International Journal on Emerging Technologies **11**(2): 1034-1038(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Extended Bipolar Sigmoid Algorithm for Enhancing Performance of Constructive Neural Network

Jaswinder Kaur[1] and Neha Gupta[2]
[1]Ph.D. Scholar, Department of School of Engineering & Technology Ansal University, Gurgaon, India.
[2]Assistant Professor, Department of School of Engineering & Technology Ansal University, Gurgaon, India.

(Corresponding author: Jaswinder Kaur)
(Received 01 February 2020, Revised 23 March 2020, Accepted 25 March 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: To build a neural network the main point to be considered is how to select the model. Various approaches like Constructive Networks, Pruning or Destructive Networks, Hybrid Networks etc. can be used for building a neural network. In this paper constructive approach is considered for building the network. The proposed algorithm Extended Bipolar Sigmoid Algorithm (EBSA) based on constructive type of network built in cascaded style. It constructs a minimal neural network dynamically starting with one hidden node and one hidden layer, and new hidden layer with one hidden node is added when the network is not able to converge properly. When there is no significant reduction in error after some cycles the residual error is required to be reduced by adding a candidate node and analyzing the effect on mean square error. It is tested on ten real time regression problems. From the results it can be seen that it has better generalization performance and fault tolerance ability.

Keywords: Constructive Networks, Pruning or Destructive Networks, Hybrid Networks, Extended Bipolar Sigmoid Algorithm, Hidden Nodes and Candidate Nodes.

Abbreviations: EBSA, Extended Bipolar Sigmoid Algorithm; SC, Standard Cascade; MSE, Mean Square Error; Std. dv., Standard Deviation.

I. INTRODUCTION

A good network topology is required for achieving good performance of an artificial neural network. Artificial neural networks mapping ability is always dependent on the size and structure of the network. A large neural network with many hidden layers and hidden nodes may result in poor generalization performance. A small neural network built with few hidden nodes and few hidden layers may not be able to solve complex problems. It is difficult to select an optimal neural network topology. It is usually decided using trial and error method.

An artificial neural network is of layered type namely input layer, hidden layer and output layer. There can be one or more hidden layers, only one input layer and one output layer. Each layer receives its input from layer pervious to it and feeds the output to the layer next to it. This is a feed forward network. It can be single layered (one input layer and one output layer) or multilayered (one input layer, one output layer and one or more hidden layer). The output of hidden node is sum of weighted values of input on which activation function is applied. This output is applied to next layer till following the same procedure the last layer output layer is reached. Mostly Backpropogation algorithm is used for learning in theses neural networks. Only problem with this algorithm is the convergence rate is very poor [1].

The main problem for neural network building is selection of model (complexity). There are various approaches for altering the architecture of neural network namely constructive network, pruning or destructive network, hybrid (constructive-pruning) network etc.

II. APPROACHES FOR BUILDING NEURAL NETWORK

Constructive Network begins with network of small size and then it grows according to the problem to be solved. There is no need to decide the size of the network initially. This is the main advantage of this method. These are very efficient for hardware implementation as they are small in size [2].

Constructive networks are built by adding new features like hidden nodes, hidden layers and connections in turn changing the network topology. These are incremental networks which change the topology during the process of learning.

Constructive Algorithms can implement input output mapping of any problem provided with sufficient hidden nodes. The performance of the algorithm depends on size of network, technique of weight initialization, activation function used, number of hidden nodes, connections, number of layers and algorithm used for learning. The network can built using a number of input nodes and a number of output nodes which are decided by the problem in hand. The number of hidden nodes, layers and connections can be altered using various learning algorithms. While training the network the error is reduced using gradient descent algorithm like back propagation, quickprop etc. There are various activation functions like tanh function, Gaussian function, symmetric sigmoid function, asymmetric sigmoid function, radial function etc. which can be used in these networks [3].

Constructive networks can be expanded with layers and hidden nodes. Various methods can be used to expand the network namely Inverted Pyramid Construction in

which new growing hidden nodes have connections with all previous hidden nodes, Tower Network Construction, Tiling Network, Upstart Neural network, Resource Allocation Neural Network, Cascade Correlation Neural Network etc.

These networks have very fast ability of learning, requires very less weights to achieve the same accuracy and the number hidden nodes are automatically adjusted depending on the complexity of the problem.

