International Journal of Control Theory and Applications

ISSN : 0974–5572

© International Science Press

Volume 10 • Number 23 • 2017

Neural Darwinism Inspired Implementation of an Artificial Neural Network Model

P. Chanthini[a] and K. Shyamala[b]

[a]Research Scholar, PG and Research Department of Computer Science, Dr. Ambedkar Government Arts College (Autonomus), Affiliated to University of Madras, Chennai, India.

E-mail: chanthini19@gmail.com

[b]Associate Professor, PG and Research Department of Computer Science, Dr. Ambedkar Government Arts College (Autonomus), Affiliated to University of Madras, Chennai,India

E-mail: shyamalakannan2000@gmail.com

Abstract : Vast scope in exploration of the biological neural system and brain functions continually open rooms for improvement in Artificial Neural Network (ANN) studies. This work is an effect of an effort in adopting the biological theory "Neural Darwinism: Selection" by GM. Edelman into Artificial Neural Network Model (ANNM). The newly implemented ANNM has provided scopes in designing new ANNMs using different biological theories in addition to the traditional way. This work illustrates an ANNM having two distinct portions, one physically static and other functionally dynamic. Rather than using the conventional method of training for weight adjustment, this model uses static weight representation and dynamic selection of artificial neural representations according to different problems, mimicking a biological neural selection theory- experiential selection. The result of this work depicts the successful implementation of an ANNM through newly adopted theory, solving multiple unipolar problems like XOR and N-Parity problems where the conventional method will require two or more separate feed-forward networks trained for each problem.

Keywords: *Neural Darwinism: Selection, Artificial Neural Network Model, Experiential Selection, Virtual Model, Feed-forward Network, Unipolar Problem.*

1. INTRODUCTION

Artificial neural network is originally designed to approximate biological neuron and also it is one of the Artificial Intelligence tools to implement the cognitive functionalities of human brain like learning, problem-solving, reasoning etc. The efforts started in the direction of developing a variety of network models like feed-forward, feedback, recurrent network and various learning algorithms.

In the period of 1940s and 1950s, formal neuron models, perceptron and associative memories are introduced. Later in the period of 1960s to 1990s, different types of algorithms are evolved and from the year 2000, the efforts started towards optimization and improvisation of existing models [1]. The algorithms developed during this period followed the basic models developed before 1978 when GM. Edelman [2] proposed the theory of “Neural Darwinism: Selection”. As Edelman himself have accepted that the previous biological studies have created advancements in molecular cellular biology and cognitive psychology but have left a large gap in understanding biological bases of psychological phenomena, here also the earlier models have effectively solved different specific problems but have not proceeded in building ANN performing activities like that of a human brain. The following parts of the paper are divided into

1. Section I. A shows analysis of the current neural model and their symmetry with the period before the proposal of the theory “Neural Darwinism: Selection”.
2. Section I. B suggests a variant adaptive neural system highlighting the drawbacks of traditional neural models.
3. Sections II and III depict the related review and the choice of the area, which has the scope to implement the theory.
4. Sections IV and V show the intuition, implementation, and report about a neural system following the theory of “Neural Darwinism: Selection”.
5. Section VI elucidates the results of the work and proposes the scope of improving the current model and also adding further new models implementing rest of the theory proposed by GM. Edelman.

1.1. Physical Model VS. Virtual Model

Every neuron by itself is a functional unit, which receives and responds to its input neuron and its dependent neurons respectively according to their objectives. The computing systems available today, only try to simulate or emulate a biological neural network. The current architecture of the computing systems is with one or minimal high capacity functional units were they manage to create an illusion of imitating the biological neural network to a limited level [3]. Such limitations have diverted the intellects towards specific modeling of neural networks and even towards number crunching which is not found to be the nature of biological neural networks. In the diametrically opposite direction, physical modeling of neural networks using Nano-inductors or functional memories [4] is also under process. For such physical models which have shown to be future possibilities, the virtual models which try to mimic a human brain will always be a requirement to understand the possibilities of construction of physical model architecture.

The above facts are in coherence with the observation by GM Edelman [2] that “Although there is an obvious commonality of neural structure within a species, the degree of individual variation far exceeds that which could be tolerated for reliable performance in any machine constructed according to current engineering principles”.

