IJCTA *Vol.7 No.2 July-Dec 2014, pp. 63-72*

ISSN:- 0974-5572

REPRESENTATION OF FUNCTIONS BY SIGMOID FUNCTION CONTINUOUS MAPPINGS USING MULTILAYER FEEDFORWARD NEURAL NETWORKS

R. Murugadoss[1] and M. Ramakrishnan[2]
[1]*Sathyabama University, Research Scholar*
Department of Computer Science and Engineering, Chennai,
St Ann's College of Engineeing and Technology, Chirala-523157. Andhrapradesh.
[2]*Professor and Head, Department of Information Technology*
Velammal Engineering College, Chennai.
Email:- murugadossphd@gmail.com, mdossresearch@gmail.com, ramkrishod@gmail.com

Abstract- *In this work, The ability of artificial neural network-based parallel processing architecture. This makes them in pattern recognition, the system is useful to identify and control problems. Multilayer Perceptron is an artificial neural network with one or more hidden layers. Activation function determines the Multilayer Perceptron performance. In Multilayer Perceptron, the most common activation function is S-shaped and bipolar sigmoid activation function. Activation function for converting the active unit (neurons) of the output signal level. There are many artificial neural network (ANN) using ordinary activation. Our objectives were to analyze different activation functions and provide it a benchmark. The purpose is to determine the optimal activation function problems.*

Index Terms *- FPGA, Sigmoid Activation Function, Artificial Neural Network (ANN),*

1. INTRODUCTION

Artificial Neural Network (ANN) is calculated as the area has received considerable attention in the last decade of. So far, the Multilayer Perceptron (MLP) is the most common neural network model. Back-propagation algorithm is the most commonly used MLP learning algorithm. The ability of the MLP can be from three different angles to observe: first, because you have the ability to implement logic functions, and the second and it is divided into classification model space capabilities third, it is possible to approximate the nonlinear transform function the problem.

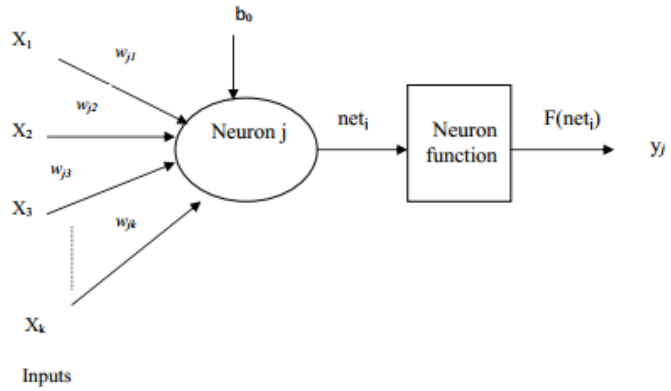

Figure 1. The Basic Neuron Model

Motivation for doing the work is the use of RNA to improve the current technology. So far, there have been some studies emphasize the freedom to adjust the parameters of the activation

function. Activation function network (neuron adaptive), seem to be using a fixed activation function, providing a better fit from the traditional architecture of. In the literature, the actual variable 1 (gain) and b (slope) in the generalized S-shaped activation function is set in the learning process. It is reported that, in static and dynamic simulation system compares with the classic RNA, which concluded that the S-shaped lead to better adapt to data modeling. In, the activation is sufficient to build a piecewise approximation has to adapt to any shape neuronal function, and allows the overall size of the neural network is reduced cubic spline function is found, the negotiations in order to connect with the activation function of the complex complexity. Other authors also studied the use of neural network adapter activation function neurons attributes. The ability of RNA approximate any continuous function, this author has established a set of neural network model can approximate a continuous function of position (such as analog intermittent financial data device).

