ISSN: 2229 – 6956(ONLINE)
DOI: 10.21917/ijsc.2011.0023

BACKPROPAGATION TRAINING ALGORITHM WITH ADAPTIVE PARAMETERS TO SOLVE DIGITAL PROBLEMS

R. Saraswathi
Department of Bachelor of Computer Applications, The American College, Tamil Nadu, India
E-mail: ponrajsaras@gmail.com

Abstract
An efficient technique namely Backpropagation training with adaptive parameters using Lyapunov Stability Theory for training single hidden layer feed forward network is proposed. A three-layered Feedforward neural network architecture is used to solve the selected problems. Sequential Training Mode is used to train the network. Lyapunov stability theory is employed to ensure the faster and steady state error convergence and to construct and energy surface with a single global minimum point through the adaptive adjustment of the weights and the adaptive parameter β. To avoid local minima entrapment, an adaptive backpropagation algorithm based on Lyapunov stability theory is used. Lyapunov stability theory gives the algorithm, the efficiency of attaining a single global minimum point. The learning parameters used in this algorithm is responsible for the faster error convergence. The adaptive learning parameter used in this algorithm is chosen properly for faster error convergence. The error obtained has been asymptotically converged to zero according to Lyapunov Stability theory. The performance of the adaptive Backpropagation algorithm is measured by solving parity problem, half adder and full adder problems.

Keywords:
Single Hidden Layer, Lyapunov Stability Theory, Adaptive Learning Parameter

1. INTRODUCTION

Recently, neural networks have been extensively studied and used in many areas of science and engineering. The power and usefulness of artificial neural networks have been demonstrated in several applications including speech synthesis, diagnostic problems, medicine, business and finance, robotic control, signal processing, computer vision and many other problems that fall under the category of pattern recognition. For some application areas, neural models show promise in achieving human-like performance over more traditional artificial intelligence techniques.

Feedforward neural networks are known to be universal approximators for nonlinear functions. Training feedforward neural networks can be viewed as a nonlinear optimization problem in which the goal is to find a set of network weights that minimize an error function. The backpropagation approach is a well known approach for supervised learning for neural networks to minimize an error function. One of the major problems with backpropagation is local minimum entrapment. The Lyapunov Stability theory has been applied to detrap the local minima.

Backpropagation, or propagation of error, is a common method of teaching artificial neural networks how to perform a given task. It was first described by Paul Werbos in 1974, but it wasn't until 1986, through the work of David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams, that it gained recognition, and it led to a renaissance in the field of artificial neural network research. Researchers at Ford Motor Company are devising a neural network system that diagnoses engine malfunctions. Marko et al.[1990] trained a neural network to diagnose engine malfunction, given a number of different faulty states of an engine such as open plug, broken manifold etc. The trained network had a high rate of correct diagnoses. Neural Networks have also been used in the banking industry, for example, in the evaluation of credit card applications.

Man et al.[12] has proposed Lyapunov stability based on adaptive backpropagation (LABP) for discrete systems. It can be applied to various aspects of adaptive signal processing. A Lyapunov function of the error between the desired and actual outputs of the neural network is first defined. Guo, H., Gelfand, S.B.,[7] have introduced Analysis of gradient descent learning algorithms for multilayer feedforward neural networks, which is notable for easy stability monitoring and good sensitivity performance. However, their high cost for computing gradients is a problem.

The gradient-based Back-propagation training algorithms may have a slow convergence and the searching for the global minimum point of a cost function may be trapped at local minima during gradient descent. If a neural network has large bounded input disturbances, the global minimum point may not be found. To avoid those problems Backpropagation training algorithm based on Lyapunov Stability theory with adaptive parameters has been used in this paper. The adaptive parameter β, and other positive numbers like λ_1, λ_2, λ_3 and α and the weight selection plays an important role in error convergence. The efficiency of the proposed method is demonstrated for selected problems namely Parity problems and Half-adder and Full-adder problems. The proposed training technique and performance of the algorithm for the selected examples is presented in Section 2 and Section 3 respectively.

2. TRAINING NEURAL NETWORK

In training a three-layered feedforward neural network, the input is presented to the network and it is passed through weighted connections to calculate the net and output in the hidden layer.

