DESIGN AND DEVELOPMENT OF NEURAL NETWORK SIMULATOR

Mohd Helmy Abd Wahab[a], Tg. Halim Tg. Othman[b], Ayob Johari[c], Hafizul Fahri Hanafi[d], Asma Hanee Ariffin[e]

Faculty of Electrical and Electronic Engineering
Universiti Tun Hussein Onn Malaysia
P.O. Box 101, Parit Raja, 86400 Batu Pahat, Johor
helmy@uthm.edu.my, ayob@uthm.edu.my

TMNet Sdn. Bhd.
No. 138C, Ground Floor, Wisma Sentosa, Jalan Sultan Zainal Abidin, 20000 Kuala Terengganu, Terengganu
tghalim@tmnet.com.my

Faculty of Art, Computing and Creative Industry
Universiti Perguruan Sultan Idris
Tanjung Malim, Perak
hafizul@fskik.upsi.edu.my, asma@fskik.upsi.edu.my

Abstract

Neural Networks (NN) are computational models with the capacity to learn, to generalize, or to organize data based on parallel processing. Among all kinds of networks, the most widely used are multi-layer feed-forward Neural Networks that are capable of representing non-linear functional mappings between inputs and outputs and are hailed as "Universal Approximators". These networks can be trained with a powerful and computationally efficient method called error back-propagation. This paper presents the development of Neural NetworkSimulator using Backpropagation algorithm using Visual Basic 6.0. The methodology used in this development is System Development that consists of five phases. The phases include Preliminary Study, Analysis, Design, Implementation and Maintenance. Testing has been made on logic gate data AND, OR and XOR. Results show that the Neural Networks is 99% accurate.

Keywords: ***Neural network simulator, logic gate, backpropagation algorithm***.

1. Introduction

Artificial Neural Network (NN) is a computational paradigm that comprises mathematical, statistical, biological sciences and philosophy. ANN has been shown to be successful as predictive tool (Wahab, et. al., 2010) and one of the most commonly used NN tool to perform prediction, classification and forecasting is the supervised algorithm known as Backpropagation algorithm (Haykin, 1999). Some organizations have advanced sophisticated systems that constrain the training of hundreds or even thousands of NN on a weekly basis to predict stock market index movements as well as individual stock price behavior (Abhishek, et. al., 2012). This paper presents the design and development of NNsimulator which used Backpropagation algorithm in order to render an understanding on how NN works. Many courses on neural networks primarily concern on the mathematical model of Backpropagation without considering the practical development and visualization to fortitude the concept. Besides that Backpropagation algorithm is one of the sophisticated algorithms and hard to understand how it works. This scrutiny develops a simulator to assist student's comprehension the whole process by acclimatizing the values for desired parameters. This paper is organized as follows,

the next section discussed the biological to look how the artificial neuron inspire from the biological neuron followed by the section backpropagation algorithm which described in detail how the algorithms work. Then the section discussed the development of the simulator using standard SDLC method and finally discusses the training and testing using sample data.

2. Biological Background

Artificial Neural Network (ANN) (Fig. 2) is inspired from the biological nervous system as illustrated in Fig. 1. The basic anatomical unit in the nervous system is a specialized cell called 'neuron' Neuron can be characterized as an independent electric device transmitting and receiving electrical signals. There are at least 500 dissimilar types of biological neurons, but many neurons have a general structure similar to that described here (Ku Ruhana, Azuraliza and Norita, 1998). The following description of the function of a biological neuron is simplified and a typical neuron is shown in Fig. 1.

