ANNALS OF FACULTY ENGINEERING HUNEDOARA
– INTERNATIONAL JOURNAL OF ENGINEERING
Tome IX (Year 2011). Extra Fascicule. (ISSN 1584 – 2673)

[1.] Manuela PĂNOIU, [2.] Caius PĂNOIU, [3.] Anca IORDAN, [4.] Cosmina ILLES

SOFTWARE PACKAGE FOR ANALYSIS THE PERFORMANCES OF BACKPROPAGATION NEURAL NETWORKS TRAINING ALGORITHM

[1-4] UNIVERSITY POLITEHNICA TIMISOARA, FACULTY OF ENGINEERING HUNEDOARA,
ELECTRICAL ENGINEERING AND INDUSTRIAL INFORMATICS DEPARTMENT, ROMANIA

ABSTRACT: In this paper we present a software package use to analysis the performances of backpropagation neural networks training algorithm. It was analyze the dependency of performances with the learning rate of the algorithm. The software was implemented in Java language. The software was designed with the aim of offering a very easy-to-use user interface. It can be use as an educational tool for teach the basic concepts of backpropagation neural networks.
KEYWORDS: neural networks, Java, learning algorithm, backpropagation algorithm

❖ INTRODUCTION

Artificial neural networks (ANNs) are tools that have proved to be valuable to solve complex problems in many different application fields of science and technology [1, 2]. As a result, it is increasingly usual to find notions of neural networks included in the curricula of many Engineering studies [1].

Neural networks are interdisciplinary and have been used extensively in various fields ranging from electrical engineering to computer science from biology to image processing. Neural networks can solve prediction, estimation, classification, clustering, forecasting, control and decision making problems accurately and quickly [2].

Studying and researching neural networks, results the mathematical nature and underlying complexity. For this reason many students find the neural networks behavior difficult [9]. Moreover, neural networks are dynamic systems, which evolve in time, especially during their design and training phases. For engineering students, the most important is to understand this dynamic nature, and the real utility and operation of neural networks. Classroom-based teaching, books, and lecture notes are not sufficient to transmit this kind of knowledge [1].

Neural Networks have ability to learn from its environment and improve the performance through learning. The procedure used to selflearning process of an NN is called a learning algorithm. Some important application areas of neural networks are: Engineering and industrial applications, Business and financial applications, Medical applications. Some engineering and industrial applications are [2]: control, monitoring and modeling; process engineering; technical diagnosis; nondestructive testing; power systems; robotics; transportation; telecommunications; remote sensing; image processing.

The use of neural networks offers the many useful properties and capabilities [2]: nonlinearity; adaptivity; contextual information; fault tolerance; uniformity of analysis and design; neurobiological analogy. In this paper is described an educational software useful for a better understanding the basic concepts of backpropagation neural networks and their training algorithm.

❖ THE INFORMATICS SYSTEM DESIGN AND IMPLEMENTATION

The application was implemented in Java using NetBeans platform. The graphical user interface for this application is presented in figure 1. Using main menu for the application you could select the main options that are available.

The first and second options present the principle of the simplest artificial neuron operating. The simplest model of artificial neuron is represented by a nonlinear limiting or threshold element which excites if their inputs activation, fig. 2.

In according to threshold-logic units model each of continuous-valued input signals shall represent the electrical activity on the corresponding input line, or alternatively, the momentary frequency of neural impulses delivered by another neuron to this input.

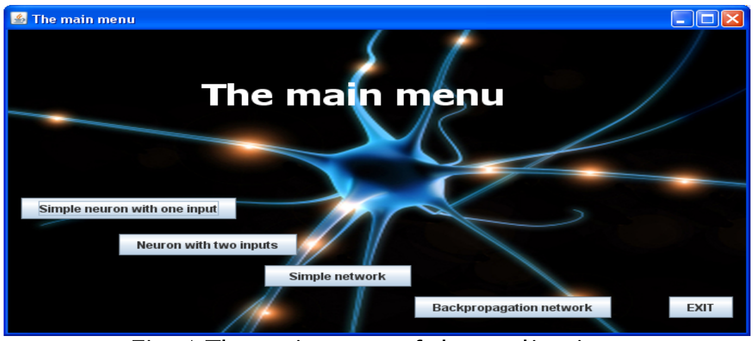


Fig. 1 The main menu of the application

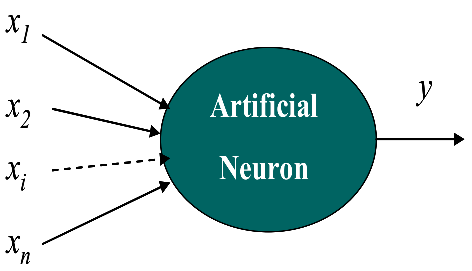


Fig. 2 The simplest model of the artificial neuron

The output frequency *y* is approximated by a function

$$y = c \cdot f\left(\sum_i x_i w_i - \theta\right), \tag{1}$$

where f() is the Heaviside function:

$$f(\cdot) = \begin{cases} 1, \; for \;\; positive \;\; argument \\ 0, \; for \;\; negative \;\; or \;\; zero \;\; argument \end{cases} \tag{2}$$

The principle of a single input neuron operating is present in fig. 3a and in fig. 3 b the principle of multiple inputs neuron operating.

In both these windows it can be separately select the input, the threshold, the weight and the activation function.

For implement this software package it was design a Java class, Neuron, was implemented. This Java class was use in all modules of the package software.

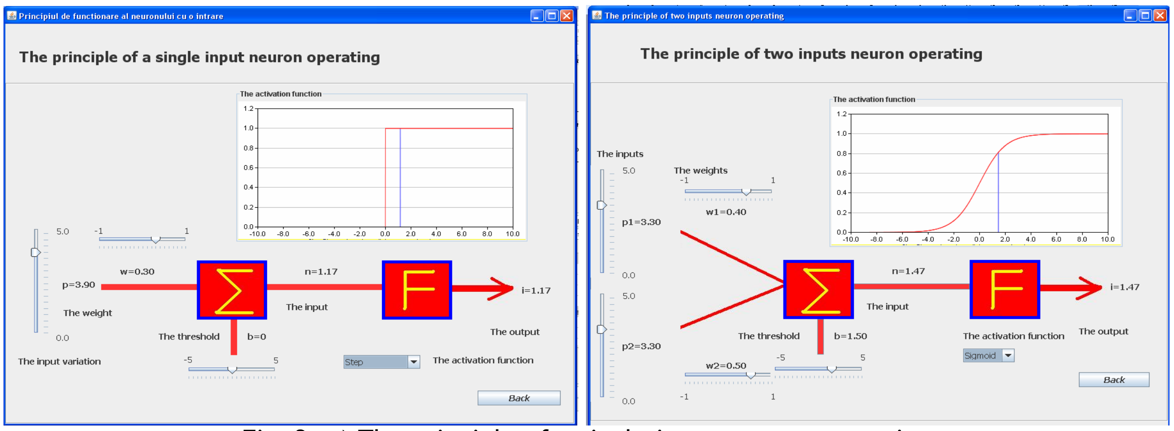


Fig. 3. a) The principle of a single input neuron operating
b) The principle of two inputs neuron operating

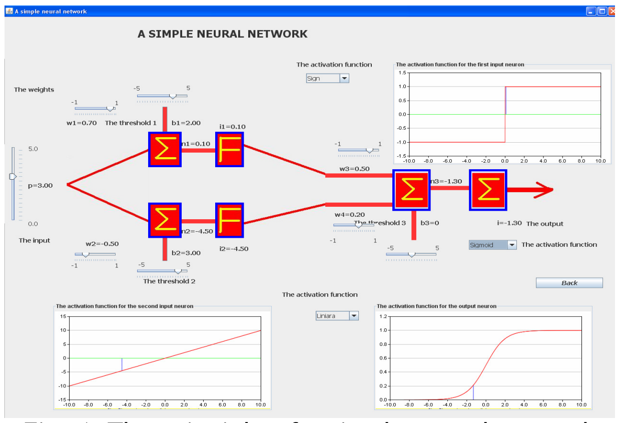


Fig. 4. The principle of a simple neural network operating

The “Simple network” option is design in order to analyze a simple neural network operating mode. Like the other two previous options it can be separately select the input, the threshold, the weight and the activation function for each neuron.

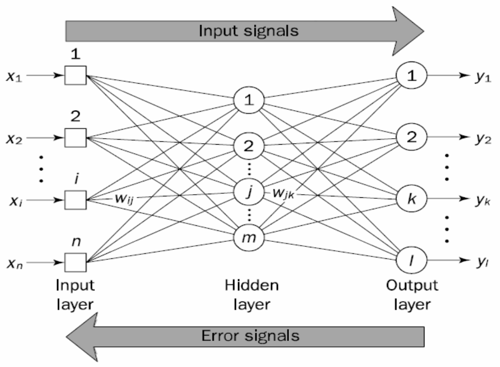


Fig. 5 A three layer backpropagation network

The last option of the software package is design in order to study the backpropagation network and the training algorithm for this network.