1.2. Neural Model VS. Neural System

The non-adoptive, application specific and variation less traditional models are developed with following characters.

1. The efforts are in the direction of finding suitable neural models for specific applications.
2. The efforts are in the direction of reducing the totality of neurons specifically in hidden layers to suit the current architecture of computing system.
3. The efforts are in the direction of applying the available neural models to the problems like number crunching in data mining in contrast to its primary objective of mimicking the brain.
4. The efforts are in improving the learning algorithms and weight assignment algorithms just to increase the possibility of convergence.

Any effort to improve parallelism will clearly indicate that the physical neural models are the best in achieving massive parallelism over virtual models. Even though works in physical models are under way and their probable success will not undermine the need of virtual models whose testing and economical utilities are always going to be high. The parallelism in virtual models can be improved in utilizing multi-core systems [5] or cloud environment to a limited level.

The objective of this work is to provide the virtual model of a neural system which is generic, physically static and functionally dynamic one following the experiential selection theory of GM. Edelman. The first step in generalizing a model is taken as a task and the feed-forward network is found suitable in generalization of multiple unipolar (binary) problems.

2. RELATED REVIEW

The Novel approaches for solving the simple non-linear separable problem has random initial weight to the network are not always end up with the global minimum solution [6], the efforts towards adjusting learning rate and momentum also end up with slow convergence or long time to decent towards the point of local minima [7]. In the year 2003, Bodgan M. Wilamowski et al. [8] described hybrid neuron architectures with linear and threshold-like activation function. The authors proposed work shows that the network may have identical weights or a minimal number of weights to be assigned and a number of hidden neurons also reduced according to the problem situations. Iyoda et al. [9] proposed translated multiplicative neuron model for N-Parity problem, which is entirely contrasted to the brain model. In [10], [11] weight initialization method has been improved by using Cauchy's inequality and a linear algebraic method.

A preferable learning algorithm for feed-forward network is Back-propagation learning algorithm but the drawback is very slow convergence [12] – [14], the various efforts are taken to overcome the local minima [15], [16]. All the existing models are purely problem-oriented, which have different models according to the problem domain. So it is decided to construct a virtual generalized neural system which overcomes the above drawbacks.

3. UNIPOLAR FEED-FORWARD NEURAL NETWORK

The work started with the analysis of feed-forward network and hard-limiting activation function opted as most preferable one to the unipolar (binary) problems. Figure 1 shows the hard-limiting activation function. The best performances of them are proved to be after random weight initialization and training them towards specific unipolar problem using Back-propagation learning algorithm.

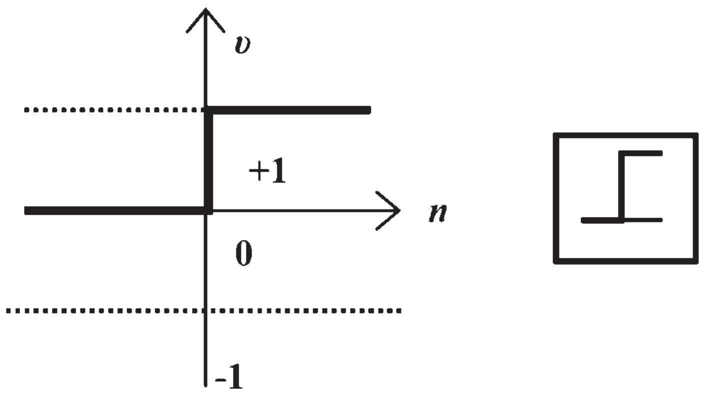


Figure 1: Hard-limiting activation function

$$\upsilon = \text{hardlim}(n)$$

$$\varphi^{hlim}(\upsilon) = \begin{cases} 1 \text{ for } \upsilon > 0 \\ 0 \text{ for } \upsilon \leq 0 \end{cases}$$

The following facts of these networks were identified as the drawbacks in generalizing a model for multiple problems and also improving parallelism.