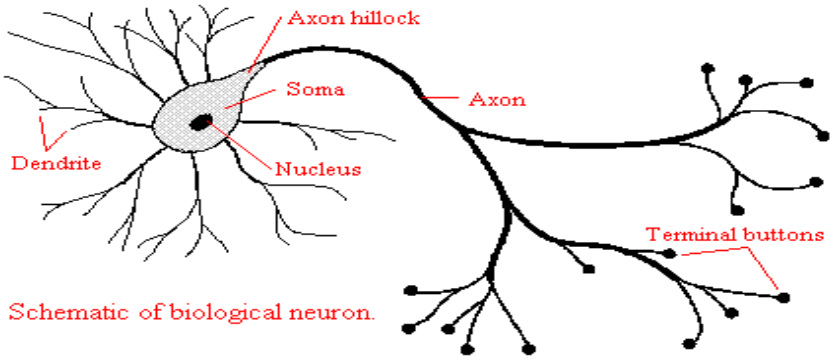

Figure 1 : Biological neuron

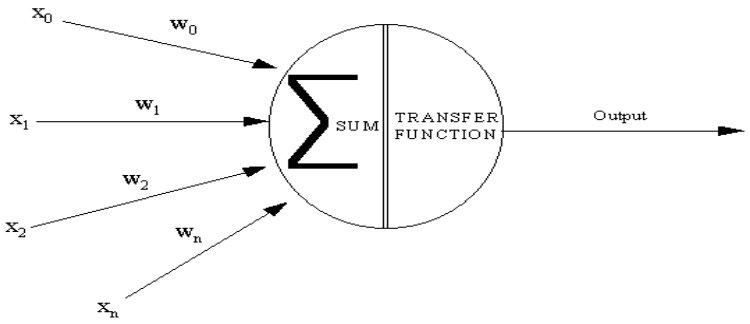

Figure 2 **:** Artificial neuron

3. Backpropagation Algorithm

This section discussed the steps how the Backpropagation algorithm works. Backpropagation learning algorithm is one of the well known algorithms in NNs. Backpropagation algorithm has been popularized by Rumelhart, Hinton, and Williams (1986) in 1980s as a euphemism for generalized delta rule. Backpropagation of errors or generalized delta rule is a decent method to minimize the total squared error of the output computed by the net. The introduction of backpropagation algorithm has overcome the drawback of previous NN algorithm in 1970s where single layer perceptron failed to solve a simple XOR problem (Wan Hussain Wan Ishak, 2004). (See Fig. 2).

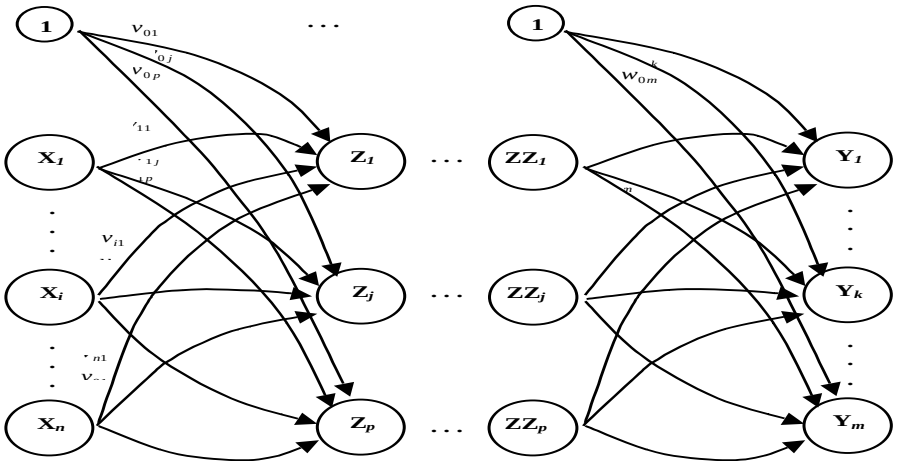

Figure 3.Backpropagation Neural Network model

The rationale of backpropagation algorithm is to exercise the net to achieve a balance between the ability to respond correctly to the input patterns that are used for training (memorization) and the ability to give reasonable (good) responses to input that is similar, but not indistinguishable to that used in training (generalization) (Fausset, 1994).Gunaseeli and Karthikeeyan (2007) described backpropagation algorithm as follows;

- Method for computing the gradient of the case-wise error functions with respect to the weights for a feed forward network.
- A training method that uses backpropagation to compute the gradient.
- A feedforward network trained by backpropagation.

In feed forward phase, each input (X_i) receives an input signal and broadcasts this signal to the hidden units $Z_1 \ldots Z_p$. Each hidden units (Z_p) computes its activation and sends its signal (z_j) to each output unit ($Y_1 \ldots Y_k$). Each output unit (Y_k) computes its activation (y_k) to form the response of the net for the given pattern.