2.1 TRAINING METHOD

The output of the jth nonlinear node in the hidden layer can be calculated as:

$$net = \sum_{k=1}^{n} (w_{ij}, x_i) \quad (1)$$

where the nonlinear function F_j(net) is of the form,

$$F_j(net) = \frac{1}{1+e^{-\alpha net}} \quad (2)$$

where α is a positive constant.

Then, this output value of the hidden layer and the weights between the hidden and output layers are used to calculate the output of the neural network.

The output of the neural network can be calculated as:

$$net = \sum(Vij, Hij), \text{ and } Y = net \quad (3)$$

where $V_{i,j}$ is the weight between hidden and output layers.

Then, the error will be measured by comparing the actual output of the network and the target output to ensure that the correct output is derived.

The error e_k can be defined as:

$$e(k) = y(p) - d(p) \quad (4)$$

where k is the time scale, y(p) is the actual output of the network and d(p) is the desired output.

The candidate of the Lyapunov function is chosen to ensure faster error convergence using,

$$V(k) = \beta^k e^2(k) \quad (5)$$

where β is a positive constant and should be greater than 1 in order to guarantee the asymptotic error convergence. If β is closer to 1, the error convergence is very slow. If β is much greater than 1, the error convergence will be very fast.

Then, the weights of the network should be adaptively updated to ensure the error convergence and leads to the actual output.

$$\Delta V(k) = V(k) - V(k-1)$$
$$\Delta V(k) = -(\beta^{k-1} - 1)e^2(k-1) \quad (6)$$

2.2 WEIGHT UPDATION LAWS

The weight updation law used between the hidden and output layer is:

$$V_{ij}(k) = \frac{\beta^{-k/2} e(k-1) + d(p)}{nH_j(k-1) + \lambda_1} \quad (7)$$

The weight updation law used between the input and hidden layer is:

$$W_{ij}(k) = \frac{1}{nx_i(p) + \lambda_2} G_j \frac{\beta^{-k/2} e(k-1) + d(p)}{nV_{ij}(k) + \lambda_3} \quad (8)$$

where λ_1, λ_1 and λ_3 are small positive numbers, n is the number of neurons in the particular layer and

$$G_j(x) = F_j^{-1}(x) \quad (9)$$

The training algorithm will be terminated based on the specified Sum of Squared Error (SSE).

$$SSE = \frac{1}{2}\sum(y(p) - d(p))^2 \quad (10)$$

2.3 TRAINING ALGORITHM

Step1: Construct the three-layered feedforward neural network.

Step2: Generate random numbers from the range [-1,1] for the weight vectors.

Step3: Do steps 4 to 7 for all training patterns one by one.

Step4: Compute output of the node in the hidden layer using (1).

Step5: Compute the output Y using (3).

Step6: Define the error of the network using (4).

Step7: Choose a candidate of the Lyapunov function V(k) using (5).

Step8: Update the weights of the neural network at time instant k using (7) and (8) to make $\Delta V(k) < 0$ as in (6).

Step9: The Steps 3 to 8 is repeated until the expected minimum error (10) is reached.

3. SIMULATION RESULTS

Backpropation Training Algorithm with Adaptive Parameters has been used to solve parity problem, half adder and full adder problems. A single hidden layer network is considered to solve the selected problems intelligently with the hidden layer activation function as $\frac{1}{(1+e^{-\alpha x})}$.

A three-layered feedforward architecture with one output unit has been used for 3-bit and 4-bit parity problems. For Half adder and Full adder problems, a three-layered feedforward architecture with two output units has been used. The selected problems have been executed using 5 different initial weights. The initial values of the adaptive parameter β should be carefully chosen to get faster and asymptotic error convergence. If the initial value of β is chosen to be much closer to 1, the error convergence will be very slow. Otherwise, if its initial value is chosen to be much greater than 1, the error convergence will be very fast.

The tables and figures show the results obtained using one set of weight.

