International Journal on Emerging Technologies **11**(2): 991-996(2020)

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

Bipolar Sigmoid Algorithm for Designing 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 16 January 2020, Revised 14 March 2020, Accepted 16 March 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Constructive Neural Network is a reliable, fast and efficient technique of constructing neural network for solving difficult problems. Cascade Correlation algorithm is a popular method of constructing artificial neural network. It learns very quickly. Recurrent Cascade Correlation algorithm is a recurrent type of CCRA. It learns fast and does not have to deal with continuous time steps. A constructive neural network Bipolar Sigmoid Algorithm is being proposed. It employs the bipolar sigmoid function as its activation function. The weight freezing and another measure correlation score is employed in this algorithm. It operates every time on only one layer of inputs as other weights are frozen. It is tested on twelve regression functions for mean square error obtained on testing data set and hidden nodes. It develops compact neural network as compared to CCRA and has better generalization characteristics where hidden nodes have less chances of saturation.

Keywords: Constructive Neural Network, Cascade Correlation (CCRA), Recurrent Cascade Correlation, Bipolar sigmoid function.

Abbreviations: CCRA, Cascade Correlation; RCCA, Recurrent Cascade Correlation Algorithm; BSA, Bipolar Sigmoid Algorithm.

I. INTRODUCTION

Constructive Neural Network is a fast and very reliable technique of constructing a neural network. It is very efficient in solving problems of optimization, regression, classification and pattern recognition with ability of enormous parallelism [1]. It is grown according to the problem in hand to appropriate size with no need of guessing the number of layers in advance using trial and error method [2]. A network of size appropriate to the particular problem is generated during finding the solution of the problem.

Cascaded networks have higher advantage as the layers of hidden nodes responsible for processing are also easily adaptable to problem in hand. These types of networks have advantage of solving certain difficult problems over other known feed forward layered neural networks [3]. These types of networks automatically specify the size and topology of the network being used. Cascade Correlation Algorithms advantage is that most of the network is frozen and only one layer is trained at any given time. It reduces the herd effect in which the candidate hidden nodes are allowed to reply to the error. Candidate nodes which are in pool can learn the error without any interaction between the other nodes. It has been a very successful algorithm. Small networks do not have severe overfitting problem. Progress of training can be taken into consideration as an extra factor for determining the stoppage condition [4].

In the earlier work on Cascade Correlation Algorithm it has been written that different non-linear activation functions can be used at hidden layer [5]. There can be sigmoid activation function, radial activation function, Gaussian activation functions and many more. It may result in more sophisticated and compact solutions to problems as compared to homogeneous network. Bipolar Sigmoid Algorithm (BSA) provides with the opportunity to use improved bipolar activation function for building a hierarchal cascaded network structure. Explanation is given in the following sections starting with some cascade algorithms followed by the new algorithm.

II. CASCADED ALGORITHMS

There are a number of Cascaded algorithms like Cascaded Correlation algorithm, Recurrent Cascaded Algorithm, Cascaded Error Projection, Casper algorithm and many more. Two of them are discussed below.

A. Cascade Correlation Algorithm (CCRA)

Cascade Correlation algorithm is a supervised type of learning algorithm used for developing artificial neural networks. This algorithm is very popular as it determines the network size and network topology on its own. There is no need to decide in advance the size, topology and depth of the network. It is very quick in learning. There is no back propogation through the network connections of the error signal. Even if the set used for training changes, its structures remains the same. This algorithm was designed to overcome the problems and limitations of back propogation learning algorithm (step size problem and moving target problem). The network consists of only input units, connections and output units initially. It allows only one hidden node to evolve while holding the neural network constant at any given time. There is no requirement of deciding the size of network,

depth of network and connectivity pattern in advance. Automatically a small neural network is built with a mixture of nonlinear activation functions like sigmoid, Gaussian, Radial and many more. It results in compact and elegant problem solutions with a mix of unit types that can adapted [6]. It learns very fast as each hidden unit gets a fixed problem and can decisively solve the problem. It builds deep neural networks which are feature detector of high order. In this new information is being added to network which has been already trained known as incremental learning. Only weights of one layer are trained at any given time, the rest network does not change and the results are cached. The results are not required to propagate backward the error signals and the connections send signal in one direction. There is no interaction between the candidate hidden units and gets the same input and error signal. It creates the hidden units dynamically. The units are stacked in multiple layers, at any given time only weights of one layer are trained and are trained using hill climbing. The units are freezed after being added to the network. This algorithm is called greedy algorithm as each hidden unit try to garb most of the error information.

