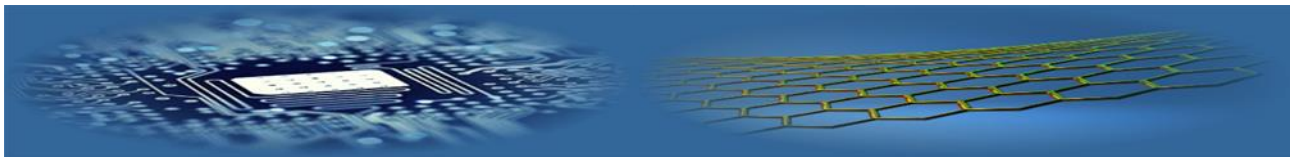


Comparative analysis of simulators for neural networks Joone and NeuroPh

E Zdravkova, N Nenkov*

Faculty of Mathematics and Informatics, University of Shumen "Episkop Konstantin Preslavsky"
115, Universitetska St., Shumen 9712, Bulgaria

**Corresponding author's e-mail: elitca_spasova@abv.bg, naydenv@shu-bg.net*

Received 1 March 2016, www.cmnt.lv

Abstract

This paper describes a comparative analysis of two simulator neural networks - Joone and NeuroPh. Both simulators are object-oriented and java - based. The analysis seeks to show how much these two simulators are similar and how different in their characteristics, what neural networks is suitable to be made through them, what are their advantages and disadvantages, how they can be used interchangeably to give certain desired result. For the purpose of comparative analysis of both the simulator will be realized logic function, which is not among the standard, and relatively complex and is selected as a combination of several standard logical operations.

Keywords:

neural network
neural network simulator
logical function
exclusive OR
neural network architecture

1 Introduction

Both the simulator selected for the study are Java - based and object - oriented simulators. The used simulators are Joone 4.5.2 and NeuroPh 2.92. Joone is object - oriented frameworks allows to build different types of neural networks. It is built on combining elements which can also be expanded to build new training algorithms and architecttures for neural networks. The components are interchangeable with programming code modules that connect to be performed on the data stream and be deriving obtained information and relevant results. New components that the user adds, can be planned and again. Beyond simulation, Joone has opportunities for multiplatform deployment. Joone has a graphical editor for graphically deployment and testing of each neural network, and the teaching and testing of many examples, the network is configured and can be trained even from multiple remote machines. As of 2010 Joone, NeuroPh and Encog are main component - based environments for neural networks of java - platforms. [1, 2]

Joone can be considered not only as a simulator of neural networks such frameworks were, but as a fully integrated development environment. Unlike its trading partners, it has a strong focus on the code, building a neural network, but not on the visual design. In theory Joone can be used to build a wide range of adaptive systems (including those with mal-adaptive elements) but generally his focus is on building backpropagation - neural networks.

NeuroPh is lightweight frameworks allowed to simulate neural networks. It is java - based and can be use basic for the development of standard types of neural network architectures. It contains well designed open source library and a small number of core classes that correspond to basic concepts in neural networks. There is a good graphics editor toquickly build java - based components of neural networks [3].

2 Methodology

To be tested and analyzed both the simulator will realize logical function, and which is relatively complex and is not among the standard. The generated neural network calculate the result of the following logical function:

((((A XOR B) AND C) OR D) → ((((E XOR F) AND G) OR H)) ↓ I)) AND (((J XOR K)↓(L XOR M)) OR (N ↓O))

The logical function implemented in the neural network includes several logical operators:

- AND - Provides value 1 (TRUE) if both inputs are 1 argument.
- OR - Provides value 1 (TRUE) if at least one input argument is 1.
- XOR - EXCLUSION OR - Provides value 1 (TRUE) if the two input arguments are different.
- → (implication) - Provides a value of 0 (FALSE) if the first argument is 1, the second 0. In other cases, 1.
- ↓ (Peirce's arrow NOR) - negation of the disjunction (in logical circuits can be represented as a combination OR - NOT). Gives a value of 1 (TRUE) if both input argument is

First we will make the realization of simulator neural networks Joone.