2. LITERATURE SURVEY

A group of neural network is a group of generalized neural networks, where each element is a whole neural network, and product and the other two elements of the definition. However, although the authors have used artificial neural network model to simulate data sets, three issues have emerged. One is that although the group artificial neural network model can simulate nonlinear continuous function of non-contiguous blocks of non-smooth point and a point, but hard to really find these discontinuities and uneven. Another reason is that the actual learning algorithms have not yet been developed set of artificial neural network model. The third problem is that if the piecewise continuous function is approximated by an infinite number of parts of a continuous function, and then the neural network must contain an unlimited neural network group. This makes the simulation of very complex, because each successive portion to be independently and separately estimates. Therefore, it is necessary to use a neural network (rather than a group) approximation to consider piecewise continuous function. Motivation of this work is to find a new neural network model can not only simulate nonlinear piecewise continuous function is not continuous and smooth dotted, but also can automatically discover these discontinuities and uneven through simple learning algorithm. Enter the number of nodes must enter the number of attributes and their training and test data. The number of hidden nodes can be any value you want. Typically, this is slightly larger than the number of input nodes. You can try to run the effect of this decision on your network. Number of output nodes must attribute classification match their training and test data. Training iterations to determine the stopping criteria for the training phase. Select a larger value, you can make the training will take some time. Because this is the only stopping criterion, the training cycle will eventually end, and not fall into an infinite loop. Artificial Neural Network (ANN) is suitable for this type of problem. They conducted a survey and applied to the actual [a] system [3]. Genetic algorithms have been presented in various references [3] [5]. In the short period of its development, shares of the company demonstrated its superior capabilities and has been successfully applied in many fields. MLP network by using a source, one or more of the hidden layer and the peeling layer neurons in the input layer. Based on the input variables and the number of the number of classes required for the input layer and the output neuron, respectively. Effect of the number of hidden layers and the number of neurons in each hidden layer of the network to generalize. In this article, use the following first-order time derivative [21] ideas, dynamic development and analysis of general recursive neural network model implicit variable matrix inverse problem solving time. Elegant definition of the usual cost function value scalar neural dynamics, making the calculation error can be reduced to zero error monitoring function asymptotic value matrix global level, rather than the introduction. As noted, there is always an error and linearity. Even under circumstances linear activation function, nonlinear phenomena occur in hardware. This value can be as small as 100 (if you're going to have to go through several training sessions). 1000 is a single training or training sessions had good initial choice. 10000 seems to be enough for a single training course, but may be too large, you can train a categorical data (see ANN literature understanding and overtraining related issues). The right to apply to re-training of the network each iteration learning rate limiting

misclassification department. This value need not be very large; 0.02 seems to work very well. Value is too large may correct network oscillations caused by too fast, without a good converge to a stable network. Have this relationship between the numbers of iterations of the value of training. Small selection of this value, the need to train more iterations. Boost factor is not used, but must be incorporated into future versions of this application. The slope of each node activation sigmoid activation function to control the rate of the S-shaped. S-shaped function is a: c is the value of this equation is the activation rate of S-shaped. The higher the value the steeper the slope, the end step function similar to a function of the value of C is large (> 50). This parameter is a good initial value, you can try using a larger value, look tolerant network, accuracy and noise impact they generate. Name of training and test files are simple text strings. Determine the data file that training session. Does not include the file extension. Assuming extended DAT. Use any other extension data file will not be found application. These names cannot contain spaces. The application will automatically create the output file classification using these names. Each output file will be individually numbered rankings for the training provided so that multiple training sessions can compare the results of the session. Use the same file name new training courses covering old from any previous output file. These two files

3. METHODOLOGY

Each neuron model includes a processing element input and an output of synaptic connections. Input neurons, the signal flow hi, is considered to be unidirectional. Neuron output signal [7] by the relation o = f(Σ), which is given in Figure 2 shows