Associated error for the pattern is determined from a comparison between output unit (y_k) and its associate target value t_k. Based on the error, the factor δ_k (k = 1, …, m) is computed. During the backpropagation phase of learning, signals are sent in the reverse direction. δ_k is used to distribute the error from output unit y_k back to all units in the previous layer (hidden units that are connected to Y_k). The error information is then used to update the weights between the output and the hidden layer. In a similar manner, the factor δ_j (j = 1, …, p) propagates the error back to the input layer and updates the weights between hidden and input layer. Taken as a whole, the adjustment to the weight w_{jk} is based on the factor δ_k and the activation z_j of the hidden unit Z_j. The adjustment to the weight v_{ij} is based on the factor δ_j and the activation x_i of the input unit.

The training procedure for the backpropagation is as follows;

Step 0:

Initialize Weight

Step 1:

While stopping condition is false, do steps 2 - 9

Step 2:

For each training pair, do steps 3 - 8

(Feed forward)

Step 3:

Broadcasts input signal (x_i where $i = 1, \ldots, n$) to all units in hidden layer.

Step 4:

Sum weighted input signals for each hidden unit (Z_j, $j = 1,\ldots,p$).

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} ,$$

compute output signal

$$z_j = f(z\_in_j)$$

Step 5:

Sum weighted input signals for each output unit (Y_k, $k = 1,\ldots,m$).

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} ,$$

compute output signal

$$y_k = f(y\_in_k)$$

(Backpropagation of error)

Step 6:

Compute error information term for each output unit (Y_k, $k = 1,..,m$)

$$\delta_k = (t_k - y_k) f'(y\_in_k)$$

calculate weight correction term ($\mu = [0,1]$)

$$\Delta w_{jk}(t+1) = \alpha \delta_k z_j + \mu \Delta w_{jk}(t)$$

calculate bias correction term

$$\Delta w_{0k} = \alpha \delta_k$$

Step 7:

Sums delta inputs for each hidden unit (Z_j, $j = 1, \ldots, p$)

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$$

calculates error information term

$$\delta_j = \delta\_in_j f'(z\_in_j)$$

calculate weight correction term

$$\Delta v_{ij}(t+1) = \alpha \delta_j x_i + \mu \Delta v_{ij}(t)$$

calculate bias correction term

$$\Delta v_{0j} = \alpha \delta_j$$

(Update weights and biases)

Step 8:

Update bias and weights ($j = 0,\ldots,p$) for each output unit (Y_k, $k = 1, \ldots, m$)

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

Update bias and weights ($i = 0,\ldots,n$) for each output unit (Z_j, $j = 1, \ldots, p$)

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$

Step 9:

Test stopping condition

NN have been shown as effective implementation in many medical applications such as basic sciences (Abidi and Goh, 1998; Prank et. al., 1998), clinical medicine (Bottaci and Drew, 1997; Pofahl, W et.al., 1998), signal processing and interpretation (Lagerholm, et. al., 2000; Dybowski, 2000) and image processing (Poli and Valli, 1995; Ahmed and Farag, 1998) have discussed several of related research in this applications domain.

Since the aim of this paper is to implement backpropagation NN, three simple problems are taken as sample data. The data are logical gates AND, OR and XOR. The truth tables of the logic gates are illustrated in Table 1, 2, and 3.

Table 1 **:** AND truth table

Input1 (X1)	Input (X2)	Target (T)
1	1	1
1	0	0
0	1	0
0	0	0

Table 2 : OR truth table

Input1 (X1)	Input (X2)	Target (T)
1	1	1
1	0	1
0	1	1
0	0	0

Table 3 : XOR truth table

Input1 (X1)	Input (X2)	Target (T)
1	1	0
1	0	1
0	1	1
0	0	0

Table 1, 2 and 3 illustrate the sample data (binary input) that are used in the simulator. The simulator runs training and testing based on the backpropagation algorithm and then demonstrates the results that are indicated as Y.

4. Development Of The Simulator

In order to develop the simulator, a system development methodology which comprises of five phases was used to construct the simulator. The phase includes:

1. Preliminary Study
2. Analysis
3. Design
4. Implementation and Maintenance
5. Testing

Preliminary Study involves collecting the information regarding the implementation of NN algorithm available on the World Wide Web. Since the implementation of NN program immerses more on the application development to solve a specific problem, the parameters used are varied based on the problem that the application needs to solve. As this study is cogitated to discover on how NN algorithm works, very simple data like the logic gate is used to be the parameter for backpropagation algorithm. Based on the preliminary study, a context diagram was defined as in Fig. 4.