The neural network is built by JOONE 4.5.2.0.
Contains: an input layer - Input - 15 neurons,
4 hidden layers respectively:
I 15 neurons
II with 10 neurons
III 8 neurons
IV with 15 neurons
And one output layer 10 neurons.
Learners data are taken from the file: xor1. Txt
Test examples are taken from the file: test_pattern.txt
The results are recorded in the file: results.txt

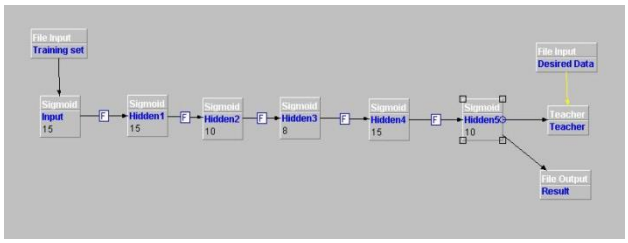


FIGURE 1 Layers of the neural network in Joone

Development of the model to reach the optimal architecture is as follows:

First, we test neural network with I hidden layer with 15 neurons and output layer with one neuron:

After 10,000 epochs: RMSE - 7.75

After 20,000 epochs: RMSE - 6.83

Then, we test neural network with II hidden layer (15 neurons - 10 neurons), and an output layer with 1 neuron:

After 10,000 epochs: RMSE - 4.64

After 20,000 epochs: RMSE - 3.97

It is seen that the search it textures do not give a good enough value of error. Therefore, the development of the model should be continued. This is done by testing successively on following architectures:

II hidden layers (15 neurons - neurons 15) and output layer with one neuron:

After 10,000 epochs: RMSE - 5.57

After 20,000 epochs: RMSE - 4.77

III hidden layer (15 neurons - 10 neurons - neurons 10) and an output layer with 1 neuron:

After 10,000 epochs: RMSE - 4.57

After 20,000 epochs: RMSE - 3.93

III hidden layers (15 neurons - neurons 10 - 8 neurons) and output layer with one neuron:

After 10,000 epochs: RMSE - 4.27

After 20,000 epochs: RMSE - 3.03

IV hidden layers (15 neurons - neurons 10 - 8 neurons - neurons 8) and output layer with one neuron:

After 10,000 epochs: RMSE - 4.51

After 20,000 epochs: RMSE - 3.88

IV hidden layers (15 neurons - 10 neurons - neurons 8 - 10 neurons) and output layer with one neuron:

After 10,000 epochs: RMSE - 4.38

After 20,000 epochs: RMSE - 3.76

IV hidden layers (15 neurons - neurons 10 - 8 neurons - neurons 15) and output layer with one neuron:

After 10,000 epochs: RMSE - 4.18

After 20,000 epochs: RMSE - 3.57

IV hidden layers (15 neurons - neurons 10 - 8 neurons - neurons 15) and output layer 10 neurons:

After 10,000 epochs: RMSE - 0.001

It is seen that the last result is satisfying and we can stop the development of the model.

The learning set consists of 100 training examples that we taken from the file xor1.txt. Test cases are 20 and are located in the file test_pattern.txt.

The following table shows the results of testing with learning set. Below are the input values - from 1 to 15, the expected result of these inputs and the actual result. The expected result is determined by a complex logical function, which we described above. The actual result is the output of the system - Joone.

TABLE 1 Result on learning set in Joone

Inputs															Outputs	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Desired**	**Actual**
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00185941651318
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00171267854721
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.00174068713052
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.00171297868822
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.00171263203490
1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.99819065530154
0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.99874748651663
1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.99840020882580
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.00171176387902
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00171063441000
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.001711942825887
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00170547142383
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.001704159243449
1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.998690422813351
0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.99825801114161
0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.99859104574956
1.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.99866411324620
0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00170372477138
0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00170383697555
0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00170418058508
0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00170950487012
0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00171284151805
0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00171307490515
0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.99868704088361
0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.99886986914373
0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.99879712659079
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00181573650909
1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00181573650909
1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00181828998258
1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.00196073740338
1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.99831946230466
1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.998699389310955
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.998776103873972
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.996709046387602
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.004418785357284
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00182640654367
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.00179886966065
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.00178497066089
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.00178017710286
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.001723142165490
1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.00170867604999