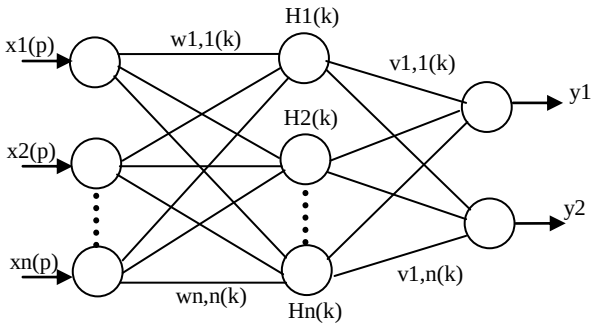


Fig.1. Single hidden layer neural network with two outputs

Sequential mode of training has been applied to train the network to solve all the selected problems. To train the network, the initial values of the weights are chosen randomly from the uniformly distributed numbers within the range [-1,1].

3.1 THREE-BIT PARITY

The training patterns used for 3-bit parity are: 000 --- 0; 001 --- 1; 010 --- 1; 011 --- 0; 100---1; 101---0; 110---0;111---1. Three-layered feedforward architecture with input, hidden and output neurons are 3-4-1 respectively, is used to solve this problem. The initial value of the adaptive learning parameter β is chosen as 1.1. The β value is incremented by 0.0005 per iteration. The parameters used are: $\alpha = 0.03$; $\lambda_1 = 0.88$; $\lambda_2 = 0.69$; $\lambda_3 = 0.85$. Table.1. depicts the minimized error and number of epochs during training.

Table.1. Simulation results for 3-bit Parity Problem

Problem	No of hidden Neurons	SSE	Termination Condition	Epoch	Time in Secs.
3-bit Parity	4	0.001150	0.0012	765	0.89

The following table shows the expected and obtained output for the 3-bit parity problem.

Table.2. Expected vs Obtained output for 3-bit parity problem

Input	Expected Output	Obtained Output
000	0.1	0.099645
001	0.9	0.919476
010	0.9	0.917117
011	0.1	0.10062
100	0.9	0.926101
101	0.1	0.098562
110	0.1	0.09904
111	0.9	0.930705

The output chart for the 3-bit parity problem showing expected and obtained output is as follows:

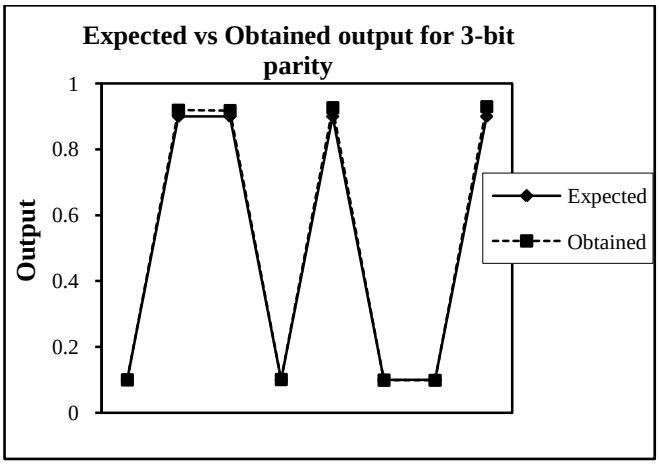


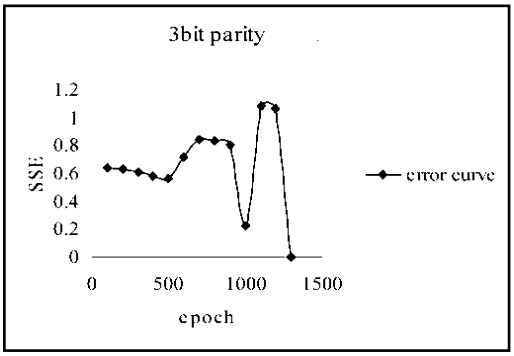
Fig.2. Expected vs obtained output chart for 3-bit parity problem

β with various initial values and the corresponding reduced number of epochs to attain the output is shown in Table.3.

Table.3. Simulation results for 3-bit Parity with different initial values of β

Problem	Initial value of β	SSE	Epoch
3-bit Parity	1.1	0.001150	764
	1.2	0.001150	732
	1.3	0.001150	701
	1.5	0.001150	639

The learning curve for the 3-bit parity problem without using the adaptive parameter β is as follows:



In the above figure, the error reaches a local minimum point at 1000th epoch. To avoid the entrapment of local minima, the adaptive parameter should be used and chosen properly.