B. Recurrent Cascade Correlation Algorithm (RCCA)

Recurrent Cascade Correlation is Cascade Correlation algorithms recurrent type. The hidden units are added the neural network with recurrent type connections one at any time. It learns from examples in order to map the inputs to the desired outputs. There are connections between previous hidden units outputs and the hidden units in the layer. The output from new hidden unit is added back to the existing unit as new input. The network forms a recurrent loop [7]. The candidate unit with self recurrent weighted input is trained with other input weights in order to maximize the candidate correlation with residual error. It uses only self recurrent connections. The hidden unit will work as flip flop if recurrent connection is positive and will be in previous state until forced by other inputs to change. The hidden unit oscillates between negative and positive for each time step until held by other inputs in place if the recurrent connection is negative. The hidden unit will work like some kind of gate if the recurrent connection weight is zero. When new hidden node (candidate) is added to the network weight of the self recurrent unit is frozen. The other weights are also frozen along with the weight of new hidden node. It learns fast, has good generalization ability, selects the network topology automatically, creates complex feature detector of high order and learns incrementally. It does not deal with continuous time steps [8].

III. DESIGN OF CONSTRUCTIVE BIPOLAR SIGMOID ALGORITHM (BSA)

Constructive algorithm is based on cascade form of architecture. It is a supervised learning algorithm which builds a multilayer neural network. This constructive algorithm is based on bipolar sigmoid activation function which creates incremental cascaded neural network architecture. At each stage of network construction this algorithm uses bipolar sigmoid function. In this algorithm there is no need to decide the network size in advance and it builds automatically a network which is reasonably small. The algorithm begins with training the output node to approximate the target output. The network starts with some input nodes, no hidden nodes and some output nodes. The nodes in input layer and output layer are decided by the problem in hand. The input node is connected to output node with a connection weight that can be adjusted. The bias input node has a value of 1. The input connections and output connections are trained directly on the training set. quickprop algorithm is used for training the output weights. This algorithm acts as delta rule, for no hidden nodes. It converges very fast. The output nodes output is linear sum of weighted input and bias. The hidden nodes are added to the network one at a time. The hidden node is connected to the input nodes and already existing hidden nodes. The input weights of the hidden node are frozen when it is added to the network and the output connections are repeatedly trained. Each new hidden node is added to a new layer in the network. After some training cycles there is no major reduction in the error. The network is run last time to calculate the error on the entire set of training. Training will be stopped if the network gives satisfactory results. If the result is not satisfactory training is done to reduce the residual error. This is done by addition of new hidden node, which is done by algorithm (unit creation). The new hidden node is created using a candidate node which connected to the external inputs of the network and connections of already existing hidden nodes. It is not connected to the network active at present. The weights of candidate node are adjusted after each pass for a set number of passes of training set. A pool of candidate nodes is trained with each node having a different initial weight selected randomly. Each node is given same input and error signal. The candidate nodes have no interaction with each other and all of them are parallelly trained. The candidate node with best correlation score is installed when there is no progress further. It speeds up the process of training and there are fewer chances of not useful nodes being installed. When candidate node is trained there are no changes in weights of the network active at that time. Bipolar sigmoid activation function is used by hidden and candidate nodes. The hidden node works a feature detector and it is not altered once built. There are two phases of training namely training of weights of hidden nodes and output nodes [9].

Let us assume the training pattern is k, y_{ok} is the observed output, t_{ok} is the desired output, f' is the bipolar sigmoid activation functions derivative, I_{ik} is the input node or hidden node, W_{oi} is the weight of input node i to output node o, candidate node output is y,$\bar{y}$ is the average of candidate node output, residual error is e, $\bar{e}$ is the average of residual error, Corr is correlation, s_o is the sign of the Corr, ΔW is the weight change, α is the learning rate and μ is the momentum factor. Input nodes are connected to output nodes with connections. There are no hidden nodes initially. This network is trained for minimization of the error using quickprop algorithm. When output node weights are trained all the other weights are frozen. The output nodes are trained for minimization of sum squared error of the network. The performance of the network is evaluated once there is no improvement in the error level.

$$SE = \frac{1}{2}\sum_{o,k} p^2 \quad (1)$$

Where $p = (t_{ok} - y_{ok})$ is the difference between desired output and observed output.
This sum squared error is minimized using gradient descent by calculation the error term. The gradient of SE vector is partial derivative of SE with respect to weights. By adjusting the weights the error can be rapidly reduced.

$$\frac{\partial SE}{\partial W_{oi}} = \sum_k e_{ok} I_{ik} \quad (2)$$

$$e_{ok} = pf' \quad (3)$$

Improved bipolar sigmoid activation is used in this algorithm. This activation function has been designed for minimizing error by selecting a representation which maintains the weights of the network in the range of +1 and -1.

$$f(a) = \frac{1-e^{-2a}}{1+e^{-2a}} \quad (4)$$

This activation function is graphically represented as follows.