1. Initializing random weights make parallelization difficult as the different weights are dependent to a randomizing function.
2. The random initialization to every weight that is either from input to hidden layer or hidden layer to output layer and adjusting them to suit to a problem makes every part dynamic so that the adoption to multiple problems is difficult.
3. 100 percentage of convergence is not assured in the Back-Propagation algorithm.
4. Reduction of hidden neurons to the maximum possible level to suit a specific problem is in contrast to the objective of the generalization.

4. PROPOSED SYSTEM

4.1. Intuition Behind The Proposed System

A human brain response to unipolar problems doesn't match a feed-forward neural model which always responds to every input with an output. Whereas the brain seems to produce an output to an input if needed only, were inputs to it are endless and this makes a possible assumption to be done that, the inputs continuously create an effect in the brain but all are not with the response to an output level. That is, the effect created by an input up to a mid-level brain is a standard one, whereas the response to an output needed is a selection of mid-level representation according to our problem situation. Even in biological terms, the neural selection is categorized as a controversial process in contrast to learning procedures [17]. It gave the intuition that generic problems can be represented and responses may be a selection of a subset of generic problem situations.

4.2. Artificial Neural Network Model

With the limitations of feed-forward network analyzed and intuition about the brain gave the following proposals to generalizing a feed-forward network design to suit multiple problem situations. Figure 2 shows the complete design, which represents a generic model for the neural system.

1. The proposed system should suit multiple problems which do not exceed the number of input neurons (n) in the designed system.
2. The number of hidden neurons will be 2^n were any combination of input can be responded to by a hidden neuron.
3. The weights of input to hidden layer neurons are initialized in such a manner that any one of the hidden neurons only will get excited to an input combination.
4. For every problem, let there be an output neuron that selects the hidden neurons to get excited with according to a problem situation. The model can have the possibility to solve 2^{2n} number of problems.

The above model is in coherence with the statements of GM. Edelman [2] under experiential selection: "After most of the anatomical connections of the primary repertoires have been established, the activities of particular functioning neural groups continue to be dynamically selected by ongoing mechanisms of synaptic change driven by behavior and experience. This selection occurs within populations of synapses, strengthening some and weakening others without major changes in anatomy".

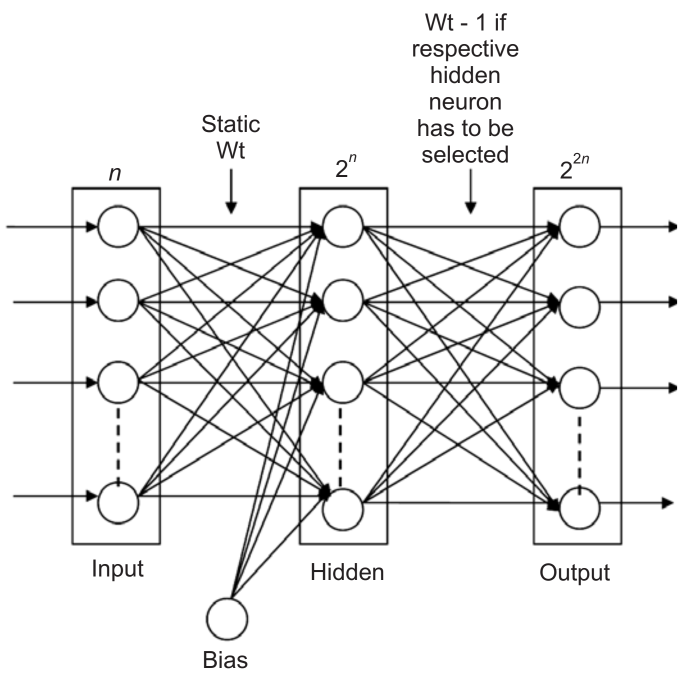


Figure 2: Generic model for neural system

5. RESULTS AND DISCUSSION

The proposed design was tested using octave environment and nnet-0.1.9 package. The work has been started to create the network with the size of four input neuron, 16 hidden neurons. The weight initialization between the input to hidden neurons are considered to be a static part of the network, the selection process is used to train the problem and the output neuron is created according to the number of selected problem.