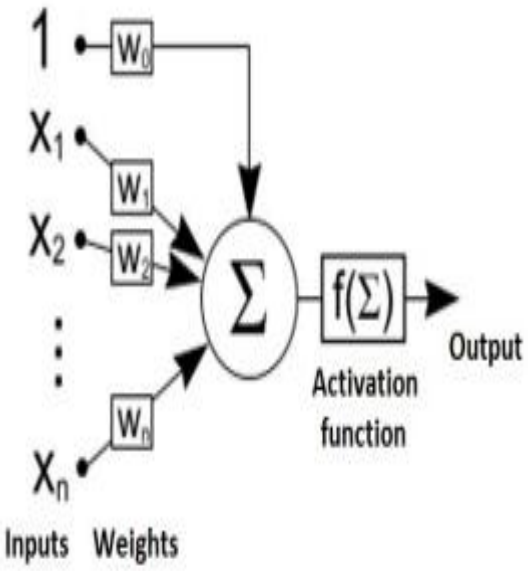

Figure 2. Preprocessing element with synaptic input connections

The functions and parameters, which describe
• x is input to activate the feature,
• y is the output,
• s is the slope and
• d is derived.

S-shaped function can also generate a positive number between 0 and 1 is a one of the most common activation function between 0 and 1 sigmoid activation function is for training data is most used activation function. S-shaped activation function is a piecewise linear approximation of the usual gradual S-shaped function with output between 0 and 1. Faster than the sigmoid colon, but a little inaccurate.

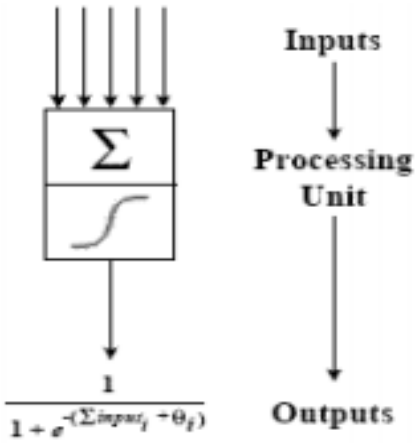

Figure 3. Artificial Neuron Network

The activation function is symmetric S-shaped Departure from the usual hyperbolic tangent function S between One of the at least one. It is one of the most commonly used one Activation function. Symmetric function is an S-shaped activation. The usual hyperbolic tangent of piecewise linear approximation With a negative output S-type function One. Than symmetric S-shaped, but faster Bit is not accurate. Activation is one of the key parameters in the neural network. Evaluate different options to activate the function return until there is no much difference between them. When the network has been successfully trained, each activation function is approximately the same effect thereto. It clearly showed that the extent of activation is very important. Select a specific network or node activation function is an important task. But as the results show, if a network successfully trained to activate a specific function, then there is a high probability that others will also lead to the activation function properly trained neural network. We emphasize that, although the choice of a neural network or a node activation function is an important task, other factors such as the size of the training algorithm, and learning the network parameters of the network is important to proper formation show simulation results when different the activation function is configured with minimal training is only one difference.

4. PROPOSED METHOD

Instead of the standard S-type function, ABFNN S-type function uses a variable defined as:

$$O_f = \frac{a + tanh(x)}{1 + a} \tag{5}$$

This rule is explained in these terms: increasing the input product for a given weight in proportion to connect two neurons that supply output.

