*IOSR Journal of Mathematics (IOSR-JM)*
*e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 11, Issue 6 Ver. II (Nov. - Dec. 2015), PP 13-16*
*www.iosrjournals.org*

# On The Application of Hyperbolic Activation Function in Computing the Acceleration of Reverse Analysis Method

Augustine Pwasong [a] & Caleb Nimyel[b]
[a]*Department of Mathematics University of Jos, 2084 Jos, Nigeria*
[b]*Department of Computer Engineering Plateau State Polytechnic, Barkin-Ladi, Plateau State, Nigeria*

***Abstract:*** *Hyperbolic activation function is examined for its ability to accelerate the performance of doing data mining by using a technique named as Reverse Analysis method. In this paper, we describe how Hopfield network perform better with hyperbolic activation function and able to induce logical rules from large database by using reverse analysis method: given the values of the connections of a network, we hope to determine what logical rules are entrenched in the database. We limit our analysis to Horn clauses. The analysis for this study was simulated using Microsoft Visual C++ software, 2010 Express.*
***Keywords:*** *Reverse analysis, data mining, hyperbolic activation function, neural network*

## I. Introduction

Neural network is an information processing paradigm that is inspired by the way biological nervous system such as the brain, process information. Neural networks or also known as connectionist system have a large number of highly interconnected processing elements (nodes) that usually operates in parallel. It is a parallel processing network which is generated with simulating the image intuitive thinking of human, on the basis of the research of biological neural network, according to the features of biological neurons and neural network and by simplifying, summarizing and refining. It uses the idea of non-linear mapping, the method of parallel processing and the structure of the neural network itself to express the associated knowledge of input and output.

The Hopfield neural network is a simple recurrent network which can work as an efficient associative memory, and it can store certain memories in a manner rather similar to the brain. Wan Abdullah [1,2] proposed a method of doing logic program on a Hopfield network.

In Reverse Analysis [3], it is a method that when examining the connection strengths obtained during the learning process, logical rules that have been acquired may be deduced. This step will repeat until it gets the value that it wanted. From the Reverse Analysis given the values of the connections of a network, we hope to know what logical rules are entrenched in it.

In this paper, we want to integrate hyperbolic activation function in Reverse Analysis method to enhance the capability of doing data mining using Reverse Analysis method. We carried out computer simulations to verify the robustness and the effectiveness of the proposed method. This paper is organized as follows. In section 2, an outline of Hopfield network is given and in section 3, Reverse Analysis method is described. Meanwhile, section 4 contains discussions regarding the hyperbolic activation function. Finally, sections 5 and 6 occupy the simulation results and concluding remarks regarding this work.

## II. Hopfield Network

In order to keep this paper self-contained we briefly review the Little-Hopfield model [4]. The Hopfield model is a standard model for associative memory. The Little dynamics is asynchronous, with each neuron updating their state deterministically. The system consists of *N* formal neurons, each of which is described by an Ising variable $S_i(t), (i = 1, 2, ....N)$. Neurons then are bipolar, $S_i \in \{-1,1\}$, obeying the dynamics $S_i \to \mathrm{sgn}(h_i)$, where the field, $h_i = \sum_j J_{ij}^{(2)} V_j + J_i^{(1)}$ , *i* and *j* running over all neurons *N*, $J_{ij}^{(2)}$ is the synaptic strength from neuron *j* to neuron *i*, and $-J_i$ is the threshold of neuron *i*.

Restricting the connections to be symmetric and zero-diagonal, $J_{ij}^{(2)} = J_{ji}^{(2)}$, $J_{ii}^{(2)} = 0$, allows one to write a Lyapunov or energy function,

$$E = -\frac{1}{2}\sum_i \sum_j J_{ij}^{(2)} S_i S_j - \sum_i J_i^{(1)} S_i \qquad (1)$$

which monotone decreases with the dynamics.