5.1. Algorithm 1: Weight Initialization

```
for    $i = 0 : hn – 1$
    net.IW{1,1}($i$ + 1,1:in) = (bitget($i$, in:-1:1)*2-1)*0.01;
    net.$b${1,1}($i$ + 1) = 0.5-sum(bitget(i,in:-1:1))*0.01;
end for
```

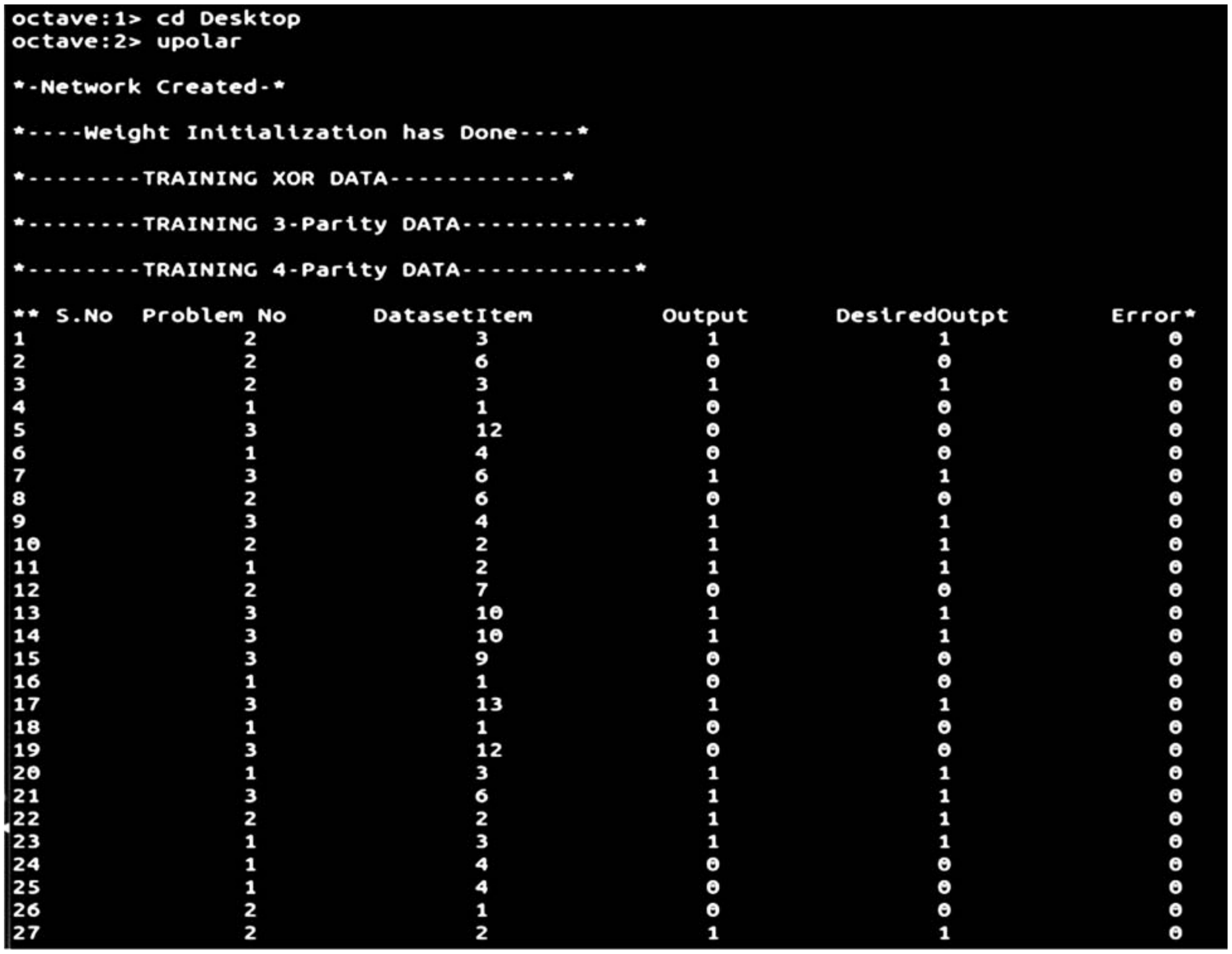
The above script is an assignment of input to hidden and bias to hidden weights ensures only unique neuron fires for a distinct combination of inputs.

