STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720 (Online)
Volume 5, Issue 1
www.stmjournals.com

A Novel Strategy for Weight Initialization in Sigmoidal Feed-forward Artificial Neural Networks

Rajashree Chaurasia*
Department of Computer Engineering, Guru Nanak Dev Institute of Technology, Delhi, India

Abstract

Among the various parameters of a neural network, the initial values of the weights and biases pertaining to each artificial neuron have been given a lot of importance. In this paper, a novel method of weight initialization is proposed. The proposed method of weight initialization distributes the initial weights and thresholds in such a manner that they lie in different regions of the activation function used at the hidden layer. The proposed method is compared with six other popular weight initialization methods on ten function approximation problems using the RPROP (Resilient Back-propagation) and Levenberg-Marquardt algorithms for training. Two types of activation functions viz. tan hyperbolic and logarithmic sigmoidal functions are used for analysis and comparison.

Keywords: *Neural networks, sigmoidal activation, weight initialization, feed-forward artificial neural networks*

****Author for Correspondence*** E-mail: rajashree.chaurasia@gmail.com

INTRODUCTION

Neural networks have been applied to varied application areas since their inception. It is due to their ability to emulate the human brain (where only basic functions of the astoundingly complex human brain have been able to be modeled up to the present) that neural networks are being used extensively in all fields involving any kind of cognition and learning.

Neural networks are able to solve complex high-dimensional problems by emulating the data processing model of the biological nervous system viz. learning by experience, recall and generalization [1]. The learning process is accomplished by means of a learning algorithm, of which, the error back-propagation algorithm is the most widely used.

Among the various parameters in a neural network, the weights and biases pertaining to each artificial neuron have been given a lot of importance. Essentially, the initial values of the weights and thresholds have shown to influence the training of the neural network [2–7]. The most common method for initialization of these parameters is uniform random weight initialization. Many modifications have been proposed in the past to improve and design weight initialization mechanisms based on the computation structure of the network and the dynamics of weight evolution [3–9]. The varieties of weight initialization methods proposed by numerous researchers have been compared mainly in two contexts: whether the network reaches deeper error minima during training and whether the network thus trained has a better generalization capability on applying a specific method.

The training of a neural network involves choosing suitable network architecture, the choice of activation function, training algorithm and weight initialization methodology.

These choices are detailed in the next part of this paper. Also, a new method of weight initialization is proposed. The weight initialization method proposed is demonstrated to be equivalent to if not better than some of the weight initialization methods considered for comparison in this work.

This paper is organized as follows: Next part of the paper describes the design of experiments including the proposed method. Results of the experiments are presented and discussed afterwards, while conclusions and future work are presented in the end.

EXPERIMENTAL DESIGN

Universal approximation theorem states that "the standard multilayer feed-forward network with a single hidden layer, which contains finite number of hidden neurons, is a universal approximator among continuous functions, under mild assumptions on the activation function" [10]. Neural networks follow this principle diligently. The design of a neural network for a function approximation task involves deciding the architecture of the network, activation functions to be used at the hidden and output layers, the training algorithm for learning, function approximation tasks, data sets and weight initialization mechanisms considered for analysis.

Architecture

Deciding the architecture of a feed-forward neural network mainly revolves around the choice of the number of nodes at the hidden layer and the number of layers, the number of inputs and outputs in the network. According to the Universal Approximation theorem, one hidden layer is sufficient to approximate any continuous function for sigmoidal networks [10]. Thus, one hidden layer has been used in the network.

The number of inputs in the input layer and the number of outputs in the output layer are decided by the number of independent and dependent variables, respectively. Furthermore, a vector function or a function with more than one dependent variable may be treated as a set of functions with the same set of independent variables. Therefore, only functions with a dependent variable will be considered, thereby implying that the networks used have only one node in the output layer.

Deciding on the number of hidden layer nodes is done via exploratory experiments wherein the number of hidden nodes is varied and a network trained for a limited number of epochs. The number of nodes with a satisfactory convergence is taken as the size of the hidden layer.