After using the adaptive learning parameter β, the learning curve for the 3-bit parity problem depicting SSE and number of epochs is as follows:

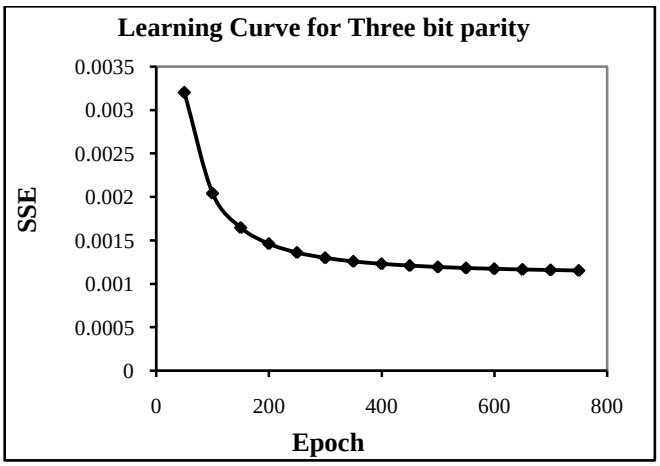


Fig.3. Learning Curve for 3-bit Parity

3.2. FOUR-BIT PARITY

The training patterns used for 4-bit parity are: 0000 --- 1; 0001 --- 1; 0010 --- 1; 0011 --- 0; 0100 --- 1; 0101 --- 0; 0110 --- 0; 0111 --- 1; 1000 --- 1; 1001 --- 0; 1010 --- 0; 1011 --- 1; 1100 --- 0; 1101 --- 1; 1110 --- 1; 1111 --- 0. A three-layered feedforward architecture with input, hidden and output neurons are 4-6-1 respectively, is used to solve this problem. The initial value of the adaptive learning parameter β is chosen as 1.11. The β value is incremented by 0.0005 per iteration. The parameters used are: $\alpha = 0.04$; $\lambda_1 = 1.4$; $\lambda_2 = 1.65$; $\lambda_3 = 1.12$. Table 4 shows the expected and obtained output for the 4-bit parity problem during training.

Table.4. Expected and obtained output for 4-bit parity problem

Input	Expected Output	Obtained Output
0000	0.1	0.104235
0001	0.9	0.936738
0010	0.9	0.93053
0011	0.1	0.100442
0100	0.9	0.941663
0101	0.1	0.1
0110	0.1	0.101298
0111	0.9	0.942687
1000	0.9	0.929562
1001	0.1	0.09956
1010	0.1	0.101059
1011	0.9	0.944361
1100	0.1	0.099215
1101	0.9	0.945311
1110	0.9	0.928893
1111	0.1	0.097835

The difference between the expected and obtained output of the 4-bit parity problem is shown in Fig.4.

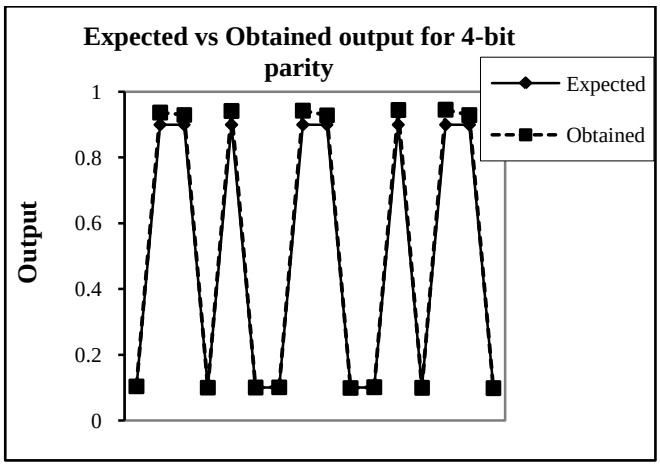


Fig.4. Expected vs obtained output for 4-bit parity problem

Table.5. depicts the minimized error and number of epochs during training.

Table.5. Simulation results for 4-bit Parity Problem

Problem	No of hidden Neurons	SSE	Termination Condition	Epoch	Time in Secs.
4-bit Parity	6	0.005798	0.0058	1831	1.10

β with various initial values and the corresponding reduced number of epochs to attain the output is shown in Table.6.