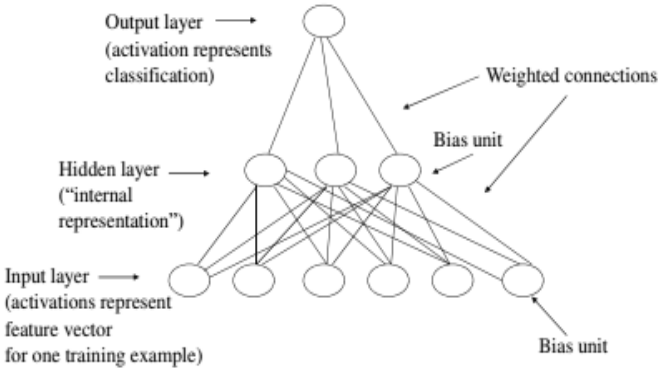

Figure 4. A two-layer neural network

Use threshold to activate a secondary function has a hidden layer of BP network initialization method of the new weights is developed. Even without the weights may be obtained directly further study. This hypothesis is based on the general location of the distribution pattern. This method can be applied to many pattern recognition tasks. Finally, simulation results, indicating that this technique greatly reduces the training time and the number of hidden neurons is typically rendered initialization. Pattern recognition applications using neural networks is increasingly attractive. The main advantage of using neural networks in parallel, adaptability and tolerance lot of noise, which is the most popular back-propagation neural network (BP) and Multilayer Perceptron (MLP) neural network. BP is the most common activation is -Limited \ hard threshold function and the S-shaped function. Use a hard limit activation function in each neuron in hidden neurons to solve the problem of the general location of the maximum number of two-class classification of the required network is a single hidden layer $\left\lceil \frac{K}{n} \right\rceil$ where K is the number of modes, n is the dimension of the input. If there is no restriction Typically, the maximum number of hidden neurons for K - Threshold recent new quadratic function 1 activation, proposed the use of the ceiling in each neuron show hidden neurons is necessary to solve a given number two category You can use the function compared to hard classification problems existing thresholds multilayer Perceptron, can be reduced by half, and the results are shown in Table 1 as a quadratic function is slightly more complex function of the hard threshold limit, learning is much more difficult for BP network quadratic function. Both the convergence of the learning period, and are not good enough to give a valid result can be observed in the typical analog. A new method for BP network weights threshold activation function used in this difficult learning initialization heavy mitigation proposed secondary with a hidden layer. Let us now describe the initialization scheme more formally. Let the input and hidden layers each having a biasing unit with a constant activation of the input values and the weighting matrix $W^{(1)}$ and $W^{(2)}$, respectively between the input and hidden layer and between the hidden layer and the output .

$$W^{(2)} = \begin{pmatrix} W_{1,0}^{(1)}, W_{1,1}^{(1)}, \cdots, W_{1,n}^{(1)} \\ W_{2,0}^{(1)}, W_{2,1}^{(1)}, \cdots, W_{2,n}^{(1)} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ W_{N_1,0}^{(1)}, W_{N_1,1}^{(1)}, \cdots, W_{N_1,n}^{(1)} \end{pmatrix} = \begin{pmatrix} W_{1,0}^{(1)}, W_1^{(1)} \\ W_{2,0}^{(1)}, W_2^{(1)} \\ \cdots\cdots \\ W_{N_1,0}^{(1)}, W_{N_1}^{(1)} \end{pmatrix} \quad (1)$$

Where $W_{j,i}^{(1)}$ signifies the linking since the *i*-th contribution element to the *j*-th hidden neuron, for $j = 1, 2, \ldots, N_1$, $i = 0, 1, \ldots, n$; $W_{1,j}^{(2)}$ signifies the linking from the *j*-th buried unit to the output neuron, where $j = 0, 1, 2, \ldots, N_1$;

$$W^{(2)} = (W_{1,0}^{(2)}, W_{1,1}^{(2)} \cdots, W_{1,N_1}^{(2)}) = (W_{1,0}^{(2)}, W_1^{(2)}) \quad (2)$$

Let $\theta^{(1)}$, $\theta^{(2)}$ be the vector of θs values of the hidden and output layer, respectively. $\theta_j^{(1)}$ and $\theta_1^{(2)}$ signify the θ value of the *j*-th hidden neuron $j = 1, 2, \ldots, N_1$ and of the output neuron, correspondingly.

$$\theta^{(1)} = (\theta_1^{(1)}, \theta_2^{(1)} \cdots, \theta_{N_1}^{(1)}); \quad \theta^{(2)} = (\theta_1^{(2)}) \quad (3)$$