Destructive Network or Pruning Network begins with large network. During training the unimportant hidden nodes are removed that are participating less. This is known as Pruning the Network. The network learns quickly as the size of network is very large. These networks avoid the over fitting problem due to large size network. The hidden nodes whose magnitude is small, weights of hidden node is insensitive are pruned from the network. Also a penalty is added to energy term for decay of weights of hidden nodes so that their participation is least [4].

Hybrid Network is combination of both the constructive and destructive network. In this the neural network is first built using constructive approach in which hidden nodes are added one by one to this minimal size while training it. When the network becomes considerably very large due no proper stoppage condition, destructive approach is used. This approach removes the hidden nodes which are participating less in improving the performance and hence are irrelevant for the network [5].

III. CONSTRUCTIVE NETWORK BUILDING APPROACHES

In this paper constructive approach of building the network will be used. Various constructive algorithms are studied before constructing the new algorithm EBSA.

Cascade Correlation Algorithm [6] is mostly preferred as feed forward method over other algorithms. CA changes the architecture of the neural network while training. It starts with one layer with input and output nodes fully connected. It installs hidden nodes which are selected from a pool of nodes known as candidate nodes. Candidate nodes which have highest correlation are selected for installation in the network. These are connected to earlier installed hidden nodes and input nodes. Weights of installed hidden node are frozen and training of output nodes is performed. New hidden node is added and output nodes are retrained till the maximum hidden nodes are being added. Training is continued till not much improvement further.

Dynamic Learning Algorithm [7] uses methodologies like Cascade Correlation and pruning algorithm for producing a compact neural network.

An extension to Cascade Correlation Training [8] is based on some variations in original cascade algorithm. It focuses on training candidate nodes with different patience parameter. In this algorithm firstly each candidate node has a patience parameter and solves the problem of this node running out of patience. Secondly, there are subgroups of candidate node having same features. This group is trained in blocks. The nodes in candidate can be divided into two parts based on activation function (one part with Gaussian function and second part with sigmoid function). The candidate pool will be trained with different patience value in two groups.

A New Learning Algorithm for Feed Forward Neural Networks [9] uses incremental training method for training patterns which are learned one by one. It starts with one training pattern and one hidden node in hidden layer. It uses weight scaling techniques for escaping from the problem of local minima. After various attempts of controlling local minima a new hidden layer is added to the network. An optimization method is used for reaching the error tolerance level. It creates a minimal network and after adding hidden layer, some or no training is required for bringing down the error.

Cascade Error Projection [10] is a method based on cascade correlation which uses single layer perception followed by calculation for next layer. It is a reliable and fast learning method in hardware. It is feasible for hardware and software based methods. It can be started with weight zero instead of assigning random values. There is no pool of candidate nodes. Instead, only one node is added at a time to the hidden layer. Due to this the hardware is very simple.

Recurrent Cascade Correlation Architecture [11] is a recurrent form of cascade architecture. In this new nodes having recurrent connections are being added to the network one by one when required while training. It learns from examples while mapping inputs into outputs desired by the problem.

Dynamic Node Creation in Backpropagation Networks [12] tries to solve issues like training of large neural networks and testing of neural networks having different number of nodes in hidden layer. In this hidden nodes are added in sequential manner and training is done using backpropogation algorithm to achieve the required accuracy. In this the hidden node is connected to all the hidden nodes of the previous layer and it is also connected to all the next layers. Minimal network is constructed using this method.

IV. EXTENDED BIPOLAR SIGMOID ALGORITHM (EBSA)

Extended Bipolar Sigmoid Algorithm (EBSA) is an incremental algorithm based on cascaded type learning network. It is starts with a simple network and continues with addition of nodes and connections between nodes in order to reduce the error. The network topology and calculation of connection weighs are done using a learning algorithm like quickprop. Input nodes receive input signal and output nodes give results to the corresponding input signals. Connections are links between different nodes.

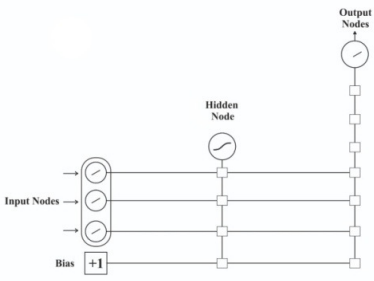

Fig 1. Extended Bipolar Sigmoid Neural Network.