Activation Function

Hidden node output functions are required to be sigmoidal in nature [11]. The generally used activation functions are the anti-symmetric hyperbolic tangent functions:

$$\varphi(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

and the asymmetric1 logistic or the log-sigmoid function:

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

Preference is made for the anti-symmetric function, thus, the activation function used at the hidden nodes is the hyperbolic tangent function [8].

Training Algorithm

The algorithm used for adjusting the weights, so as to minimize the error from the network are based on local non-linear optimization methods, though global optimization techniques have also been utilized for the training of sigmoidal FFANNs. The local non-linear optimization methods based training algorithms range from gradient descent based algorithms like the standard back-propagation algorithm to second order algorithms using the Hessian. In this experiment, resilient back-propagation algorithm (RPROP) is being used. RPROP is a fast convergent first order method that has lesser memory requirements than the second order methods.

Weight Initialization

The connection between the nodes and the thresholds of the nodes are collectively known as the weights of the network. Usually, the weights are initialized to small random values in the range $[-\lambda, \lambda]$, where $\lambda > 0$ is a small value in the range (0, 1]. In the experiments for random weight initialization, $\lambda = 0.5$. Random weight initialization is done to break the weight symmetry during training. If the weights are initialized to equal values, they move and evolve during training in tandem. The RPROP algorithm works in batch mode, that is, the weights are updated based on the error values corresponding to the complete error over the training set. If the p^{th} output desired is tp, and the output obtained from the network is yp for the p^{th} input set, and if there are P number of training exemplars, then the error is measured

by the mean squared error (MSE), defined as follows:

$$\text{MSE} = \frac{1}{p}\sum_{p=1}^{P}\left(t_p - y_p\right)2$$

The schematic diagram representing a one hidden layer network is shown in Fig. 1.

The number of input nodes are N, the number of hidden nodes is M, the weights between the hidden nodes and the inputs are labeled ω_{ij} i.e. the weight between the i^{th} hidden node and the j^{th} input node, the threshold of the i^{th} hidden node is θ_i. For any one input tuple, the net input to the i^{th} hidden node is:

$$\text{net}_i = \sum_{j=1}^{N} \omega ij Xj + \theta_i$$

The output from the node is $h_i(x_1,\ldots,x_N) = \tanh(\text{net}_i)$. The weight between the i_{th} hidden node and the output node is represented by α_i, where i ϵ 1, 2, …, M and the threshold of the output node is γ. Then the output from the network is:

$$y = \sum_{m=1}^{M} \alpha_m h_m + \gamma$$

Therefore, the network output function uses a linear node. The experiments were carried out using MATLAB version R2011a partly on a 32-bit Intel Core 2 Duo based Microsoft Windows XP system with 1 GB RAM and partly on a 32-bit Intel i7 based Microsoft Windows 7 Professional system with 2 GB RAM.

Function Approximation Tasks

The following 10 function approximation tasks are used for the experiments:

1. One dimensional input function taken from the MATLAB sample file humps.m:

$$f_1(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6$$

where x ϵ [0,1]. See Fig. 2a for a plot of this function.

2. Two dimensional input function taken from MATLAB sample file peaks.m:

$$f_2(x_1,x_2) = 3(1-x_1)^2\, e^{(-x_1^2-(x_1+1)2)} - 10(x1/5 - x_1^3 - x_2^5)\, e^{(-x_1^2-x_2^2)} - (1/3)\, e^{(-(x_1+1)2-x_2^2)}$$

where x_1 and x_2 ϵ [-3,3]. See Fig. 2b for a plot of this function.

3. Two dimensional input function from [11], [12], [13]:

$$f_3(x_1,x_2) = \sin(x_1,x_2)$$

where x_1 and x_2 ϵ [-2,2]. See Fig. 2c for a plot of this function.