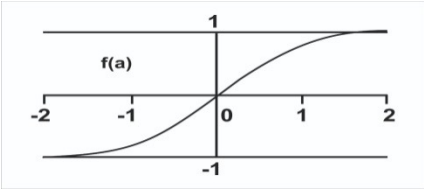

Fig. 1. Bipolar Sigmoid Function.

New hidden node is added to the network to reduce the error. It starts with a number of candidate nodes. Every candidate node is connected to all the external input connections of the network and to already existing hidden nodes. Its output is not connected to the network active at present. Candidate node output is not connected to the network active at present. A number of run on the training set examples are conducted in which the weights of the candidate node are adjusted after every pass. Every candidate node independently tries to maximize correlation using hill climbing. The main aim of these adjustments is to maximize the correlation of output of candidate node y and the error at outputs known as the residual error e.

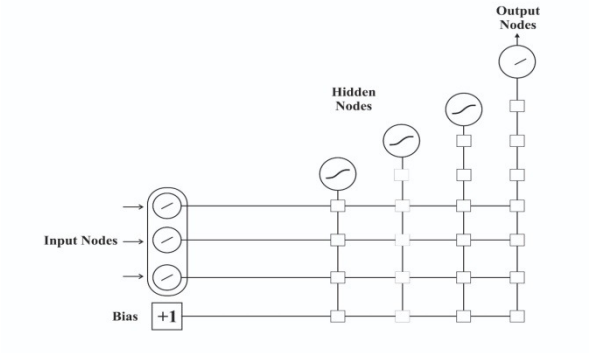

Fig. 2. Bipolar Sigmoid Neural Network.

Candidate hidden nodes are trained to maximize the correlation between the candidate node output and residual error at the output of the network. Corr is calculated as

$$Corr = \sum_o |\sum_k (y_k - \bar{y})(e_{ok} - \bar{e})| \quad (5)$$

This Corr is computed for all the training pattern k.

$$\delta_k = \sum_o s_o (e_{ok} - \bar{e}) f' \quad (6)$$

Gradient ascent is used for maximizing the Corr.

$$\frac{\delta Corr}{\delta W_i} = \sum_k \delta_k I_{ik} \quad (7)$$

Every candidate node in the pool begins with a different pattern of initial weights. Each candidate node tries for maximization of its Corr value independently. Gradient descent or ascent is used to minimize sum squared error and maximize correlation with the above computed values. Weights are updated using $W(r+1) = W(r) + \Delta W(r)$.
α is the learning rate which controls the step size and μ is the momentum factor ranges between 0 and 1. Learning rate is used to avoid disruption in the direction of learning, when an unusual training pair is presented from the pattern. To increase the convergence rate a momentum factor can be added and to add momentum previous one or more training weights need to be saved. Momentum helps in large adjustment of weights and accelerates the convergence of the error propagation algorithm. The network proceeds in the direction of the previous and current(combined)gradient. $W(r+1) = W(r) + \alpha\delta_k I_{ik} + \mu\Delta W(r-1)$
The stoppage criteria in based on the performance of the network during training. The training can be terminated if there is no decrease in training error after addition of a number of certain hidden nodes or the error is below a certain value. Another way of stopping training is terminate training after all the data used for training have been considered by the procedure and have been seen by the problem [10]. The other stoppage condition can be number of epochs, loss of generalization. If the algorithm uses these conditions then the generalization performance of the network may be biased. These stoppage methods are simple and the error can observed directly while training. This algorithm is sensitive to stoppage conditions. The parts of the network will not give good enough results, if training is for a short period and if training is for a long period, it will cost more computation cost and time, and can result in poor generalization and overfitting [11, 12].
This algorithm requires less number of hidden layers in the network substantially by using improved bipolar sigmoid activation function. It does not affect the ability of network for generalization. This algorithm can work as an effective and simple method for developing modest depth network. In many important applications where time of forward propagation is important, depth of network needs to be minimized. In difficult problems of generalization the algorithm makes the convergence of network difficult and the hidden nodes may explode.