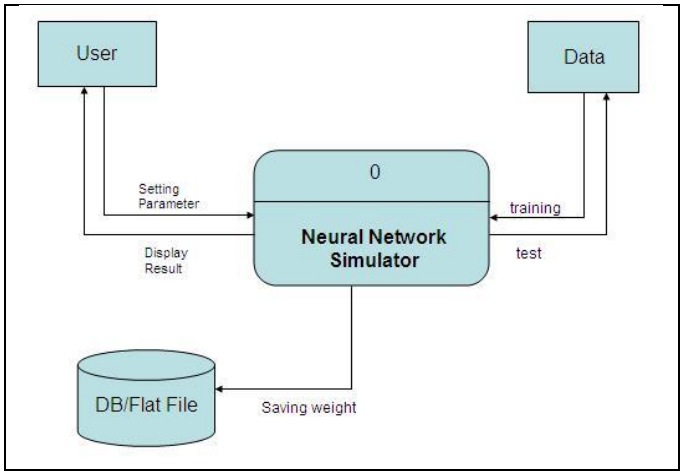

Figure 4 : Simulator context diagram

Analysis phase implicates construction of the data flow diagram to exemplify data flow in the algorithm. This assisted the researcher to make sure all formulas applied are correct. The data flow diagram is illustrated in Fig. 5.

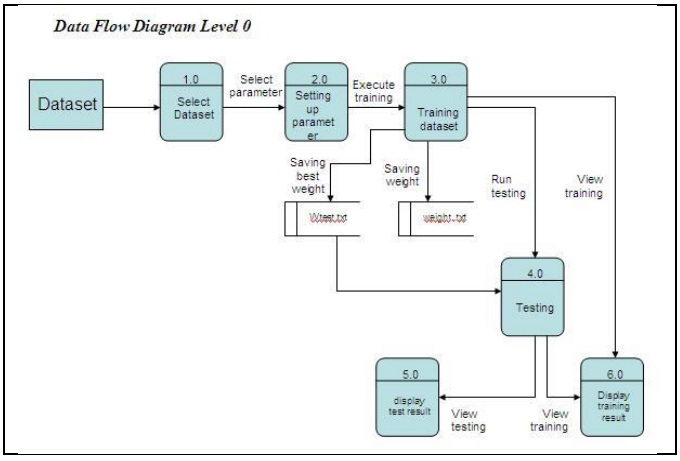

Figure 5 : Data Flow Diagram Level 0

Figure 4 illustrates the overall data flow diagram that shows the movement of data for the simulator. It consists of six inclusive processes to complete the whole processes of training and testing. However, the main foremost process that determines whether the process is successful in training and testing is process 2.0, which is the 'setting up the parameter'. The detail of process 2.0 is shown in Fig. 5.

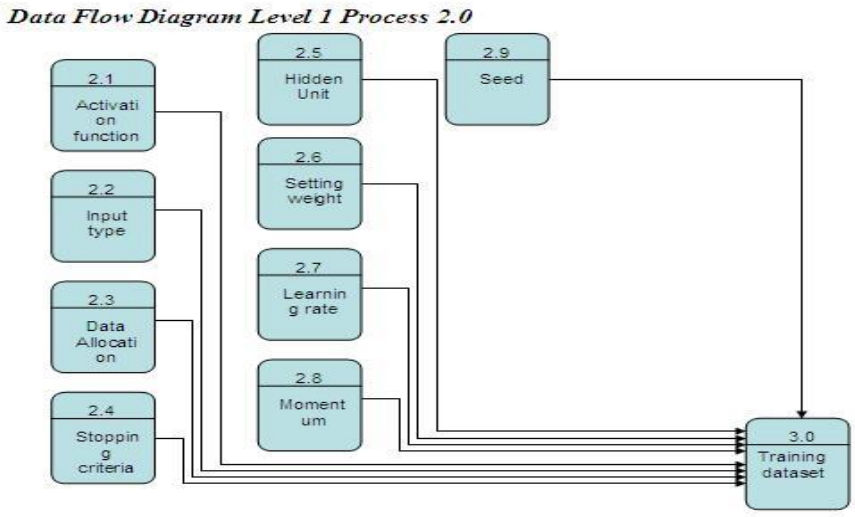

Figure 6 **:** Data flow diagram level 1

As illustrated in Figure 6, it shows the detailed process 1.0 which is related to parameter setting. The parameters involved are activation function, input type, data allocation, stopping criteria, hidden unit, weight, learning rate, momentum and seed. The parameters reflect the results of training and testing process.