It constructs a neural network dynamically. The input nodes of input layer are fully connected to hidden nodes in hidden layers which are connected to output nodes in output layer. The bias input is permanently set to 1.In this first initialization of network is done, then normal network with no hidden nodes is trained, then candidate nodes are trained with new activation function, then selected node is added to network.

The idea is develop an optimal neural network which generalizes well. This algorithm is built after studying the past methodologies which helps in selection of best techniques for building the network. It has ability to converge at fast speed as compared to other non incremental algorithms. In this training is carried out and less weight adjustments are made at each training step.

EBSA is explained as follows

Initialization Phase: This is the first phase in which there are some input nodes, no hidden node and one output node. Then initialization of weights is done and parameters of training are fixed. The input nodes are connected to output nodes. Bias input is set to 1. The training is done using quickprop algorithm.

Training Phase: In this phase, calculations of the weights are being carried out using learning algorithm. Then all weights are updated that satisfy the given condition. If the maximum size network is built, training is stopped. If there is significant improvement after a given number of cycles, then go to Training step (previous step) otherwise go to Candidate training step (next step).

Candidate Training Phase: In this phase a pool of candidate hidden nodes is created. Every candidate node has different initial weight, has temporary connection to input node and to a virtual output node. Extended Bipolar Sigmoid Activation function is used at the hidden layer. It is given by the equation

$$f(u) = \frac{1-e^{v}}{1+e^{-v}} - s \quad (1)$$

where s is ½. It is trained in order to reduce the corresponding output error for a number of cycles. The best candidate is considered as hidden node to be added to the network. All the temporary hidden nodes and their weights are removed. The selected new node is added and the training of weights is done in order to minimize the error. Then again train the network with training phase.

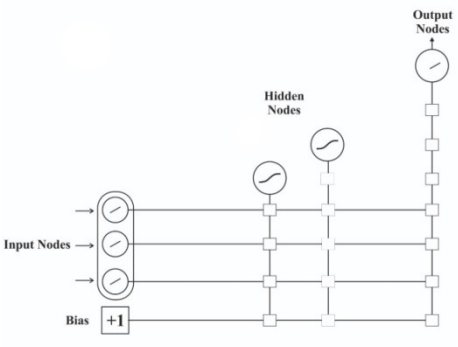

Fig. 2. Extended Bipolar Sigmoid Neural Network.

The features are learned incrementally which are required for training the network as it progresses. The main aim is to build neural network with minimal hidden nodes.

The stoppage condition in this algorithm is the maximum hidden nodes that can be added to the neural network.

V. RESULTS OF SIMULATION

This section presents the comparison of proposed algorithm EBSA with the Standard Cascade (SC) [13]. Some regression problems of real time and different dimensions are being worked out to check how much effective is the new algorithm. Neural networks performance is measured in terms of mean square error for all the following problems.

A. 1 Dimensional Regression Function

$$u(x) = 0.2\left\{1+\frac{1}{10}(v-7)^2\right\}\cos(2v) + 0.5\, e^{-2v}\sin(2v - 0.1\pi) \quad (2)$$

This regression function is very complicated. 1200 randomly distributed samples between 0 and 1 are selected for training the network. 100 exemplars are used for training, 100 for validation and 1000 are used for testing.

Table 1: One dimensional regression function results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46905	0.44188	5	1
Max	0.47891	0.47834	10	10
Mean	0.47402	0.4674	8.6	7.4
Std. dv.	0.003687	0.010743	1.8974	3.2042

From the results it is clear that EBSA performs better and uses than hidden nodes as compared to standard cascade.

B. 1 Regression Function

$$u = \frac{1}{(v-0.3)^2+1} + \frac{1}{(x-0.9)^2+0.04} - 6 \quad (3)$$

This one dimensional regression function uses twelve hundred randomly distributed samples between -0.5 to 1.5. One hundred samples are taken for training, another one hundred for validation and one thousand for testing.

Table 2: 1D regression function results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46932	0.43843	4	2
Max	0.47953	0.47715	10	10
Mean	0.47352	0.45966	9	6.9
Std. dv.	0.003273	0.012634	1.8257	2.8067

From the results it can be noted that EBSA performs better than standard cascade and uses less hidden nodes.

C. 2 Dimensional Function

$$u = \sin\left(2\pi\sqrt{v_1^2 + v_2^2}\right) \quad (4)$$

This a two dimensional regression function in which one thousand four hundred randomly distributed exemplars in range of -1 and 1 are selected. Two hundred exemplars are used for training, another two hundred exemplars are used for validation and one thousand exemplars are used for testing.