A BackPropagation network consists of at least three layers of units: an input layer, at least one intermediate hidden layer, and an output layer (see Figure 5) Typically, units are connected in a feedforward fashion with input units fully connected to units in the hidden layer and hidden units fully connected to units in the output layer. When a BackPropropagation network is cycled, an input pattern is propagated forward to the output units through the intervening input-to-hidden and hidden-to-output weights [3].

With BackPropropagation networks, learning occurs during a training phase in which each input pattern in a training set is applied to the input units and then propagated forward. The pattern of activation arriving at the output layer is then compared with the correct (associated) output pattern to calculate an error signal. The error signal for each such target output pattern is then back propagated from the outputs to the inputs in order to appropriately adjust the weights in each layer of the network. After a BackPropagation network has learned the correct classification for a set of inputs, it can be tested on a second set of inputs to see how well it classifies untrained patterns. Thus, an important consideration in applying BackPropagation learning is how well the network generalizes [3]. Backpropagation, or propagation of error, is a common method of teaching artificial neural networks how to perform a given task [4]. The backpropagation algorithm is:

Step 1. Initialisation.

Set all the weights and threshold levels of the network to random numbers uniformly distributed inside a small range[5], typically (-2.4/Fi , 2.4/Fi), where Fi is the total number of inputs of neuron i;

Step 2. Activation

Activate the back-propagation neural network by applying inputs $x_1(p)$, $x_2(p)$,…, $x_n(p)$, and desired outputs $y_{d,1}(p)$, $y_{d,2}(p)$,…, $y_{d,n}(p)$,

(a) Calculate the actual outputs of the neurons in the hidden layer:

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p)\cdot w_{ij}(p) - \theta_j\right] \tag{3}$$

(b) Calculate the actual outputs of the neurons in the output layer

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p)\cdot w_{jk}(p) - \theta_k\right] \tag{4}$$

Step 3: Weight training

Update the weights in the back-propagation network propagating backward the errors associated with output neurons.

(a) Calculate the error gradient for the neurons in the output layer

$$\delta_k(p) = y_k(p)\cdot[1 - y_k(p)]\cdot e_k(p) \tag{5}$$

(b) Calculate the weight corrections

$$\Delta w_{jk}(p) = \alpha\cdot y_j(p)\cdot\delta_k(p) \tag{6}$$

Update the weights at the output neurons

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \tag{7}$$

(c) Calculate the error gradient for the neurons in the hidden layer

$$\delta_j(p) = y_j(p)\cdot[1 - y_j(p)]\cdot\sum_{k=1}^{l}\delta_k(p)\cdot w_{jk}(p) \tag{8}$$

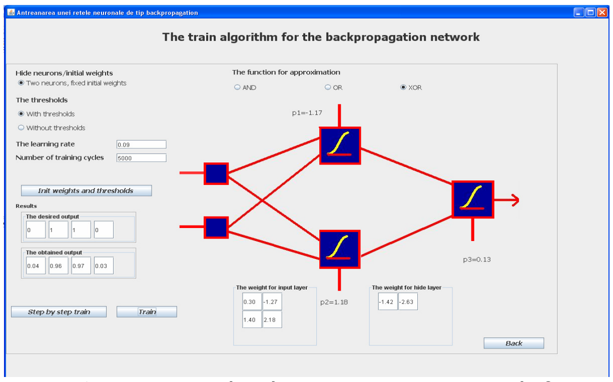
Calculate the weight corrections and update the weights at the hidden neurons

Step 4: Iteration

Increase iteration p by one, go back to Step 2 and repeat the process until the selected error criterion is satisfied.

In fig. 6 is show the graphical user interface for training algorithm. As can see, it can be selected some options, the learning rate, the number of training cycles and the logical function for approximation.

In the figures 7, 8 and 9 are the results of a study that show the dependency of mean square error to the learning rate for 500, 3000 and 5000 training cycles.