Design phase implicated the design of the user interface. It was designed using Microsoft Visual Basic 6.0. Designing an interface is one of the issues on how to captivate user and at the same time to make sure a user understands how NNs works. Figure 6 illustrates the screenshot for the simulator.

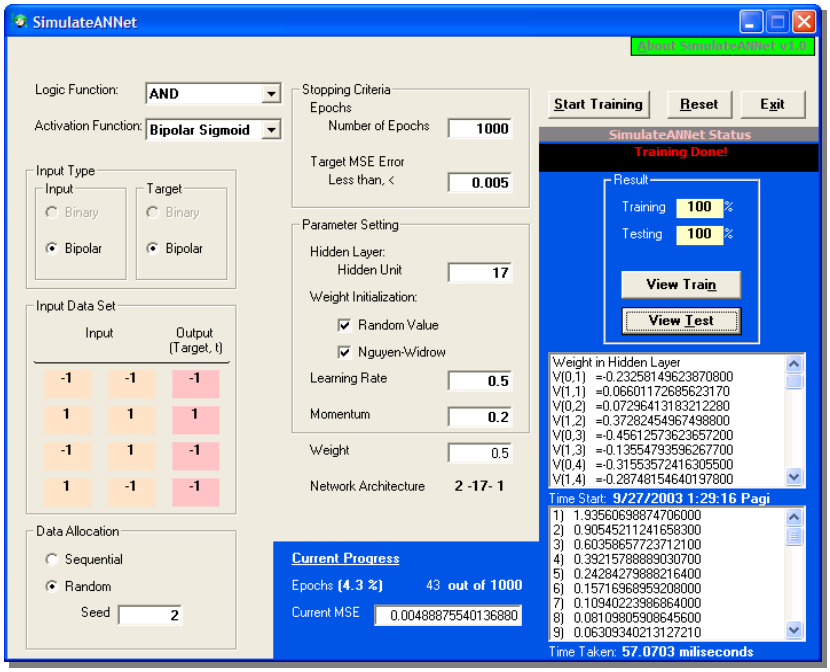

Figure 7 : Graphical user interface(GUI) for the simulator

As shown in Figure 7, all the parameters can be altered on the same interface. This facilitates the user in discernment the algorithm in terms of how it works. The bottom right area of the GUI indicates the output originated by the system. When the training process is executed, the file train.txt is created to store all training results, and the file weight.txt is created to store the weight from the simulator. After training process is done, testing can be made by clicking the button View Test and testing process will be executed.

5. Training And Testing Sample Data

This section deliberates the results of training and testing. In this experiment, the AND logic gate with Bipolar inputs and bipolar output has been screened. The results are shown are depicted in Figure 8.