The weight assignment from hidden to an output neuron-specific to a unique unipolar (binary) problem eliminates training as it simulates the process of neural selection in the brain. It represents an epoch process, which ensures that the problem is successfully represented in the feed-forward network.

5.2 Algorithm 2: Neural Selection

```
for    $i = 1 : hn$
    if(resu(1, $i$)! = OP(1, $i$))
        net.LW{2,1}($i$) = 1;
    endif
endfor
```

The following octave source and output generated in LINUX environment in an *I3* Machine demonstrates the positive results of the implementation of multiple unipolar (binary) problems. As the initial step, we have taken the problems like XOR, Parity-3, and Parity-4 in the single network through an output neuron representing each of them. The following code will also highlight the network creation and weight assignment from input to hidden layer is done only once. The selection of weights for every output neuron is done once for each problem. From the Figure 3 and 4, the output demonstrates 100% success rate by randomly selecting the problem and also the random input from the dataset.



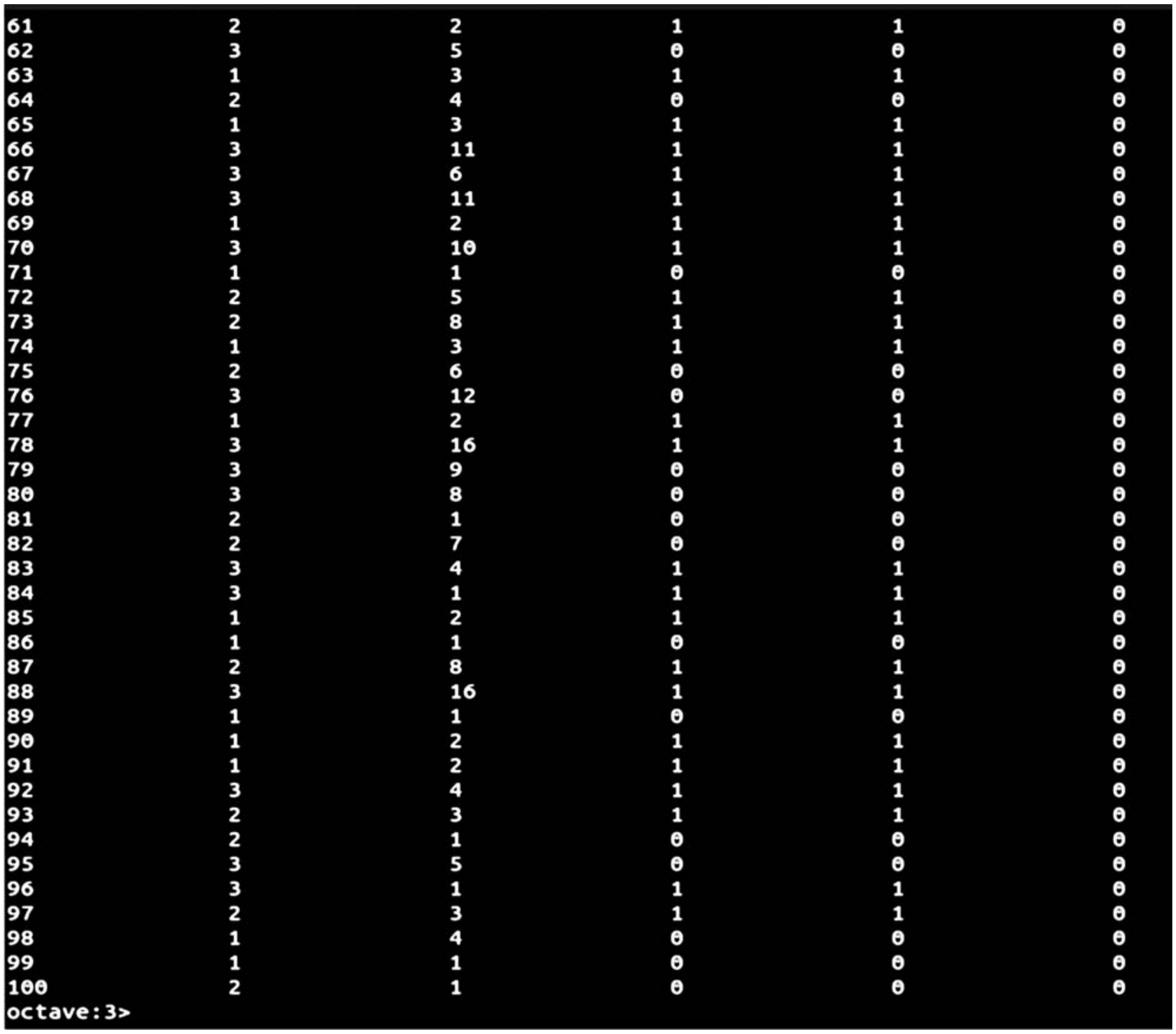
```
octave:1> cd Desktop
octave:2> upolar

*-Network Created-*

*----Weight Initialization has Done----*

*--------TRAINING XOR DATA-----------*

*--------TRAINING 3-Parity DATA-----------*

*--------TRAINING 4-Parity DATA-----------*

** S.No  Problem No       DatasetItem          Output        DesiredOutpt          Error*
1               2               3               1               1               0
2               2               6               0               0               0
3               2               3               1               1               0
4               1               1               0               0               0
5               3               12              0               0               0
6               1               4               0               0               0
7               3               6               1               1               0
8               2               6               0               0               0
9               3               4               1               1               0
10              2               2               1               1               0
11              1               2               1               1               0
12              2               7               0               0               0
13              3               10              1               1               0
14              3               10              1               1               0
15              3               9               0               0               0
16              1               1               0               0               0
17              3               13              1               1               0
18              1               1               0               0               0
19              3               12              0               0               0
20              1               3               1               1               0
21              3               6               1               1               0
22              2               2               1               1               0
23              1               3               1               1               0
24              1               4               0               0               0
25              1               4               0               0               0
26              2               1               0               0               0
27              2               2               1               1               0
```