IV. SIMULATIONS AND RESULTS

In order to investigate the effects of Bipolar Sigmoid Algorithm, Twelve benchmark problems are selected. Cascade algorithm CCRA is compared with Bipolar Sigmoid Algorithm BSA. Methodology for comparing the performance generalization for the two functions is as follows. The data is partitioned into training set, validation set and testing set. For each of the activation functions twenty training runs are being performed with different starting random initial weight values. Training was done till ten hidden nodes. After installation of every hidden node the sum of squared error is measured for both the algorithms and score of hidden nodes is also reported in the tables of each regression function. In

each table the minimum, maximum, mean and standard deviation of mean square error on testing data set and hidden nodes is tabled.

Let us consider one dimensional regression functions [13]. For each of the function 1500 uniformly distributed points at random were generated in one dimensional space.

1. $v = 1 + \sin(u)$ (8)

where value of v lies between -2π and 2π

Table 1: Results of One dimensional regression function 1.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.20472	0.29186	1	1
Maximum	0.30239	0.2935	10	10
Mean	0.27347	0.2926	6.5	7.8
Std.deviation	0.03284	0.00059	2.6352	2.9364

From the table above it can be inferred that mean square error of BSA is less as compared to CCRA. Both BSA and CCRA have same number of minimum and maximum hidden units.

2. $v = 3 \sin u(u+1) + \cos u^2 + 2$ (9)

where u lies between $-\pi$ and π.

Table 2: Results of One dimensional regression function 2.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.2053	0.29227	1	8
Maximum	0.30844	0.29351	8	10
Mean	0.27934	0.29271	4.9	9.1
Std.deviation	0.030189	0.00045	2.4244	0.9944

From the above table it can be noted that for the above one dimensional function the new algorithm BSA has less mean square error as compared to the original CCRA. BSA requires less number of hidden units than CCRA.

3. $v = \sin(u)/u$ (10)

where u lies between 0 and 2π.

Table 3: Results of One dimensional regression function 3.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.27743	0.29196	1	2
Maximum	0.31081	0.2931	10	10
Mean	0.2939	0.2926	6.4	7.9
Std.deviation	0.01165	0.00037	3.2728	2.5582

From the above table it can be seen that BSA performs better than CCRA and requires nearly same number of hidden units.

Let us consider two dimensional regression functions for testing the algorithm [14-16]. For each problem two hundred twenty five random inputs which are uniformly distributed are generated in interval of 0 to 1 for training of network in two dimensional spaces. Uniformly sampled ten thousand points are used for testing the trained network generalization performance with maximum ten hidden nodes for maximum three hundred epochs. Five regression problems of varying complexity are described as follows.

4. Radial Function

$v = 24.234\ ((u_1 - 0.5)^2 + (u_2 - 0.5)^2)\ (0.75 - (u_1 - 0.5)^2 - (u_2 - 0.5)^2)$ (11)

where u lies between 0 and 1.

Table 4: Results of Two dimensional regression function 4.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.26486	0.28221	2	8
Maximum	0.31077	0.29298	10	10
Mean	0.28709	0.28897	6.8	9.5
Std.deviation	0.01509	0.00349	2.7809	0.8498

From the above table it can be seen BSA has less mean square error as compared to CCRA. BSA requires less number of hidden nodes as compared to CCRA which needs 9.5 hidden nodes on an average.

5. Complicated Interaction Function

$v = 1.9\ (1.35 + e^{u1} \sin(13(u_1 - 0.6)^2)\ e^{-u2} \sin(7u_2))$ (12)

where u lies between 0 and 1.

Table 5: Results of Two dimensional regression function 5.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.27743	0.28282	1	2
Maximum	0.31081	0.29351	8	10
Mean	0.2939	0.28872	4.6	7.9
Std.deviation	0.11647	0.00351	2.8363	2.5582

From the above table it can be seen BSA is performing better than CCRA in terms of mean square error and requires slightly less hidden units than CCRA.

6. Harmonic Function

$v = 42.659\ (0.1 + (u_1 - 0.5)\ (0.05 + (u_1 - 0.5)^4 - 10(u_1 - 0.5)^2(u_2 - 0.5)^2 + 5(u_2 - 0.5)^4))$ (13)

where u_1 and u_2 lies between -0.5 and 0.5

Table 6: Results of Two dimensional regression function 6.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.26486	0.29186	1	8
Maximum	0.31077	0.2935	10	10
Mean	0.28709	0.2926	6.5	9.5
Std.deviation	0.01509	0.00059	2.6352	0.8498

From the above table it can be seen that BSA performs better than CCRA and needs less hidden units as compared to CCRA.