Table 3: 2 dimensional regression function results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.47192	0.44909	5	1
Max	0.47816	0.47892	10	10
Mean	0.475	0.4661	8.5	6.6
Std. dv.	0.002226	0.011237	1.4337	2.8752

From the results in the above table it can be seen that EBSA is better in performance when compared to standard cascade. It requires less hidden nodes than standard cascade.

D. 2 Dimensional Function

$$u = e^{(v_1 \sin(\pi v_2))} \quad (5)$$

This is a two dimensional regression function which uses one thousand four hundred randomly distributed samples in range of -1 and 1. Two hundred samples are used for training, another two hundred samples are used for validation and one thousand samples are used for testing.

Table 4: 2D regression function results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46886	0.4481	1	3
Max	0.47985	0.47757	10	10
Mean	0.47419	0.46185	7.6	7.4
Std. dv.	0.00382	0.011787	2.9136	2.7162

From the results in the above table it can be analyzed that EBSA performs better than standard cascade. EBSA uses slightly less hidden nodes as compared to standard cascade

E.3 Dimensional Function

Three Dimensional Simple Analytical Function (SAF) is expressed as

$$u = \frac{1}{1+e^{-e^{v_1}+(v_2-0.5)^2+3\sin(\pi v_3)}} \quad (6)$$

v is uniform between 0 and 1.Three hundred samples selected randomly are generated for training, Three hundred samples are generated for verification of network generalization and One Thousand samples are generated for testing the network. Sixteen hundred uniformly distributed samples are generated randomly between 0 and 1. Ten hidden nodes are added to the maximum and each hidden node is trained maximum till 800 epochs.

Table 5: 3D regression function results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46661	0.43843	4	2
Max	0.47953	0.47757	10	10
Mean	0.47261	0.45843	9.1	6.6
Std. dv.	0.0038142	0.013707	1.8529	2.7968

From results above in the table it can be seen EBSA performs better and needs less hidden nodes when compared with standard cascade.

F. Boston Housing Problem

The Boston house price prediction problem is a real world problem. It is available in the UCI Machine Learning Repository. The data set used is sampled randomly in which all inputs and outputs are in the range of -1 and 1.

It has 506 information entries having 13 attributes of houses in the suburbs of Boston. The motive is to predict the prices of these houses located in Boston depending on the attributes like number of rooms, crime rate per capita of town, location of house, property tax etc. [14].

Table 6: Boston Housing problem results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46905	0.44186	5	1
Max	0.47891	0.47834	10	10
Mean	0.47402	0.46616	8.6	7.5
Std. dv.	0.0037773	0.013933	1.8974	3.2745

From the results EBSA is slightly better in comparison to standard cascade and also needs less hidden nodes.

G. Auto MPG Problem

It is about consumption of fuel miles per gallon for cars in the city for different types of cars. It has 8 attributes out of which five are continuous and three are discrete. It has attributes like car name, origin, model year, displacement, horsepower etc. It predicts automobiles fuel efficiency. It is available in UCI Machine Learning Repository [15].

Table 7: Auto MPG problem results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46254	0.44836	6	3
Max	0.47931	0.479	10	10
Mean	0.47451	0.47303	8.8	7.7
Std.dv.	0.004940	0.009583	1.5492	2.4967

From the above results it can be seen that EBSA performs slightly better than standard cascade. It also uses less number of hidden nodes.

H.4 Dimensional Function

$$u = e^{(2v_1 \sin(\pi v_4))} + \sin(v_2 v_3) \quad (7)$$

This four dimensional regression additive function uses one thousand eight hundred randomly distributed samples in range of -0.25 and 0.25. Four hundred samples are used for training, another four hundred samples are used for validation and one thousand samples are used for testing.

Table 8: 4D regression function results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46905	0.44188	5	1
Max	0.47891	0.47834	10	10
Mean	0.47402	0.4674	8.6	7.4
Std. dv.	0.003682	0.010743	1.8974	3.2042

From the above results it can be seen EBSA performs better with less number of hidden nodes.

I. 4 Dimensional Function

$$u = (v_1 + 10v_2)^2 + 5(v_3 - v_4)^2 + (v_2 - 2v_3)^4 + 10(v_1 - v_4)^4 \quad (8)$$

In this four dimensional function one thousand eight hundred randomly distributed samples are selected. Four hundred are used for training, another four hundred for validation and one thousand for testing.