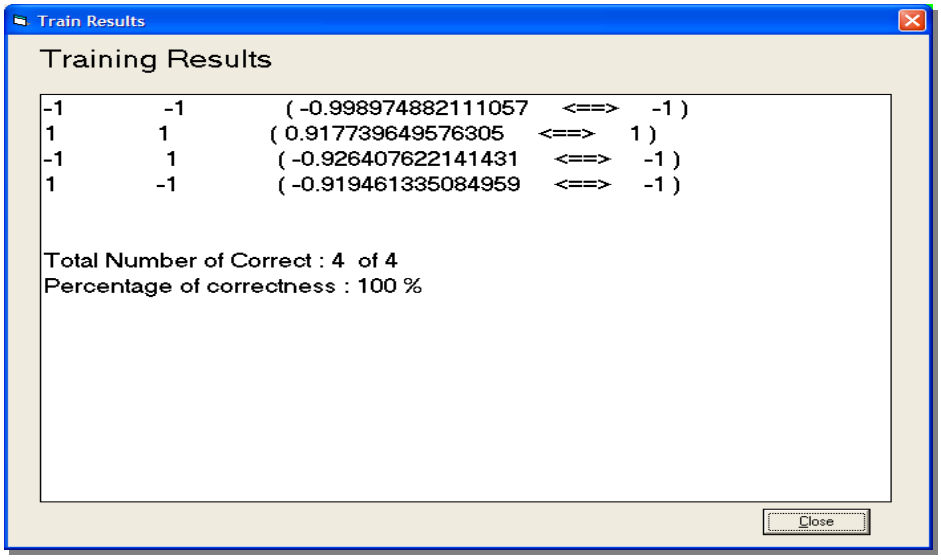

Figure 8 : Training result for AND logic gate

From Figure 7, the total number of correctness is 4 out of 4, which are equivalent to 100% of the training successful. Based on the results, the output and the target are equated to analyze the differences. In this case, the difference (error) is too small, so this study infers that the training done was are successful. In relation to Table 1, the input and output were changed into bipolar and the outcomes are shown in Table 4.

Table 4 **:** Comparison between Training results and the actual target

Input1 (X1)	Input (X2)	Target (T)	Output (Y)
1	1	1	0.91
1	-1	-1	-0.92
-1	1	-1	-0.91
-1	-1	-1	-0.99

Table 4 depicts that the training results achieve 100% training and this to indicate that the backpropagation algorithm has an ability to learn from the data.

After the training, process is accomplished the testing processes will take place to see whether the model is working successfully or not. The testing results are provided in Figure 9.

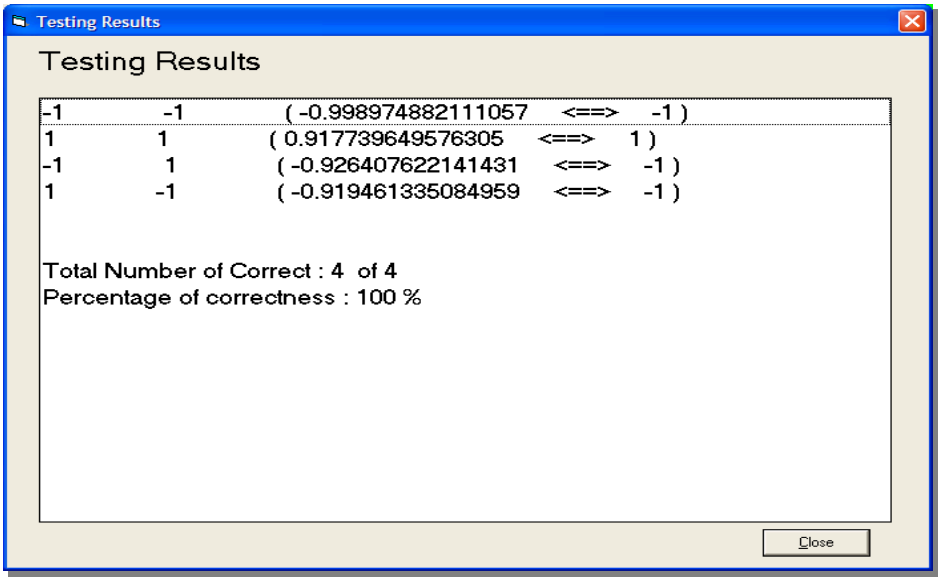

Figure 9 : Testing result for AND gate

In figure 8, testing results shows that the percentage of correctness is 100%. This discloses that the dissimilarity between output and target is small. From the results, the output and the target were then compared to analyze the differences. Since the difference (error) is too small, this study concludes that the training was running successfully. By relating to the Table 1, we changed the input and output into bipolar and the outcome can be seen in Table 5.

Table 5 : Comparison of the testing results and the actual target

Input1 (X1)	Input (X2)	Target (T)	Output (Y)
1	1	1	0.91
1	-1	-1	-0.92
-1	1	-1	-0.91
-1	-1	-1	-0.99

Table 5 depicts the comparison between the target, and the output produced by the simulator. The error is very small in which every input yielded result approaching 1. Instead of the input, several parameters such as learning rate, momentum, stopping criteria is the parameter that supplies to the algorithm to learn and produce accurate results. The use of a high number of hidden layers also contributed to the network to learn due to too many weight updates and until the errors are less than 0.005, and the network stops the process.