1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.00171115476091
0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.00194277916844
0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.00180661004916
1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.00170405227369
0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.99872473082876
1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.00345636992553
0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.001706998074978
1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.99820203498989
0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.99876787843303
1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.00201477197411
0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00170808545531
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.001862992242455
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00208965556292
0,0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.99796070484324
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00208866209964
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.001925035898582
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00175139675889
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00172983744967
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0,0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00171707163438
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00194790523692
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.00181551577185
1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00181828998258
0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.998664650256829
0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00231109937107
0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00171578793161
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00172035245579
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.001794173831323
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.00172606629957
1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00196073740338
0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.99875045408016
0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00170605685128
0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00172491821548
0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00172441171231
1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.99876893064456
0.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.998708135281306
0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.99862109900078
0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00171646520616
0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00170500832053
0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00170441208624
0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00170422833047
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00170494386384
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.001707391479248
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00172391240963
0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00179535552546
0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00175517036174
0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.001707701495688
0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.001706150804015
0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.001705704003614
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00170749323605
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.00171540639749
0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.00175856759291
0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.00170792028372
0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.00170627806636
0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.001705806627616
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.00170767275065
0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.99864765385988
1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.998167383168365
0.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.99875043991273
1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.998158499362219

Testing the examples of training set gives result described in the table below.

TABLE 2 Results on training set in Joone

Inputs															**Output**	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Desired**	**Actual**
0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0017054441008677 28
0.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0017365073818188474
0.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0017126790834616611
0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0017221518565343809
0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0017881340101438775
1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.8408661190565192
1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0017213972306672999
1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0018150456091549265
1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0017396394623614357
1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0017129149981214703
0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.022991489166898076
0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0018174232780409597
0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.9987247308287605
0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.9935189616059639
0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1,0	0.0	0.0	1.0	0.0	0.0	1.0	0.8113680710622061
1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0017054508579993635
1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0017108325172039915
1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.00170405227369 8014
1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0017046034834530507
1.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0017054124015384057

In the above table are again described input values which have already from training set. Here we have 20 examples. In the last two columns are the result which must be prepared according to the logical function, and actual result from the system - Joone.

The designed neural network can be used to predict the output of the implemented logic function. It gives good results on learning set and on examples of the test set.

It is follows to implement the same architecture of the neural network simulator NeuroPh.

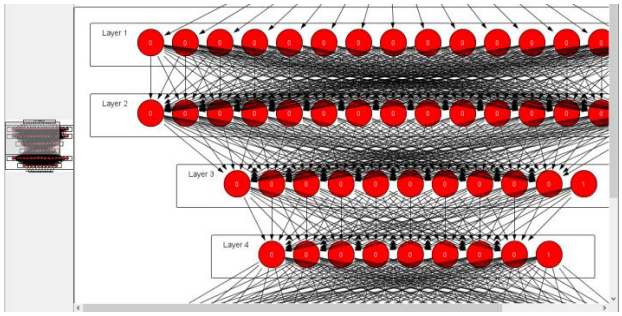


FIGURE 2 Neural network in NeuroPh

The neural network has again four hidden layer neurons. Its structure is as follows - input layer (15 neurons), output layer (10 neurons), I hidden layer (15neurons), II hidden layer (10 neurons), III hidden layer (8 neurons), IV hidden layer (15 neurons).

In testing and training with the same learning set it proved that the neural network is not trained and cannot properly classify examples of the test set. Training and learning set are the same as those used in the previous simulator Joone.

To be completed research we including an example of realization of the standard logical function - exclusive or (XOR). This is a logical function which gives true when one of the input statements is true and the other is false.

Neural network, realize this logical function is shown in the following figure. It has one neuron in the input layer, two neurons in the hidden layer and one neuron in the output layer.