Table.6. Simulation results for 4-bit Parity with different initial values of β

Problem	Initial value of β	SSE	Epoch
4-bit Parity	1.1	0.005798	1831
	1.2	0.005798	1825
	1.3	0.005798	1818
	1.5	0.005798	1804

The learning curve for the 4-bit parity problem depicting SSE and number of epochs is as follows:

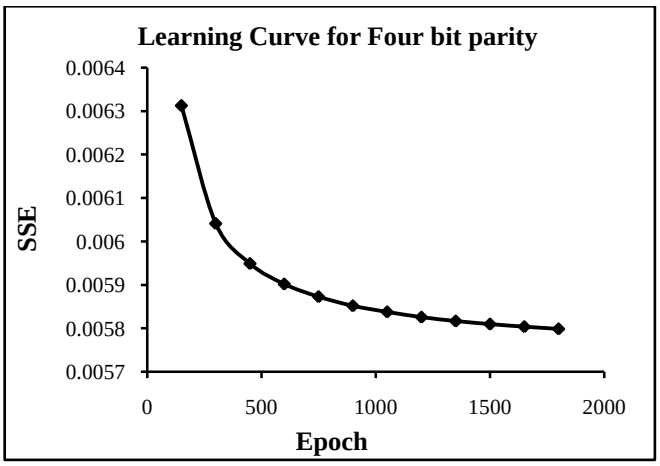


Fig.5. Learning Curve for 4-bit Parity

3.3. HALF ADDER

The training patterns used for Half Adder Problem are: 00 --- 0 0; 01 --- 0 1; 10 --- 0 1; 11 --- 1 0; A three-layered feedforward architecture with 2-5-2 input, hidden and output neurons respectively, is used to solve this problem. The initial value of the adaptive learning parameter β is chosen as 1.1. The β value is incremented by 0.01 per iteration. The parameters used are: $\alpha = 0.12$; $\lambda_1 = 0.999$; $\lambda_2 = 0.85$; $\lambda_3 = 1.16$.

Table.7. shows the expected and obtained output for half adder problem.

Table.7. Expected vs obtained output for half adder

Input	Expected Sum	Obtained Sum	Expected Carry	Obtained Carry
00	0.1	0.105173	0.1	0.105153
01	0.9	0.936635	0.1	0.104853
10	0.9	0.900576	0.1	0.100085
11	0.1	0.096712	0.9	0.870388

The curve depicts the difference between the expected and obtained sum in Fig.6.

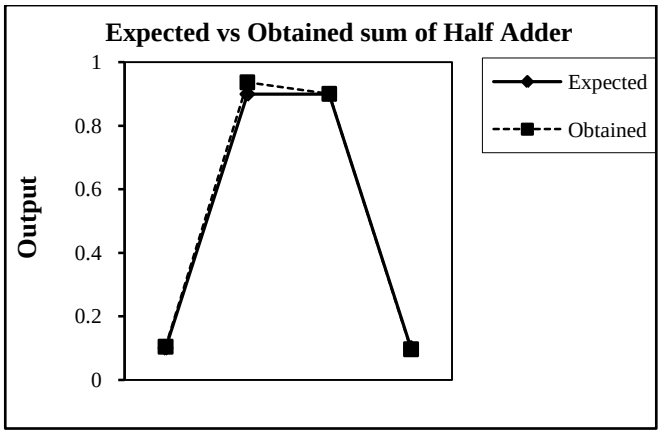


Fig.6. Expected vs obtained sum of half adder

The curve depicts the difference between the expected and obtained carry in Fig.7.

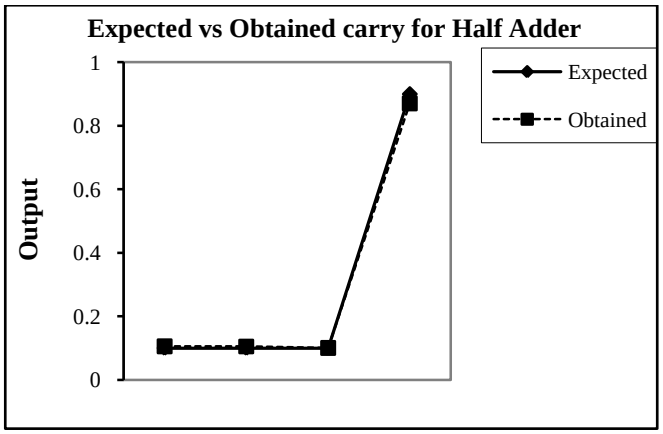


Fig.7. Expected vs obtained carry of half adder

Table.8. shows the minimized error and number of epochs during training.