The two-connection model can be generalized to include higher order connections. This modifies the "field" to be

$$h_i = .... + \sum_j \sum_k J_{ijk}^{(3)} S_j S_k + \sum_j J_{ij}^{(2)} S_j + J_i^{(1)} \qquad (2)$$

where "….." denotes still higher orders, and an energy function can be written as follows:

$$E = ..... - \frac{1}{3} \sum_i \sum_j \sum_k J_{ijk}^{(3)} S_i S_j S_k - \frac{1}{2} \sum_i \sum_j J_{ij}^{(2)} S_i S_j - \sum_i J_i^{(1)} S_i \qquad (3)$$

provided that $J_{ijk}^{(3)} = J_{[ijk]}^{(3)}$ for *i, j, k* distinct, with […] denoting permutations in cyclic order, and $J_{ijk}^{(3)} = 0$ for any *i, j, k* equal, and that similar symmetry requirements are satisfied for higher order connections. The updating rule maintains

$$S_i(t+1) = \text{sgn}[h_i(t)] \qquad (4)$$

In the simple propositional case, logic clauses take the form $A_1, A_2, ....., A_n \leftarrow B_1, B_2, ...., B_m.$ which says that ( $A_1$ or $A_2$ or …. or $A_n$ ) if ( $B_1$ and $B_2$ and … and $B_n$ ); they are program clauses if $n = 1$ and $m \geq 0$ : we can have rules e.g. $A \leftarrow B, C.$ saying $A \vee \neg(B \wedge C) \equiv A \vee \bar{B} \vee \bar{C}$, and assertions e.g. $D \leftarrow .$ saying that $D$ is true.

A logic program consists of a set of program clauses and is activated by an initial goal statement. In Conjunctive Normal Form (CNF), the clauses contain one positive literal. Basically, logic programming in Hopfield model can be treated as a problem in combinatorial optimization. Therefore it can be carried out in a neural network to obtain the desired solution. Our objective is to find a set of interpretation (i.e., truth values for the atoms in the clauses which satisfy the clauses (which yields all the clauses true). In other words, we want to find 'models'.

## III. Reverse Analysis method

We used Reverse Analysis method to induce the logical rules entrenched in a database through uncovering rules using Wan Abdullah's learning method [1,2] to calculate the connection strengths. We restricted this technique up to third order Horn clauses only. For higher order networks different techniques and upgrading methods are been used. The following are the steps involved in this method.

i) Enumerate the number of neurons from patterns in the database, to construct a Hopfield network with up to third order connections.
ii) Extract the events from the data sets and represent in binary/bipolar pattern, where 0 indicates a false state and 1 indicates a true state (for bipolar neurons, -1 represents the false state and 1 represents the true state).
iii) Calculate the connection strengths for the events using Wan Abdullah's learning.
iv) Find nonzero resultant values for third-order connections, each of which being a seed for a three-atom rule (rules represented in the Conjunctive Normal Form or CNF).
v) Decide on the form of a rule by looking at the values of relevant second-order connections, etc.
vi) Calculate connection strengths for the extracted rules and deduct these values from (iii).
vii) Repeat similar steps to IV)-VI) for second-order (for two-atom rules) and first-order connections (for one-atom rules).

## IV. Hyperbolic Activation Function

Hyperbolic tangent activation function is one of well know activation function [5]. However, logistic function which was frequently in use in neural network, introduced by McCulloch-Pitts where it is already established in original method of doing logic programming in Hopfield network proposed by Wan Abdullah. Since McCulloch-Pitts function is unbounded, smoother function such as hyperbolic tangent which have output range between -1 and +1 are preferred.

Most neurons in neural networks using a scalar-to-scalar function which was called activation function to transform their net input while the activation value is fed via synaptic connections to one or more other neurons. The activation function is sometimes called a transfer function. The transfer function with a bounded range are frequently called squashing functions. Hyperbolic tangent activation function is one of an example of squashing function [6].

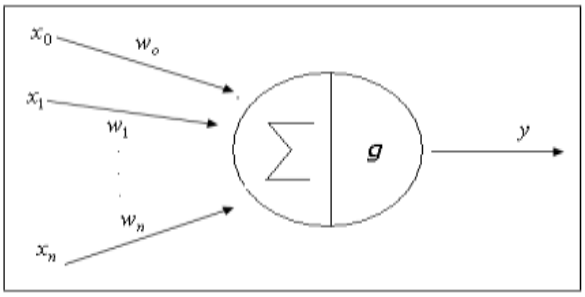


Figure 1: An artificial neuron with activation function

Figure 1 shows the artificial neuron $x$ which is a neuron with $n$ input dendrites ($x_0 \dots x_n$) and where ($w_0 \dots w_n$) are weights of inputs and one output axon $y(x)$. $g$ is an activation function that weights how dominant the output should be from the neuron based on the sum of the input. Equation 5 shows the equation of activation function.