Table 9: 4D regression function results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46932	0.43843	4	2
Max	0.47953	0.47715	10	10
Mean	0.47352	0.45966	9	6.9
Std. dv.	0.003273	0.012634	1.8257	2.8067

From the above table it can be seen that EBSA performs better than standard cascade and uses less number of hidden nodes.

J. 5 Dimensional Function

$$u = 0.0647\,(12 + 3v_1 - 3.5v_2^2 + 7.2v_3^3)(1 + cos4\pi v_4)(1 + 0.8sin3\pi v_5) \quad (9)$$

Two thousand random samples are generated. Five hundred samples are used for training, five hundred for validation and another one thousand for testing.

Table 10: 5D regression function results.

	MSE		Hidden Nodes	
	SC	EBSA	SC	EBSA
Min	0.46905	0.44188	5	1
Max	0.47891	0.47824	10	10
Mean	0.47402	0.4674	8.6	7.4
Std. dv.	0.003682	0.010743	1.8974	3.2042

From the above results it can be seen that EBSA performs slightly better than standard cascade. It also needs slightly less number of hidden nodes.

VI. CONCLUSION

The extension to Bipolar Sigmoid Algorithm presented as EBSA explained in the above sections is very effective in aiding neural network training. From the experiments it is clear that the use of improved activation function improves performance of neural network. It minimized the number of hidden nodes with no compromise in network generalization performance. It is an easy and computationally useful method for building a small size network.

VII. FUTURE SCOPE

In future EBSA can be trained using learning algorithms other than commonly used quickprop algorithm.

REFERENCES

[1]. Li, M., & Wang, D. (2017). Insights into randomized algorithms for neural networks: Practical issues and common pitfalls. *Information Sciences, 382,* 170-178.

[2]. Muzhou, H., Taohua, L., Yunlei, Y., Hao, Z., Hongjuan, L., Xiugui, Y., & Xinge, L. (2017). A new hybrid constructive neural network method for impacting and its application on tungsten price prediction. *Applied Intelligence, 47*(1), 28-43.

[3]. Lin, S., Zeng, J., & Zhang, X. (2018). Constructive neural network learning. *IEEE transactions on cybernetics, 49*(1), 221-232.

[4]. Huang, G. B., Saratchandran, P., & Sundararajan, N. (2005). A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE transactions on neural networks, 16*(1), 57-67.

[5]. Gutiérrez, P. A., & Hervás-Martínez, C. (2011). Hybrid artificial neural networks: models, algorithms and data. In International work-conference on artificial neural networks (pp. 177-184). Springer, Berlin, Heidelberg.

[6]. Hwang, J. N., You, S. S., Lay, S. R., & Jou, I. C. (1996). The cascade-correlation learning: A projection pursuit learning perspective. *IEEE Transactions on Neural Networks, 7*(2), 278-289.

[7]. Waugh, S. (1994). Dynamic learning algorithms. Department of Computer Science, University of Tasmania.

[8]. Waugh, S. (1994). Extensions to Cascade-Correlation Training. Department of Computer Science, University of Tasmania.

[9]. Liu, D., Chang, T. S., & Zhang, Y. (2001, September). A new learning algorithm for feedforward neural networks. In Proceeding of the 2001 IEEE International Symposium on Intelligent Control (ISIC'01)(Cat. No. 01CH37206) (pp. 39-44). IEEE.

[10]. Duong, T. A., & Stubberud, A. R. (2000). Convergence analysis of cascade error projection-an efficient learning algorithm for hardware implementation. *International journal of neural systems, 10*(03), 199-210.

[11]. Fahlman, S. E. (1991). The recurrent cascade-correlation architecture. In Advances in neural information processing systems (pp. 190-196).

[12]. Ash, T. (1989). Dynamic node creation in backpropagation networks. Connection science, 1(4), 365-375.

[13]. Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In Advances in neural information processing systems (pp. 524-532).

[14]. Leisch, F., & Dimitriadou, E. (2010). Machine Learning Benchmark Problems.

[15]. Han, C. W. (2017). Auto MPG Prediction using Tree Architectures of Fuzzy Neural Networks.

How to cite this article: Kaur, J. and Gupta, N. (2020). Extended Bipolar Sigmoid Algorithm for Enhancing Performance of Constructive Neural Network. *International Journal on Emerging Technologies*, *11*(2): 1034–1038.