4. Two dimensional input function from [11], [12], [13]:

$$f_4(x_1,x_2) = e^{\left(x_1 \sin(\pi x_2)\right)}$$

where x_1 and x_2 ϵ [-1,1]. See Fig. 2d for a plot of this function.

5. Two dimensional input function from [12], [13], [14]:

$$f_5(x_1,x_2) = 1.3356\Big(1.5(1-x_1) + e^{2x_1-1} sin(3\pi(x_1-0.6)^2) + e^{3(x_2-0.5)} sin\big(4\pi(x_2 - 0.9)\big)^2\Big)$$

where x_1 and x_2 ϵ [0,1]. See Fig. 3a for a plot of this function.

6. Two dimensional input function from [12], [13], [14]:

$$f_6(x_1,x_2) = 1.9\Big(1.35 + e^{x_1}\, sin\big(13(x_1 - 0.6)^2 e^{-x_2}\, sin(7x_2)\big)\Big)$$

where x_1 and x_2 ϵ [0,1]. See Fig. 3b for a plot of this function.

7. Two dimensional input function from [12], [13], [14]:

$$f_7(x_1,x_2) = 42.659\Big(0.1 + x_1\big(0.05 + x_1^4 - 10x_1^2x_2^2 + 5x_2^4\big)\Big)$$

where x_1 and x_2 ϵ [-0.5,0.5]. See Fig. 3c for a plot of this function.

8. Two dimensional input function from [13], [14]:

$$f_8(x_1,x_2) = \frac{1 + sin(2x_1 + 3x_2)}{3.5 + sin(x_1 - x_2)}$$

where x_1 and x_2 ϵ [-2,2]. See Fig. 3d for a plot of this function.

9. Four dimensional input function from [14]:

$$f_9(x_1,x_2,x_3,x_4) = 4(x_1 - 0.5)(x_4 - 0.5)\, sin\left(2\pi\sqrt{x_1^2 + x_2^2}\right)$$

where x_1, x_2, x_3, x_4 ϵ [-1,1].

10. Six dimensional input function from [14]:

$$f_{10}(x_1,x_2,x_3,x_4,x_5,x_6) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10\,x_4 + 5\,x_5$$

where x_1, x_2, x_3, x_4, x_5, x_6 ϵ [-1,1].

Data Sets

For each of the function approximation tasks, a set of 2200 input points are generated by uniform random sampling of the input domain of the function and the corresponding outputs are calculated.

In neural network training, three types of data sets are classified:

Training Data Set

This data set is used to adjust the weights of the neural network. From the 2200 input points, 200 are used for training the neural networks.