$$y(x) = g\left( \sum_{i=0}^{n} w_i x_i \right) \tag{5}$$

where $g(x)$ used the hyperbolic tangent function as below,

$$g(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{6}$$

At first, we substitute the total of weights and inputs into Hyperbolic Tangent function to achieve the value. If it surpasses the threshold ($\theta$ = 0), the actual output will be 1 else the value of the actual output is -1. After obtained an output pattern during the network computation, if error exists, or a difference between actual and desired output patterns, the weights will be adjusted to reduce the error.

## V. Experimental Results and Discussion

From the previous section, a new way to integrate hyperbolic tangent activation function into Reverse Analysis method. Thus, in this section we carry out computer simulations using Netlogo [7] platform to test the effectiveness, computational complexity and robustness of this method.

We randomly generate the Horn clauses (until third order) by using reverse analysis method. The number and the order of the clauses are chosen by the user using try and error technique [8]. The numbers of neurons involved are increased in each training. The maximum number of neurons is 100. After this value, the network gets larger and the complexity increased tremendously. So, the network gets stuck[9] .

The following figure 2 shows the computer processing time (CPU) consumed for the integrated method. Figure 3 shows the computer processing time (CPU) consumed for the Reverse Analysis method without hyperbolic activation function.

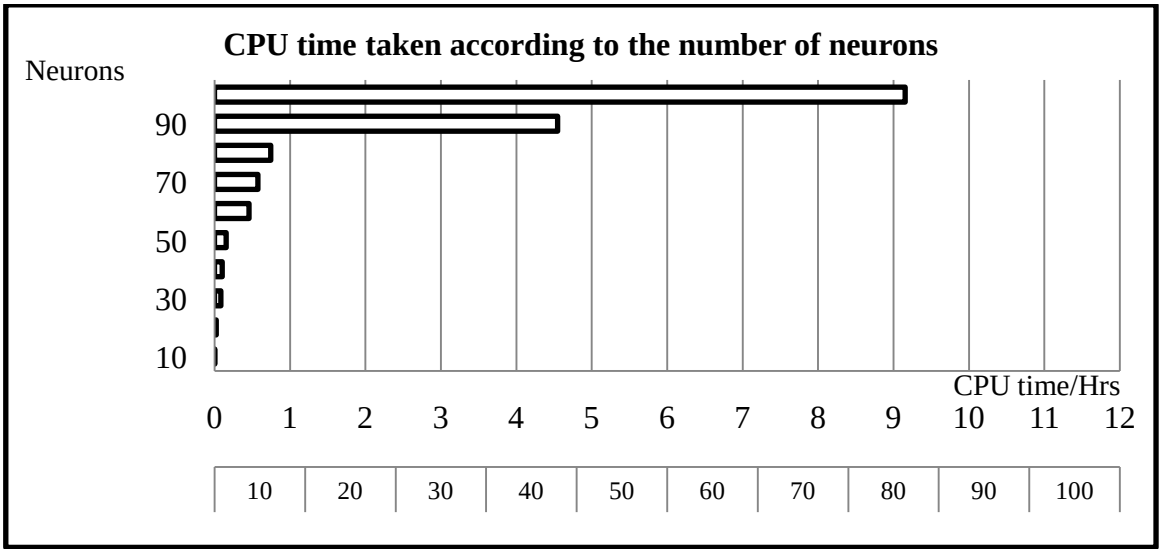


Figure 2: CPU time taken according to the number of neurons (integrated method)