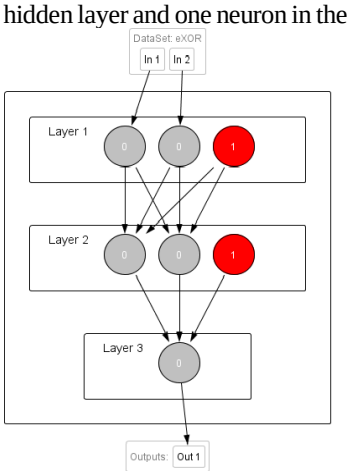


FIGURE 3 Neural network realizes a logical function "exclusive or" in NeuroPh

On the structure of the neural networkis used the results from "Research of simulators for neural networks through the implementation of multilayer perceptron". (Zdravkova, 2015).

The network is fully trained after 1900 iterations. After learning of the network supplying as input values 1 1. The results are as follows - the value of neurons in the hidden layer are 0.101 and 0.902; value of the neuron in the output layer is 0.125. Conduct a second test with values 1, 0. The results are as follows - the value of neurons in the hidden layer are respectively 0.839 and 0.998; value of the neuron in the output layer is 0.871.

We see here that the results are good. Unlike the previous function that failed to give good result after training.

Now test the neural network with the same architecture in simulator Joone [4].

Result after 20,000 epochs: RMSE - 0,29.

The following table shows the input values of the standard logical function and the corresponding result for these input values. The last column shows actual output from the system - Joone.

TABLE 3 results on Exclusive OR in Joone

Inputs		Output	
1	2	**Desired**	**Actual**
0	0	0	0.010592225910198724
0	1	1	0.6618289936765938
1	0	1	0.6718835779208182
1	1	0	0.6819571896978877

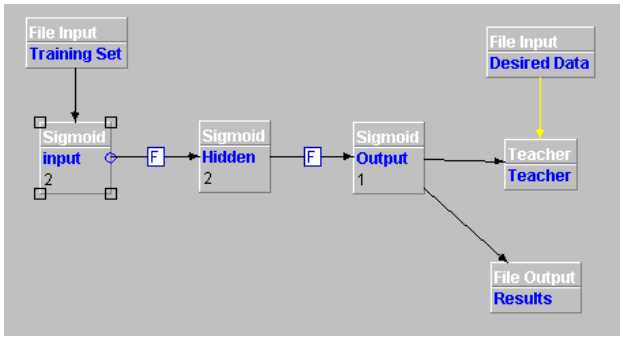


FIGURE 4 Neural network realized "exclusive or" in simulator Joone

It is noted that the results for this logical function here are not as good as in the simulator NeuroPh.

The first logical function here also gave a good result.

3 Conclusions

After the conducted tests it shows that both the simulator for neural networks have some peculiarities. Both simulators are java - based and object - oriented. But they have different results in tests with certain types of neural networks. At the same architecture of the neural network (the same number of layers and neurons) and identical sets of test data, the both simulator give different results.

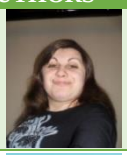
As the results shows, the simulator Joone gives much better results in the testing of arbitrary complex logical function, which is not among the standard. In this simulator there is a very good opportunities to create new types of algorithms and architectures of neural networks.

The simulator NeuroPh does not give good results in tests with random set, complex logic functions. Basic on a tests we can observe, that NeuroPh can be used in standard logic operations and it is suitable for beginners in the creation of neural networks programmers. The simulator Joone can be used by advanced programmers of neural networks.

References

[1] Heaton J T 2005 Introduction to Neural Network with Java *Paperback*
[2] David J Artificial Neural Network. Methods and Applications, Livingstone
[3] Suzuki K Artificial Neural Networks Industrial and control engineering applications
[4] Zdravkova E 2015 Research of simulators for neural networks through the implementation of multilayer perceptron, Information Technologies, Management and Society, *The 13th International Conference inInformation Technologies and Management 2015* 55-6

AUTHORS



Elitsa Spasova, 15.02.1988, Bulgaria

Current position, grades: assistant in Shumen University
University studies: Master in Artificial Intelligence
Scientific interest: Artificial Intelligence, Neural Networks, Genetic Algorithms
Publications: 3

Nayden Nenkov, 22.08.1957, Novi Pazar, Shumen region, Bulgaria

Current position, grades: Vice Dean, Faculty of Mathematics and Computer Science
University studies: Shumen University
Scientific interest: Artificial Intelligence, E-learning, Data mining; Logic Programming
Publications: 63
Experience: 28 years university lecturer