Figure 3: Random Testing process of selected Problem

5.3. Network Creation to Test Code

```
NP  =  3;  # Number of Problems considered
net  =  createffnet(4, [16 NP]);
net  =  init_wait(net, NP);
net  =  selection("XOR.txt", net); #  Load and train Problem no: 1
net  =  trainnet(net, net.IP, net.OP,1);
net  =  selection("3-Parity.txt",net); # Load and train Problem no: 2
net  =  trainnet(net, net.IP, net.OP, 2);
```

```
                    net  =  selection("4-Parity.txt", net); # Load and train Problem no:3
                    net  =  trainnet(net,net.IP,net.OP,3);
                      A  =  1; B = 3;  # Testing # Problem number from A to B
printf("\n** S.No\t Problem No \t Dataset Item \t Output \t Desired Output \t Error*\n");
                  for i  =  1:100
   # Select Random problem no
                     pn  =  floor((A – 1) + (B – (A – 1))*rand(1)) + 1;
                     di  =  dataitem(pn); # Select Random Data-item
   printf("%d\t", i);
                    net  =  testnet(net, pn, di);
   endfor
```



```
61      2       2       1       1       0
62      3       5       0       0       0
63      1       3       1       1       0
64      2       4       0       0       0
65      1       3       1       1       0
66      3       11      1       1       0
67      3       6       1       1       0
68      3       11      1       1       0
69      1       2       1       1       0
70      3       10      1       1       0
71      1       1       0       0       0
72      2       5       1       1       0
73      2       8       1       1       0
74      1       3       1       1       0
75      2       6       0       0       0
76      3       12      0       0       0
77      1       2       1       1       0
78      3       16      1       1       0
79      3       9       0       0       0
80      3       8       0       0       0
81      2       1       0       0       0
82      2       7       0       0       0
83      3       4       1       1       0
84      3       1       1       1       0
85      1       2       1       1       0
86      1       1       0       0       0
87      2       8       1       1       0
88      3       16      1       1       0
89      1       1       0       0       0
90      1       2       1       1       0
91      1       2       1       1       0
92      3       4       1       1       0
93      2       3       1       1       0
94      2       1       0       0       0
95      3       5       0       0       0
96      3       1       1       1       0
97      2       3       1       1       0
98      1       4       0       0       0
99      1       1       0       0       0
100     2       1       0       0       0
octave:3>
```