Table.8. Simulation results for Half Adder Problem

Problem	No of hidden Neurons	SSE	Termination Condition	Epoch	Time in Secs.
Half adder	5	0.001153	0.0012	987	0.11

β with various initial values and the corresponding reduced number of epochs to attain the output is shown in Table.9.

Table.9. Simulation results for Half Adder with different initial values of β

Problem	Initial value of β	SSE	Epoch
Half Adder	1.1	0.001153	987
	1.2	0.001153	984
	1.3	0.001153	981
	1.5	0.001153	976

The learning curve for the Half Adder problem depicting SSE and number of epochs is as follows:

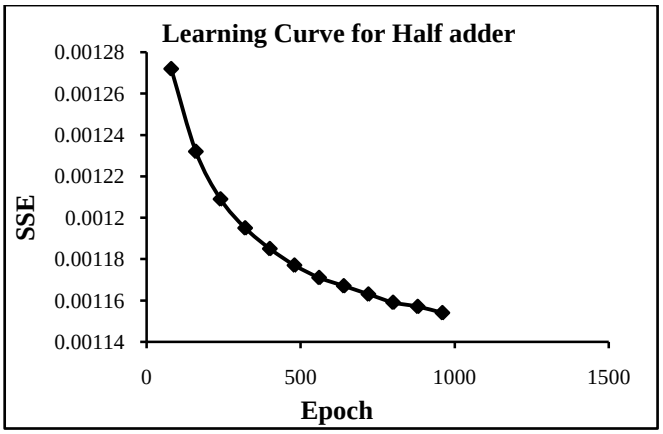


Fig.8. Learning Curve for Half Adder

3.4. FULL ADDER

The training patterns used for Full Adder problem are: 000 --- 0 0; 001 --- 0 1; 010 --- 0 1; 011 --- 1 0; 100--- 0 1; 101---1 0; 110 --- 1 0;111--- 1 1. A three-layered feedforward architecture with 3-5-2 input, hidden and output neurons respectively, is used to solve this problem. The initial value of the adaptive learning parameter β is chosen as 1.1. The β value is incremented by 0.001 per iteration. The parameters used are: $\alpha = 0.02$; $\lambda_1 = 1.14$; $\lambda_2 = 0.89$; $\lambda_3 = 0.9$.

Table.10. shows the expected and obtained output of full adder problem.

Table.10. Expected vs obtained output of full adder

Input	Expected Sum	Obtained Sum	Expected Carry	Obtained Carry
000	0.1	0.100392	0.1	0.100392
001	0.9	0.919028	0.1	0.102262
010	0.9	0.917724	0.1	0.101995
011	0.1	0.102599	0.9	0.912015
100	0.9	0.913689	0.1	0.101873
101	0.1	0.101306	0.9	0.910256
110	0.1	0.10116	0.9	0.910432
111	0.9	0.934784	0.9	0.934804

The following figure depicts the difference between expected and obtained sum of full adder.

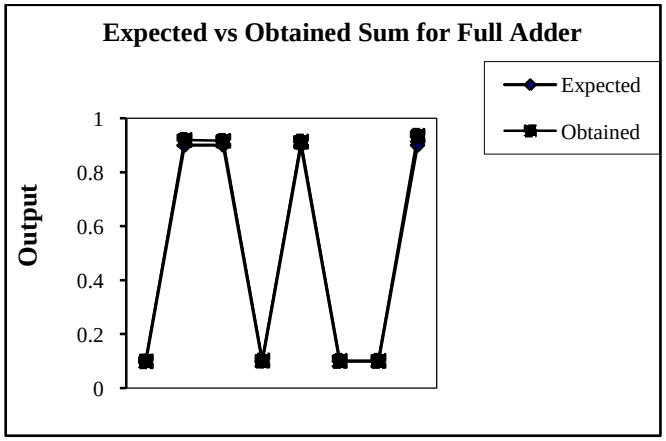


Fig.9. Expected vs obtained sum of full adder

Expected and obtained carry of full adder problem is depicted in Fig.10.

