and as training reduces the actual error function, the error on the control set will also decrease. If the control error has ceased to decrease or even began to grow, this indicates that training should be completed. Stopping at this stage is called early stopping.

Thus, it is necessary to carry out a series of experiments with different networks before a suitable one is obtained. At the same time, in order not to be misled by the local minima of the error function, each network should be trained several times.

If, as a result of successive training and control steps, the error remains unacceptably large, it is advisable to change the neural network model (for example, complicate the network by increasing the number of neurons, or use a different type of network). In such a situation, it is recommended*Xia*apply one more set - a test set of observations (Test Data), which is an independent sample from the input data. The final model is tested on this set, which provides an additional opportunity to verify the reliability of the results. Obviously, to play its part, the test set should only be used once. If it is used to correct the network, it will actually turn into a control set.

We have considered the simplest problems of the synthesis of circuits and networks, for the solution of which the most popular neural networks of direct propagation were used. In fact, NNTool allows you to solve a much wider range of problems, providing the ability to use networks of various architectures, with and without memory, with and without feedback. At the same time, it should be borne in mind that success largely depends on understanding the behavior of the constructed networks and their approximation capabilities.

Conclusion

In recent decades, the world has seen the rapid development of neuro-information technologies. The relevance of research in this direction is confirmed by a large number of different applications of neuro-information systems. These are automation of image recognition processes, adaptive control, approximation of functionals, forecasting, creation of expert systems and many other applications.

Of particular interest is the use of neuro-information technologies in info-telecommunication systems. The high efficiency of neuro-information technologies in solving problems of adaptive control of dynamic systems in the near future will make them indispensable in the creation of new generations of mobile communication networks and other wireless networks.

The areas of application of ANN and NSM in telecommunication systems are constantly expanding. The appearance on the market of inexpensive multifunctional and task-oriented neurochips and neurocomputers will facilitate a rapid transition to

the next stage in the development of telecommunication systems - the creation of intelligent telecommunication systems.

BIBLIOGRAPHY:

1. *Komashinsky V.I., Smirnov D.A.*Neural networks and their application in control and communication systems. - M.: Hot line - Telecom, 2003. - 94 p.
2. *Semeykin V.D.*The main directions of application of neuroinformation technologies in communication systems // Intern. informatization forum (IFI-2005). International congress “Communication technologies and networks (CTN-2005). - M.: MTUSI, 2005. - S. 52-54.
3. *Neural networks*. STATISTICS Neural Networks. - M.: Hot line - Telecom, 2000. - S. 392.