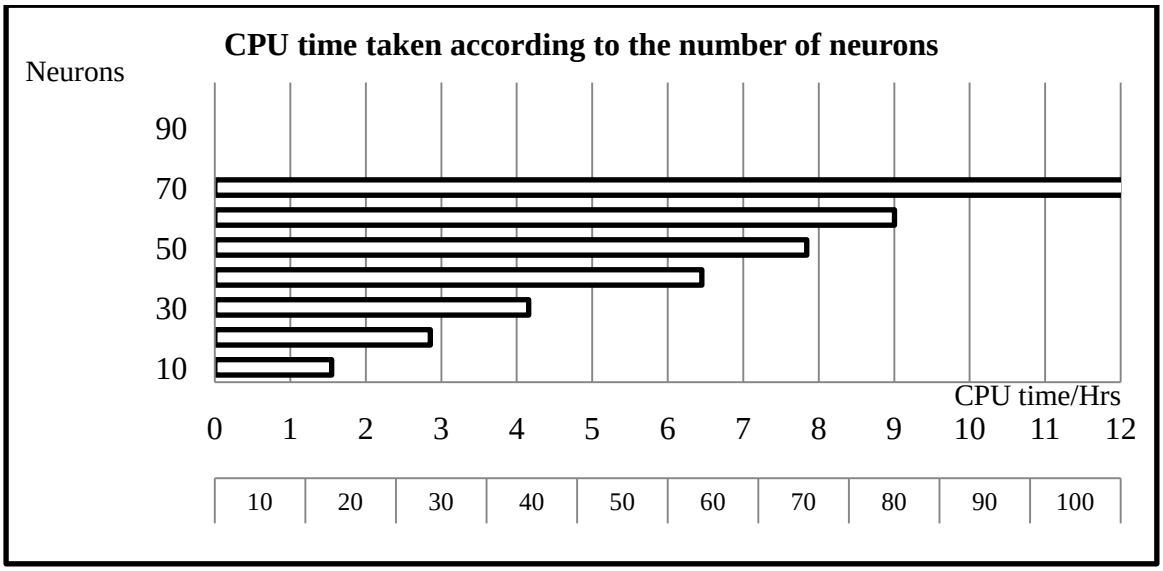


Figure 3: CPU time taken according to the number of neurons (normal method)

We can observed from the Figure 2 that the CPU time gets larger when the network gets more complex (number of neurons increased). The energy relaxation looping becomes complex when the number of neurons involved increased. When we integrate Reverse Analysis method with hyperbolic activation function, this problem can be minimize. So, we can conclude that the network performance with the integrated method is better and more efficient than the normal Reverse Analysis method.

## VI. Conclusion

From the theory refinement we find that the ability of hyperbolic activation function integrated with Reverse Analysis is better than Reverse Analysis method alone. It provides a better result in term of complexity and CPU for Horn clauses. Computer simulation verified and testified the proposed integrated method.

## Acknowledgement

This research is supported by Fundamental Research Grant Scheme (203/ PMATHS/6711368) by Ministry of Higher Education, Malaysia and also Universiti Sains Malaysia.

## References

[1] Wan Abdullah, W.A.T. (1992) Logic Programming on a Neural Network. Int .J. Intelligent Sys, **7**, pp 513-519.
[2] Wan Abdullah, W.A.T. (1991) Neural Network logic. Proc of the workshop held in Elba International Physics Center, Italy, pp 135-142.
[3] Sathasivam, S (2006) Logic Mining In Neural Networks. PHD Thesis. University of Malaya.
[4] J.J. Hopfield, Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings. Natl. Acad. Sci. USA. , 79(1982), 2554-2558.
[5] Sathasivam, S. and Wan Abdullah, W.A.T. (2010) The Satisfiabilty Aspect of Logic on Little Hopfield Network. ISSN 1450-223X (7), pp.90-105
[6] Joya, G, Atencia, M.A. and Sandoval, F. (2002) Hopfield neural networks for optimization: study of the different dynamics, Neurocomputing **43**, Amsterdam: Elsevier Science B.V pp.219-237
[7] Isaac, A. G. (2012) NetLogo Programming [Online]. [Accessed 15 May 2013]. Available from: https://subversion.american.edu/aisaac/notes/netlogo-intro.xhtml.
[8] Sathasivam, S (2011) Application of Neural Networks in predictive data mining. Proceeding of 2nd International Conference on Business and Economic Research (2nd ICBER 2011)
[9] P.H. Wen, A Review of Hopfield Neural Networks for Solving Mathematical Programming Problems, European Journal of Operational research, Vol.198, Num.3, (2008), 675-688.