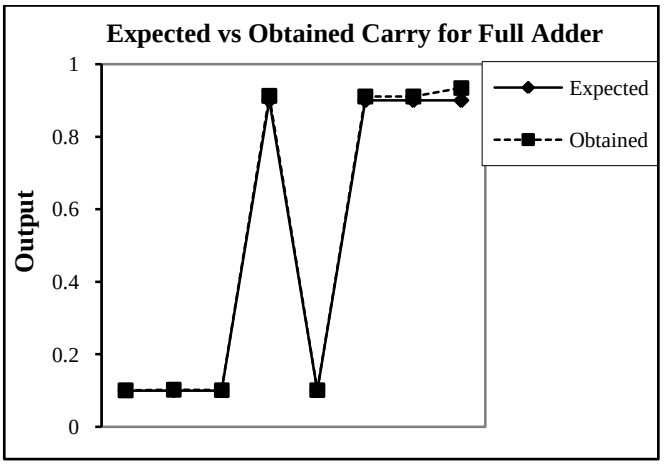


Fig.10. Expected vs obtained carry of full adder

Table.11. depicts the minimized error and number of epochs during training.

Table.11. Simulation results for Full Adder Problem

Problem	No of hidden Neurons	SSE	Termination Condition	Epoch	Time in Secs.
Full adder	5	0.001833	0.002	656	0.17

β with various initial values and the corresponding reduced number of epochs to attain the output is shown in Table.12.

Table.12. Simulation results for Full Adder with different initial values of β

Problem	Initial value of β	SSE	Epoch
Full Adder	1.1	0.001833	656
	1.2	0.001833	642
	1.3	0.001833	628
	1.5	0.001833	601

The learning curve for the Full Adder problem depicting SSE and number of epochs is as follows:

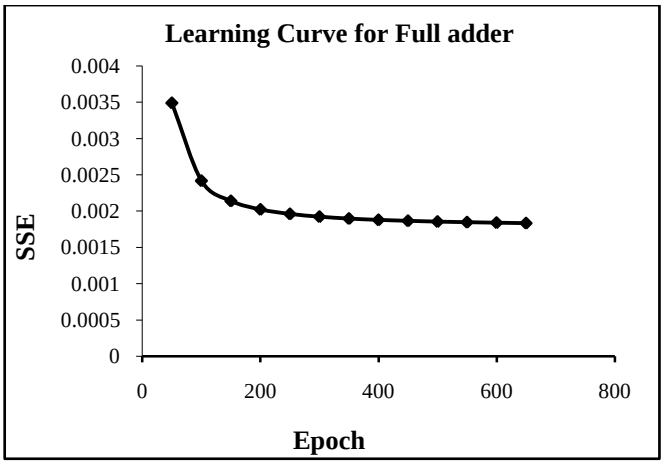


Fig.11. Learning Curve for Full Adder

4. CONCLUSION

The Backpropagation training algorithm with adaptive parameters has been applied to solve the digital problems namely parity, Half adder and Full adder problems. The efficiency of the algorithm is demonstrated by solving the problems and the simulation results are shown through the Tables. The error convergence are shown using charts in the simulation results.

The adaptive learning parameter β plays a vital role in the error convergence. The convergence of error may be fast or slow depending on the initial value of the adaptive learning parameter β. If the initial value of β is chosen to be much closer to 1, the error convergence will be very slow. Otherwise, if its initial value is chosen to be much greater than 1, the error convergence will be very fast.

The selected algorithm gives fast error convergence and it avoids local minima. Lyapunov stability theory used in this algorithm guarantees that the error of the network converges to zero. Adaptive learning parameter gives a steady state error convergence. The backpropagation algorithm with adaptive parameters requires a smaller number of neurons and epochs with better results.

ACKNOWLEDGEMENT

I am grad to express my deep sense of gratitude and my indebtedness to Dr. T. Kathirvalavakumar, M.Sc., P.G.D.C.A., M.Phil., Ph.D., Reader and Head, Department of Computer Science, V.H.N.S.N. College, Virudhunagar, for his valuable guidance and encouragement during this work.