6. Conclusion

In this study, we have verified that by using sample data that is the representative data for the desired task, NNs able to approximate any function and behave like an associative memory. In addition NNs is also accomplished of solving complex problem based on the large number of training data in a model free estimator environment. This is the key advantage comparing to traditional approaches in estimation such as the statistical methods. NNs estimate a function without a mathematical description of how the outputs functionally depend on the inputs, and they represent a good approach that is potentially robust and fault tolerant. In this simulator, we examine the properties of the backpropagation NNs and the process of determining the appropriate network inputs and architecture, and built up AND, OR and XOR problem.

7. Future Work

The simulator can be further extended to provide the various kind of data with some automatic pre-processing mechanism to be included into the simulator. In addition, the other algorithm of neural network can be further considered to be included into the simulator to enhance the capability and extending the scope of the problem that the neural network can solve.

References

Abidi, S. S. R., and Goh, A. (1998). *Neural Network Based Forecasting of Bacteria-Antibiotic Interactions for Infectious Disease Control*. In 9th World Congress on Medical Informatics (MedInfo'98), August 18-22, 1998, Seoul, Korea.

Abhishek, K., Khairwa, A., Pratap, T., Prakash, S. (2012). *A stock market prediction model using Artificial Neural Network.* In proceeding of 3rd International Conference on Computer Communication and Networking Technologies 2012, 26 – 28 July 2012, Coimbatore, India, pp. 1 – 5.

Ahmed, M. N., and Farag, A. A. (1997). *Two-stage neural network for volume segmentation of medical images. Pattern Recognition Letters*. Vol. 18, Issues 11–13, November 1997, pp. 1143–1151.

Bottaci, L., and Drew, P. J. (1997). *Artificial Neural Networks Applied to Outcome Prediction for Colorectal Cancer Patients in Separate Institutions*. Lancet, Vol. 350, Issue 9076, pp. 469-473.

Dybowski, R. (2000). *Neural Computation in Medicine: Perspective and Prospects. In Malmgren, H., Borga, M., Niklasson, L.* (eds.) Artificial Neural Networks in Medicine and Biology, Springer-Verlag, pp. 26-36.

Fausett, L. (1994). *Fundamentals of Neural Network: Architectures, Algorithms and Applications. Prentice Hall*; Englewood Cliffs.

Gunaseeli, N.; Karthikeyan, N.; (2007).*A Constructive Approach of Modified Standard Backpropagation Algorithm with Optimum Initialization for Feedforward NNs***.** InProceeding of International Conference on Computational Intelligence and Multimedia Applications 2007, 13 – 15 Dec 2007, Tamilnadu, India.

Haykin, S. (1999). *Neural Network: A Comprehensive Foundation*. 2nd Ed. New Jersey: Prentice Hall.

Ku Ruhana, K. M., Azuraliza, A. B., and Norita, N.(1998). *Neural Network Modeling to Predict House Prices Performance*. Research Report. 30 September 1998.

Lagerholm, M., Peterson, C., Braccini, G., Edenbrandt, L., and Sörnmo, L. (2000). *Clustering ECG Complexes Using Hermite Functions and Self-Organizing Maps*. IEEE Transactions on Biomedical Engineering 47, 838-848.

Prank, K., Jurgens, C., Muhlen, A., and Brabant, G.(1998). *Predictive Neural Networks for Learning theTime Course of Blood Glucose levels from the ComplexInteraction of Counterregulatory Hormones*. Neural Computation, Vol. 10, Issue 4, pp. 941-954.

Pofahl, W. E., Walczak, S. M., Rhone, E., and Izenberg, S. D. (1998). *Use of an artificial Neural Network to Predict Length of Stay in Acute Pancreatitis*. American Surgery. 64(9):868-72.

Poli, R, and Valli, G. (1995). *Optimum Segmentation of Medical Images with Hopfield Neural Networks.* Technical Report CSRP-95-12. The University of Birmingham.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). *Learning representations by back-propagating errors*. *Nature***323**, 533 – 536.

Wahab, M. H. A, Sudin, N, Sulaiman, M. S., Sidek, R. M., and Ahmad, N. (2010). *A model to predict diabetic patient using multi-layer perceptron.* In Proceeding of Seminar on Science, Technology and Social Sciences 2010, 1-2 June 2010, Kuantan, Pahang.

Wan Hussain Wan Ishak. (2004). *Notes on Neural Network Learning and Training*. URL: http://www.generation5.org/content/2004/NNTrLr.asp [Date Accessed: 22 Sept. 2014]