Antreanarea unei retele neuronale de tip backpropagation
Init weights and thresholds
Step by step train
Train
Back

The train algorithm for the backpropagation network

Hide neurons/initial weights
[✓ Two neurons, fixed initial weights]

The function for approximation: AND OR [✓ XOR]

The thresholds
[✓ With thresholds] Without thresholds

The learning rate: 0.09

Number of training cycles: 5000

p1=-1.17

p2=1.18

p3=0.13

Results

The desired output: 0 1 1 0

The obtained output: 0.04 0.96 0.97 0.03

The weight for input layer

0.30	-1.27
1.40	2.18

The weight for hide layer

-1.42	-2.63

Fig. 6 Training a backpropagation network for logical function approximation

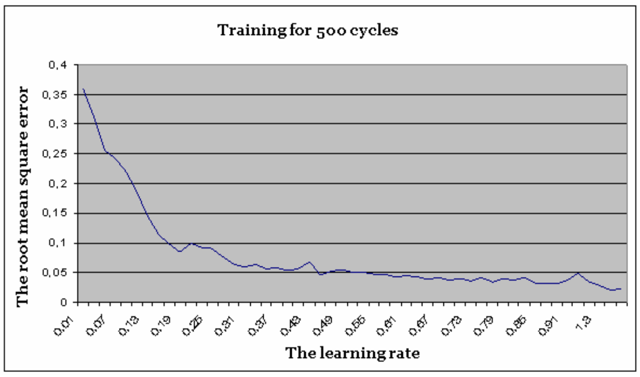


Fig. 7. The dependency of mean square error to the learning rate for 500 cycles

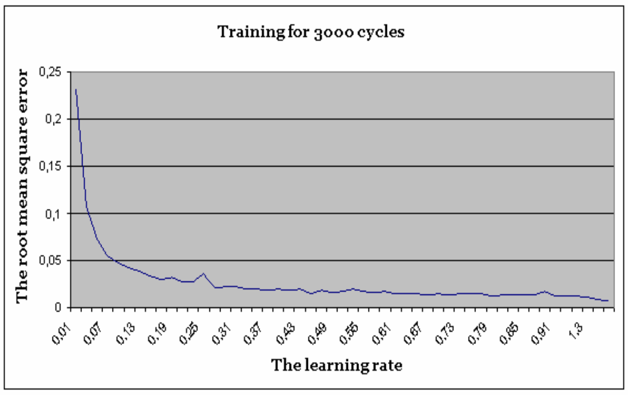


Fig. 8. The dependency of mean square error to the learning rate for 3000 cycles

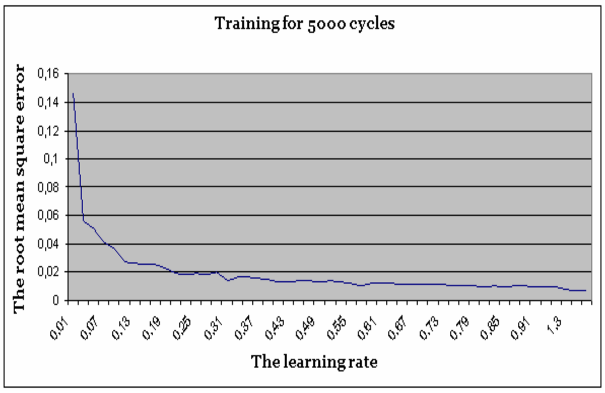


Fig. 9. The dependency of mean square error to the learning rate for 5000 cycles

❖ CONCLUSIONS

In this paper an interactive software package was describe. The software can be use as a educational tool for a better understand of neural networks especially of backpropagation neural networks. It was also present some results of running the backpropagation algorithm and the dependency of mean square error to the learning rate of the algorithm. Using object oriented design and Java language allowed us to ease extendibility for the software.

❖ REFERENCES

[1.] Emilio Garcia Rosello , Jose B. Garcia Perez-Schofield, Jacinto Gonzalez Dacosta, Manuel Perez-Cota, Neuro-Lab: A Highly Reusable Software-Based Environment to Teach Artificial Neural Networks, Computer Application in Engineering Education, vol 11, Issue 2, pp 93-102, 2003
[2.] Aybars Ugur, Ahmet Cumhur Kinaci , A Web-Based Tool For Teaching Neural Network Concepts, Computer Application in Engineering Education, Volume 18, Issue 3, pages 449–457, September 2010
[3.] http://www.itee.uq.edu.au/~cogs2010/cmc/chapters/BackProp/
[4.] http://en.wikipedia.org/wiki/Backpropagation
[5.] Michael Negnevitsky - Artificial Intelligence A Guide to Intelligent Systems, editura Addison-Wesley,2005