If $K_0 \geq K_1$, we custom the set of outlines in set H_1 consistent to class 1 to attain the weights values for the $\left\lceil \frac{K_1}{n} \right\rceil$ hidden neuron, so the amount of the hidden neurons let

$$W_{j,0}^{(1)} = 1 \tag{4}$$

$$W_{1,j}^{(2)} = 1 \tag{5}$$

$$W_{1,0}^{(2)} = -1 \tag{6}$$

$$\theta_j^{(1)} = \beta \tag{7}$$

$$\theta_1^{(2)} = \beta \tag{8}$$

where $0 < \beta < 1$ is a minor positive sum.

In mandate for each j-th hidden neuron signifies a dissimilar set of n input patterns, we resolve the following equations to get $W_{j,i}^{(1)}$, for each $j = 1, 2, \ldots, \left\lfloor \frac{K_1}{n} \right\rfloor$, $i = 1, 2, \ldots, n$.

$$\begin{cases} \sum_{i=1}^{n} W_{j,i}^{(1)} q_i^{(j-1)n+1} + W_{j,0}^{(1)} = 0 \\ \sum_{i=1}^{n} W_{j,i}^{(1)} q_i^{(j-1)n+2} + W_{j,0}^{(1)} = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum_{i=1}^{n} W_{j,i}^{(1)} q_i^{(j-1)n+n} + W_{j,0}^{(1)} = 0 \end{cases} \tag{9}$$

If $\frac{K_1}{n}$ is a integer (i.e. $\left\lfloor \frac{K_1}{n} \right\rfloor = \left\lceil \frac{K_1}{n} \right\rceil = N_1$), here we done. If $\frac{K_1}{n}$ is not a integer (i.e. $\left\lfloor \frac{K_1}{n} \right\rfloor = \left\lceil \frac{K_1}{n} \right\rceil - 1 = N_1 - 1$), the rest $K_1 - n \times \left\lfloor \frac{K_1}{n} \right\rfloor$ outlines are then signified by the N_1-th hidden neuron by the subsequent formula. For $j = N_1$, $i = 1, 2, \ldots, K_1 - n \times \left\lfloor \frac{K_1}{n} \right\rfloor$.

$$\begin{cases} \sum_{i=1}^{n} W_{j,i}^{(1)} q_i^{n*\left\lfloor \frac{K_1}{n} \right\rfloor + 1} + W_{j,0}^{(1)} = 0 \\ \sum_{i=1}^{n} W_{j,i}^{(1)} q_i^{n*\left\lfloor \frac{K_1}{n} \right\rfloor + 2} + W_{j,0}^{(1)} = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum_{i=1}^{n} W_{j,i}^{(1)} q_i^{K_1} + W_{j,0}^{(1)} = 0 \end{cases} \tag{10}$$

The equations can solve $W_{N_1,i}^{(1)}$ for $i = 1, 2, \ldots, K_1 - n \times \left\lfloor \frac{K_1}{n} \right\rfloor$. Let the other $n - K_1 + n \times \left\lfloor \frac{K_1}{n} \right\rfloor$ weights attaches to the N_1-th hidden neuron be zero.

If $K_0 < K_1$, we practice the set of outlines in set H_0 consistent to class 0 to attain the weights values for the $\left\lceil \frac{K_1}{n} \right\rceil$ hidden neuron, so the number of the hidden neurons $N_1 = \left\lceil \frac{K_0}{n} \right\rceil < \left\lceil \frac{K}{2n} \right\rceil$. The whole thing is the same excluding that let

$$W_{1,0}^{(2)} = 0 \tag{11}$$

and **q** is replaced by **p**. We currently resolve the subsequent equations to get $W_{j,i}^{(1)}$, for each $j = 1, 2, \ldots, \left\lfloor \frac{K_n}{n} \right\rfloor$, $i = 1, 2, \ldots, n$.