Figure 4: Random Testing process of selected Problem

6. CONCLUSION

The intuition of mixed nature of neural formation that is a portion of static interconnection and the other portion being dynamic according to problem situation have been proved positively with the efforts demonstrated above. The work exhibited has demonstrated only generalization, whereas implementation with concurrent responses to multiple problems is the work in progress

Further, the limitations of an octave or any such testing environment led the work towards choosing Java or C++ as an implementation language of such proposals. Java implementation has also been achieved up to testing the same network for multiple unipolar (binary) problems. It will be less to say that there will be very large scope if one tries to mimic the functions of the brain proposed by a biologist.

REFERENCES

[1] P. Alberto, P. Beatriz, et al. (2016). "Neural networks: An overview of early research, current frameworks and new challenges," *Neurocomputing*, vol. 214, pp. 242-268, Nov. 2016.

[2] Edelman, Gerald M. "Neural Darwinism: Selection and reentrant signaling in higher brain function," *Neuron,* vol. 10(2), pp. 115-125, Feb. 1993.

[3] Cox, David Daniel, and Thomas Dean. "Neural networks and neuroscience-inspired computer vision," *Current Biology,* vol. 24(18), pp. R921-R929, Sep. 2014.

[4] Baddeley, A. "Working memory: theories, models, and controversies,". *Annual review of psychology*, vol. 63, pp. 1-29. Jan. 2012.

[5] Chanthini, P. and K. Shyamala. "A Survey on Parallelization of Neural Network using MPI and Open MP," *Indian Journal of Science and Technology,* vol. 9(19), May 2016.

[6] Sarkar, Sweta, and RuchiraDatta. "A Novel Approach for Solving XOR Classical Problem," *International Journal of Emerging Technology and Advanced Engineering*, vol. 11, Nov. 2013.

[7] Parveen, Sangeeta, and Dharminder Kumar. "Minimization of error in training a neural network using gradient descent method," *International Journal of Technical Research (IJTR)*, vol. 1(1), pp. 1-8, Apr. 2012.

[8] Wilamowski, Bodgan David and Aleksander Malinowski. "Solving parity-N problems with feed-forward neural networks," *Neural Networks, Proceedings of the International Joint Conference on, vol. 4.IEEE.* pp. 2546-2551, Jul. 2003.

[9] Iyoda, Eduardo Masato, Hajime Nobuhara, and Kaoru Hirota. "A solution for the n-bit parity problem using a single translated multiplicative neuron," *Neural Processing Letters* vol. 18(3), pp. 233-238, Dec. 2003.

[10] Yam, Jim YF, and Tommy WS Chow. "A weight initialization method for improving training speed in feedforward neural network," *Neurocomputing*, vol. 30(1). pp. 219-232, Jan. 2000.

[11] Yam, Yat-Fung, and Tommy WS Chow. "Determining initial weights of feedforward neural networks based on least squares method," *Neural Processing Letters,* vol. 2(2), pp. 13-17, Mar. 1995.

[12] Phansalkar, V.V., and Sastry, P.S. "Analysis of the back-propagation algorithm with momentum," *IEEE Transactions on Neural Networks,* vol. 5(3), pp. 505-506, May 1994.

[13] Gori, Marco, and Alberto Tesi. "On the problem of local minima in back-propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 14(1), pp. 76-86, Jan. 1992.

[14] Zhang, Yudong, and Lenan Wu. "Weights optimization of neural network via improved BCO approach," *Progress In Electromagnetics Research,* vol. 83, pp. 185-198, 2008

[15] Kirar, V. P. S. "Improving the Performance of Back-Propagation Training Algorithm by Using ANN," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. *9*(1), pp. 187-192, 2015

[16] Bianchini, Monica, Paolo Frasconi, and Marco Gori. "Learning without local minima in radial basis function networks," *IEEE Transactions on Neural Networks*, vol. *6*(3), 749-756. May 1995.

[17] (2016) Theory of Neural Group Selection Wikiversity, [Online], Available: https://en.wikiversity.org/wiki/Theory_of_Neural_Group_Selection.