7. Additive Function

$v = 1.3356\ [1.5\ (1 - u_1) + e^{(2u_1 - 1)} \sin(3\pi\ (u_1 - 0.6)^2) + e^{(3(u_2 - 0.5))} \sin(4\pi\ (u_2 - 0.9)^2)\ ]$ (14)

where u_1 and u_2 lies between 0 and 1.

Table 7: Results of Two dimensional regression function 7.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.2053	0.29186	1	5
Maximum	0.30844	0.2935	10	10
Mean	0.27974	0.2926	6.5	8.8
Std.deviation	0.03036	0.00059	2.6352	1.6193

From the above table it can be seen that mean square error of BSA is less than CCRA and BSA requires four less hidden units than CCRA.

8. Simple Interaction Function

$$v = 10.391((u1 - 0.4)(u2 - 0.6) + 0.36) \quad (15)$$

where u1 and u2 lies between 0 and 1.

Table 8: Results of Two dimensional regression function 8.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.25249	0.29206	1	4
Maximum	0.30266	0.29348	10	10
Mean	0.28584	0.29282	6	8.5
Std.deviation	0.01444	0.00043	2.9814	2.3214

From the above table it can be seen that BSA performs better than CCRA in terms of mean square error. Also, requires six hidden units on average as compared to 8.5 on average for CCRA.

Let us consider three dimensional regression function [17, 18]. Random 1000 input samples are used for training the network and 500 samples are used for verification of performance of network in terms of generalization.

9. Simple Analytical Function (SAF)

$$v = \frac{1}{1+e^{-e^{u_1}+(u_2-0.5)^2+3\sin(\pi u_3)}} \quad (16)$$

Table 9: Results of Three dimensional regression function 9.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.22307	0.29172	1	6
Maximum	0.31086	0.29378	9	10
Mean	0.27922	0.2957	5.1	8.3
Std.deviation	0.0328	0.00069	3.0714	1.6364

From the table above the training error in BSA becomes much less than CCRA and performance in terms of generalization is improved greatly. BSA requires 5.1 hidden units on average as compared to CCRA which requires 8.3 hidden units on an average.

10. $v=4(u_1 - 0.5)(u_2 - 0.5)\sin(2\pi\sqrt{u_2^2 + u_3^2}))$ (17)

where u_1, u_2 and u_3 lies between -1 and 1.

Table 10: Results of Three dimensional regression function 10.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.24214	0.29199	1	7
Maximum	0.31106	0.29323	10	10
Mean	0.28073	0.29271	4.8	9
Std.deviation	0.01829	0.00041	2.6998	1.4142

From the above table it can be seen that BSA performs better than CCRA. BSA requires 4.8 hidden units on an average and CCRA requires 9 hidden units on an average.

Let us consider four dimensional regression functions. Uniformly distributed 2000 random input samples are generated in the defined domain of four dimensional space. The samples are used as 500 for training, 500 for verification and 1000 for testing. Ten hidden nodes are added at the maximum.

11. $v = \sin(e^{(2u_1 \sin(\pi u_4))}\ e^{(2u_2 \sin(\pi u_3))})$ (18)

where u_1, u_2 , u_3 and u_4 lies between -1 and 1.

Table 11: Result of Four dimensional regression function 11.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.28259	0.29223	1	2
Maximum	0.3124	0.29368	6	10
Mean	0.2943	0.2929	3	7.2
Std.deviation	0.1134	0.00046	1.3333	2.8597

From the above table it can be seen that BSA performs slightly better than CCRA and needs less hidden units as compared to CCRA.

12. $v = e^{(2u1 \sin(\pi u4))} + \sin(u_2 u_3)$ (19)

where u_1, u_2 , u_3 and u_4 lies between -0.25 and 0.25.

Table 12: Results of Four dimensional regression function 12.

	Mean Square Error		Hidden Units	
	BSA	CCRA	BSA	CCRA
Minimum	0.26056	0.29205	1	6
Maximum	0.29678	0.29336	9	10
Mean	0.28248	0.29261	5.3	8.8
Std.deviation	0.01407	0.0005	2.5841	1.7512

From the above table it can be seen that BSA performs slightly better than CCRA and requires less number of hidden units.