$$\begin{cases} \sum_{i=1}^{n} W_{j,i}^{(1)} p_i^{(j-1)n+1} + W_{j,0}^{(1)} = 0 \\ \sum_{i=1}^{n} W_{j,i}^{(1)} p_i^{(j-1)n+2} + W_{j,0}^{(1)} = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum_{i=1}^{n} W_{j,i}^{(1)} p_i^{(j-1)n+n} + W_{j,0}^{(1)} = 0 \end{cases} \tag{12}$$

If $\frac{K_0}{n}$ is not a numeral, the weights ties inputs and the N_1-th hidden neuron can be attained the similar method as for the circumstance $K_0 \geq K_1$.

To examination the consequences, when $K_0 \geq K_1$, with these preliminary values of the weights, an input $\mathbf{x} \in H_1$ such:

$$W_j^{(1)} \cdot x + W_{j,0}^{(1)} = 0$$

Means **x** dishonesties on the hyper plane that the *j*-th hidden neuron denotes. This contribution will root only the *j*-th hidden element to have an initiation value adjacent to 1, since

$$net_j^{(1)} = W_j^{(1)} \cdot x + W_{j,0}^{(1)} = 0,$$

where, $net_j^{(1)}$ characterize the net contribution magnitude of the *j*-th hidden neuron. And

$$-\sqrt{\beta} < net_j^{(1)} = 0 < \sqrt{\beta}$$

The other hidden neurons $t \neq j$ will not get activated because

$$net_t^{(1)} = W_t^{(1)} \cdot x + W_{j,0}^{(1)} \neq 0,$$

and

$$\begin{aligned} &if \quad net_t^{(1)} > 0: \quad -\sqrt{\beta} < net_t^{(1)} \sim< \sqrt{\beta} \quad ; \\ &if \quad net_t^{(1)} < 0: \quad -\sqrt{\beta} \sim< net_t^{(1)} < \sqrt{\beta} \end{aligned}$$

Thus, providing the activation of the *j*-th hidden neuron is adjacent to 1 with a QSF and all other neurons are 0, the output neuron will also get activated, since

$$net_1^{(2)} = W_1^{(2)} \cdot y + W_{1,0}^{(2)} = 1 * y_i - 1 = 0,$$
$$-\sqrt{\beta} < net_1^{(2)} = 0 < \sqrt{\beta}$$

where **y** signify the output vector of the hidden layer, $net_1^{(2)}$ characterize the net contribution scale of the output neuron. If $K_0 < K_1$, there is a parallel condition.

5. EXPERIMENTAL RESULT

Activation function results through the application of MATLAB neural network to create an induction motor speed control was shown in Figure 4 and 6 used is S-shaped function. From the output of the function is always between 0 and 1, and as a result of data input and standard output are used.

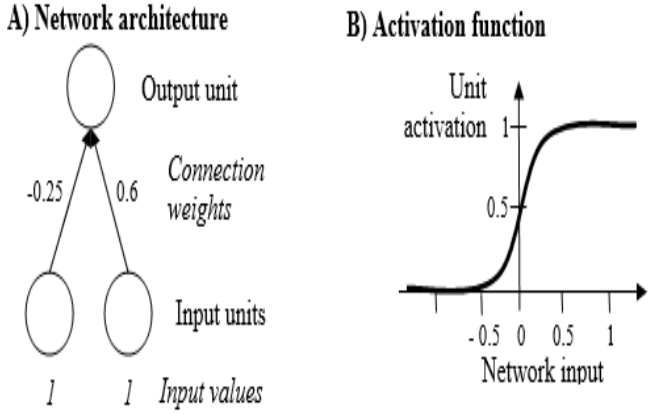

Figure 5 Sigmoidal Activating process

Back-propagation algorithm is iterative gradient algorithm designed to reduce the mean square error between the actual outputs with the expected output of the network between. This is a methodical process of training a multilayer neural network. It has a strong mathematical basis, is effective enough to be used in practice. It is on the basis of the incremental gradient multilayer forward drop events, commonly referred to as the rule network through back-propagation learning rule. It is changing the weights feed forward network computing capability approach. As a gradient descent method, the absolute square error is reduced by the network output evaluation.