I extend my sincere gratitude to Dr. T. Joshva Devadas, M.Sc., B.ED., PGDCA., Ph.D., Department of MCA, The American College for his valuable guidance and timely help to finish this work.

I wish to express my deep indebtedness to my family members for their tireless support and affectionate encouragement showed on me during the entire course of this work.

My heartful thanks to all my friends for giving valuable information and suggestions to carry out the work successfully.

REFERENCES

[1] Abid, S., Fnaiech, F., Najim, M., 2001, "A fast feedforward training algorithm using a modified form of the standard backpropagation algorithm", IEEE Transactions on Neural, pp.424 - 430.

[2] Alessandri, A., Sanguineti, M., Maggiore, M, 2002, "Optimization-based learning with bounded error for feedforward neural networks", IEEE Transactions on Neural Networks, pp.261 - 273.

[3] Cerqueira, J.J.F., Palhares, A.G.B., Madrid, M.K, 2002, "A simple adaptive back-propagation algorithm for multilayered feedforward perceptrons", IEEE International Conference on Systems, Man and Cybernetics 3, Vol.3, pp.6.

[4] De-Shuang Huang, 1998, "The local minima-free condition of feedforward neural networks for outer-supervised learning", IEEE Transactions on Systems, Man, and Cybernetics, Vol.28, No.3, pp.477 - 480.

[5] Dilip Sarkar, 1995, "Methods to speed up error back-propagation learning algorithm", ACM Computing Surveys, Vol.27, No.4, pp.519 - 544.

[6] Ebner, Th., Magele, Ch., Brandstatter, B.R., Richter, K.R, 1998, "Utilizing feedforward neural networks for acceleration of global optimization procedures [SMES problems]", IEEE Transactions on Magnetics,Vol.34, pp.2928 - 2931.

[7] Guo, H., Gelfand, S.B., 1991, "Analysis of gradient descent learning algorithms for multilayer feedforward neural networks", IEEE Transactions on Circuits and Systems, Vol.38, pp.883 - 894.

[8] Guang-Bin Huang, Yan-Qiu Chen, Babri, H.A., 2000, "Classification ability of single hidden layer feedforward neural networks", IEEE Transactions on Neural Networks, Vol.11, pp.799 - 801.

[9] Hown-Wen Chen, Von-Wun Soo, 1992, "An adaptive back-propagation learning method: A preliminary study for incremental neural networks", International Joint Conference on Neural Networks, Vol.1, pp.713 - 718.

[10] Huang Zhiwu, Shan Yongteng, Gui Weihua, Nian Xiaohong, 2007, "Stability Analysis and Design of Adaptive Observer Based on Speed Sensorless Induction Motor", Control Conference, pp.28 - 32.

[11] Lyshevski, S.E., 2001, "Nonlinear microelectromechanical systems (MEMS) analysis and design via the Lyapunov stability theory", IEEE Conference on Decision and Control, Vol.5, pp.4681 - 4686.

[12] Man, Z., Phooi, S.K.,Wu, H.R., 1999, "Lyapunov stability-based adaptive backpropagation for discrete time system", Signal Processing and Its Applications, Vol.2, pp.661 - 664.

[13] Seng Kahphooi, Zhihong, M., Wu, H.R., 1999, "Nonlinear adaptive RBF neural filter with Lyapunov adaptation algorithm and its application to nonlinear channel equalization", International Symposium on Signal Processing and Its Applications, Vol.1, pp.151- 154.

[14] Seng, K. P., Liu, H. L., and Z. Man, 2002, "Further investigation of Lyapunov stability theory", WSEAS Transactions on Mathematics, Vol.1, pp.108 - 110.

[15] Z.Man, H.R.Wu, W.Lai and T.Nguyen, 1998, "Design of adaptive filters using Lyapunov stability theory", in Proc, 6th IEEE Int.Workshop Intell. Signal Process, Commun. Syst., pp.304-308.

[16] Seng, KP, Man, Z, Wu, HR, 2000, "Complex RBF Network with Lyapunov Theory Training Algorithm", Proceedings of the International Conference on Artificial Intelligence in Science and Technology, pp.44 - 49.

[17] Shevitz, D., Paden, B., 1994, "Lyapunov stability theory of nonsmooth systems", IEEE Transactions on Automatic Control, Vol.39, pp.1910 - 1914.