So, the results of the comparison of BSA and CCRA highlight that weight freezing technique used and activation function used for building the network has ability to solve many complex problems. The cascaded hidden nodes become feature detectors of higher order. The BSA performs better on regression dataset and produces good solution of problem on the dataset. The BSA requires less number of hidden units as compared to CCRA and reduces the depth of the network. The BSA continues to maintain good mean square error during training of the network. Overall the new algorithm converges better. The period increases immensely over which the neural network is being trained as the size of network is growing.

V. CONCLUSION

Neural networks must address the problems like performance of the network while training on new data set or new training pattern in terms of generalization performance, training patterns rate of recognition. Solutions of these problems influences the neural network overall performance significantly. The proposed algorithm BSA prevents overtraining of the network and gives improved performance in terms of generalization capabilities. BSA produces compact neural networks as compared to CCRA. This algorithm reduces the number of hidden layers in the network substantially by using improved bipolar sigmoid activation function. The depth of the network is reduced by 2-6 layers. The BSA reduces the depth more than CCRA. The algorithm is able to cache its calculation due to its weight freezing technique and there is no error backpropogation. The algorithm is very efficient due to freezing as it caches

the values and no recalculation is done. It also removes the saturation problem and avoids installation of nodes performing poorly due to weight freezing

VI. FUTURE SCOPE

This algorithm BSA can be improved using different activation functions.

REFERENCES

[1]. Lachhwani, K. (2020). Application of neural network models for mathematical programming problems: a state of art review. *Archives of Computational Methods in Engineering, 27*(1), 171-182.

[2]. Zemouri, R., Omri, N., F naiech, F., Zerhouni, N., & Fnaiech, N. (2019). A new growing pruning deep learning neural network algorithm (GP-DLNN). Neural Computing and Applications, 1-17.

[3]. Lee, S. W., & Kim, S. Y. (1999). Integrated segmentation and recognition of handwritten numerals with cascade neural network. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), *29*(2), 285-290.

[4]. Treadgold, N. K., & Gedeon, T. D. (1997, June). A cascade network algorithm employing progressive RPROP. In International Work-Conference on Artificial Neural Networks (pp. 733-742). Springer, Berlin, Heidelberg.

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

[6]. Hoehfeld, M., & Fahlman, S. E. (1991). Learning with limited numerical precision using the cascade-correlation algorithm (pp. 602-611). Carnegie-Mellon University. Department of Computer Science.

[7]. Hang, R., Liu, Q., Hong, D., & Ghamisi, P. (2019). Cascaded recurrent neural networks for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, *57*(8), 5384-5394.

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

[9]. Sibi, P., Jones, S. A., & Siddarth, P. (2013). Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, *47*(3), 1264-1268.

[10]. Castellano, G., & Fanelli, A. M. (2000). Variable selection using neural-network models. *Neurocomputing, 31*(1-4), 1-13.

[11]. Green II, R. C., Wang, L., & Alam, M. (2012). Training neural networks using central force optimization and particle swarm optimization: insights and comparisons. *Expert Systems with Applications*, *39*(1), 555-563.

[12]. Lawrence, S., Giles, C. L., & Tsoi, A. C. (1998). What size neural network gives optimal generalization? Convergence properties of backpropagation.

[13]. Rosen, B. E. (1996). Ensemble learning using decorrelated neural networks. *Connection science*, *8*(3-4), 373-384.

[14]. Ma, L., & Khorasani, K. (2004). New training strategies for constructive neural networks with application to regression problems. *Neural networks, 17*(4), 589-609.

[15]. Kwok, T. Y., & Yeung, D. Y. (1997). Objective functions for training new hidden units in constructive neural networks. *IEEE Transactions on neural networks, 8*(5), 1131-1148.

[16]. Treadgold, N. K., & Gedeon, T. D. (1997, June). A cascade network algorithm employing progressive RPROP. In International Work-Conference on Artificial Neural Networks (pp. 733-742). Springer, Berlin, Heidelberg.

[17]. Ma, L., & Khorasani, K. (2005). Constructive feedforward neural networks using Hermite polynomial activation functions. IEEE Transactions on Neural Networks, *16*(4), 821-833.

[18]. Ma, L., & Khorasani, K. (2003). A new strategy for adaptively constructing multilayer feedforward neural networks. Neurocomputing, *51*, 361-385.

How to cite this article: Kaur, J. and Gupta, N. (2020). Bipolar Sigmoid Algorithm for Designing Constructive Neural Network. *International Journal on Emerging Technologies*, *11*(2): 991–996.