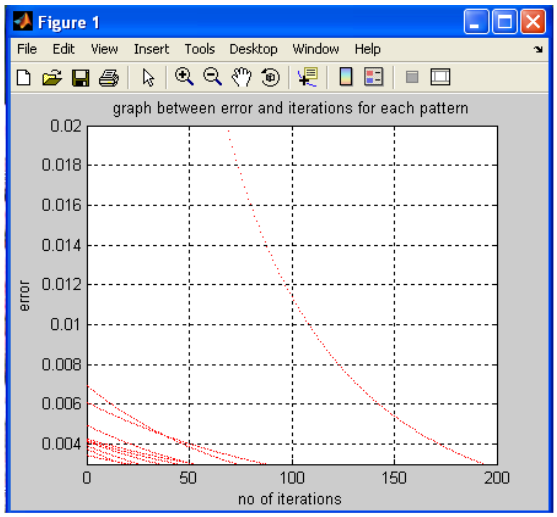

Fig 6: Graph between Error and Iterations for the training input Pattern

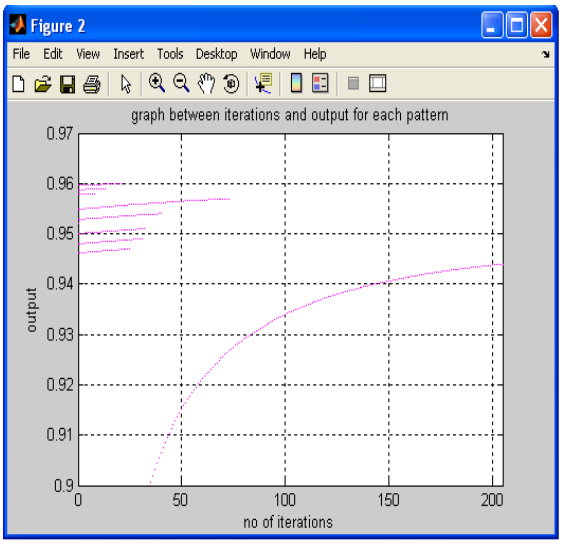

Fig 7: Graph between iterations and output for each training input pattern

The purpose of this network is to educate neural networks to achieve balance those response capability to train and provide a good answer to the ability to enter the way, correct input mode between them. Error minimization method in accordance with back-propagation neural network learning can bring a problem to produce a local minimum of the error function. You can run error is non-negative effect of variables. Training was considered successful value. Select an appropriate training parameters, you must ensure that high-quality solutions within successful computing cycles.

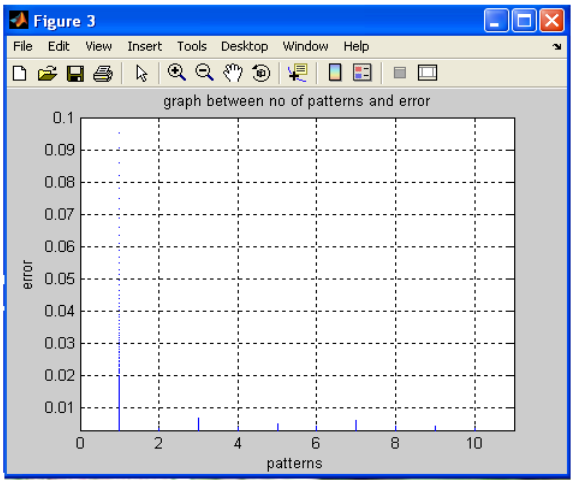

Fig 8: Number of patterns and Error

6. DISCUSSION AND CONCLUSION

This paper describes the design and implementation of neurons can be used in any neural network, the activation function of neurons is designed S-type functions. This design is the use of the tool for the schematic editor program can re-configure the device. Neuron is a neuron micro design. Neural Networks (NN), used in the production of engineering modeling various problems can be there is also three basic types: classification, prediction and functionality. Predicted route in which the robot has to be delivered to the next node (machine) portion, is based on past values of the system state and between the current values of the system state. When the neural network for MATLAB neural network toolbox training, supervised learning algorithm and sigmoid activation function. The neural network prediction in the manufacturing process parameters and characteristics of the learning process time parameters of development. Nominal simulation parameters for the purpose of manufacturing process time (estimated from empirical data), and its uniform distribution is according to the random character used for modeling.

REFERENCES

1. Karlik, B., & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized MLP architectures of neural networks. International Journal of Artificial Intelligence and Expert Systems, 1(4), 111-122.
2. Yong, P. C., Nordholm, S., & Dam, H. H. (2013). Optimization and evaluation of sigmoid function with< i> a priori</i> SNR estimate for real-time speech enhancement. *Speech Communication*, *55*(2), 358-376.
3. Campo, I. D., Finker, R., Echanobe, J., & Basterretxea, K. (2013). Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementation of artificial neurons. *Electronics Letters*, *49*(25), 1598-1600.
4. Wei, L., Shi-Yuan, L., Xiao-Fei, W., & Chuan-Wei, Z. (2011). Parametric analytical model for off-axis illumination sources based on Sigmoid function.
5. Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (pp. 6645-6649). IEEE.
6. Bao, H., Zhao, D., Fu, Z., Zhu, J., & Gao, Z. (2011, July). Application of genetic-algorithm improved BP Neural Network in automated deformation monitoring. In*Natural Computation (ICNC), 2011 Seventh International Conference on* (Vol. 2, pp. 743-746). IEEE.
7. Lin, H., Ding, C., Yuan, L. F., Chen, W., Ding, H., Li, Z. Q., ... & Rao, N. N. (2013). Predicting subchloroplast locations of proteins based on the general form of Chou's pseudo amino acid composition: approached from optimal tripeptide composition. *International Journal of Biomathematics*, *6*(02).
8. Ye, J. (2014). Compound control of a compound cosine function neural network and PD for manipulators. *International Journal of Control*, (ahead-of-print), 1-12.
9. Ito, Y. (1991). Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory. *Neural Networks*, *4*(3), 385-394.
10. Selmic, R. R., & Lewis, F. L. (2002). Neural-network approximation of piecewise continuous functions: application to friction compensation. *Neural Networks, IEEE Transactions on*, *13*(3), 745-751.
11. Poznyak, A. S., Yu, W., Sanchez, E. N., & Perez, J. P. (1999). Nonlinear adaptive trajectory tracking using dynamic neural networks. *Neural Networks, IEEE Transactions on*, *10*(6), 1402-1411.
12. Selmic, R. R., & Lewis, F. L. (2000). Deadzone compensation in motion control systems using neural networks. *Automatic Control, IEEE Transactions on*,*45*(4), 602-613.
13. Ting, W., & Sugai, Y. (1999). A wavelet neural network for the approximation of nonlinear multivariable function. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on* (Vol. 3, pp. 378-383). IEEE.
14. Schiller, H., & Doerffer, R. (1999). Neural network for emulation of an inverse model operational derivation of Case II water properties from MERIS data.*International Journal of Remote Sensing*, *20*(9), 1735-1746.
15. Schäfer, A. M., & Zimmermann, H. G. (2006). Recurrent neural networks are universal approximators. In *Artificial Neural Networks–ICANN 2006* (pp. 632-640). Springer Berlin Heidelberg.
16. Hara, K., & Nakayamma, K. (1994, June). Comparison of activation functions in multilayer neural network for pattern classification. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on* (Vol. 5, pp. 2997-3002). IEEE.