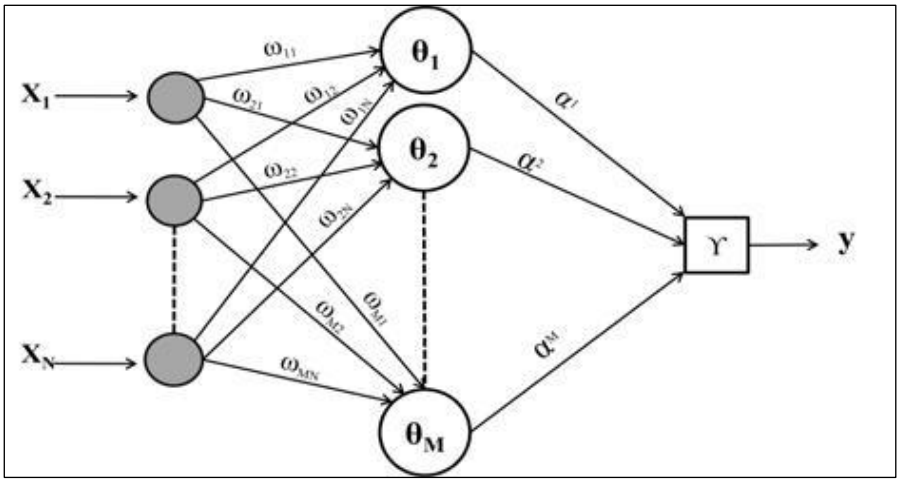

Fig. 1: *The Schematic Diagram of a Single Hidden Layer Network.*

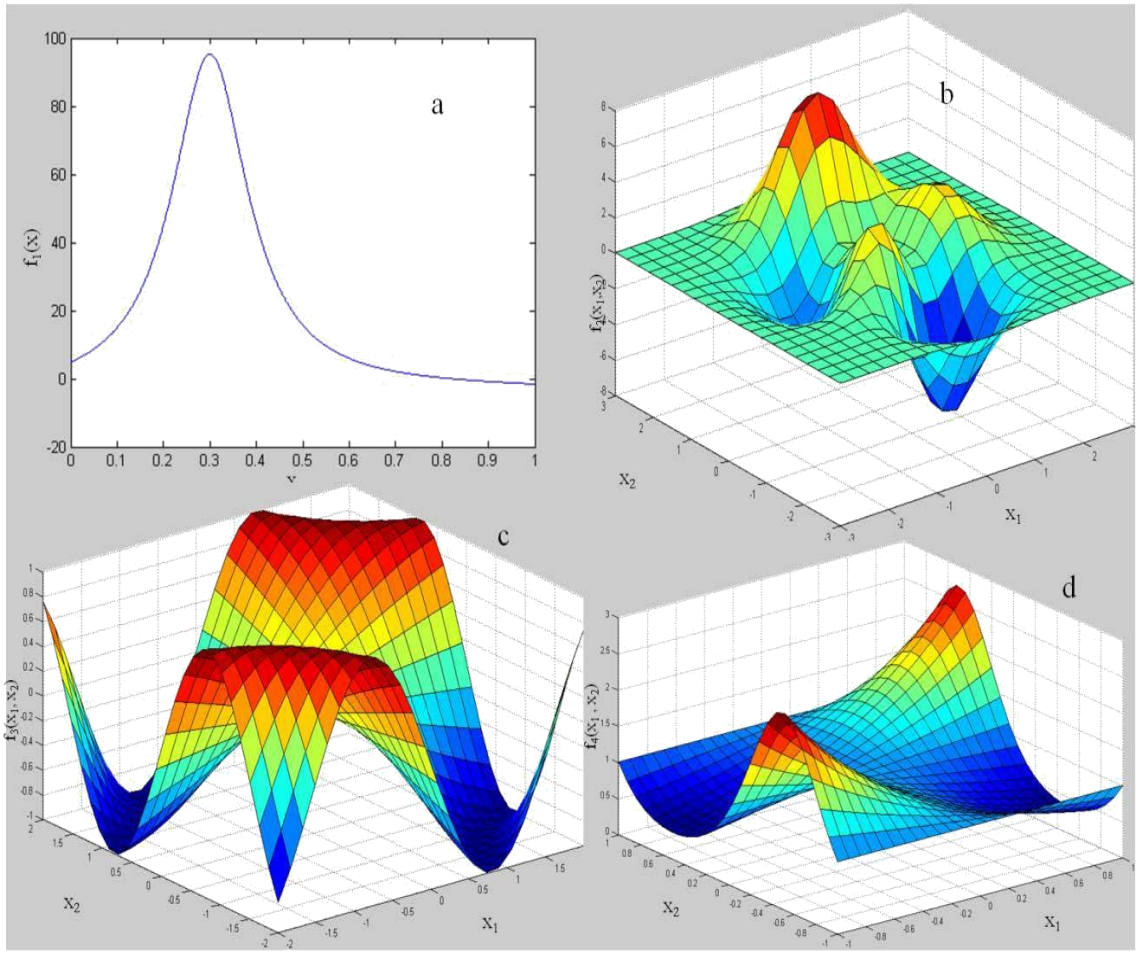

Fig. 2: *Functions f_1 to f_4.*

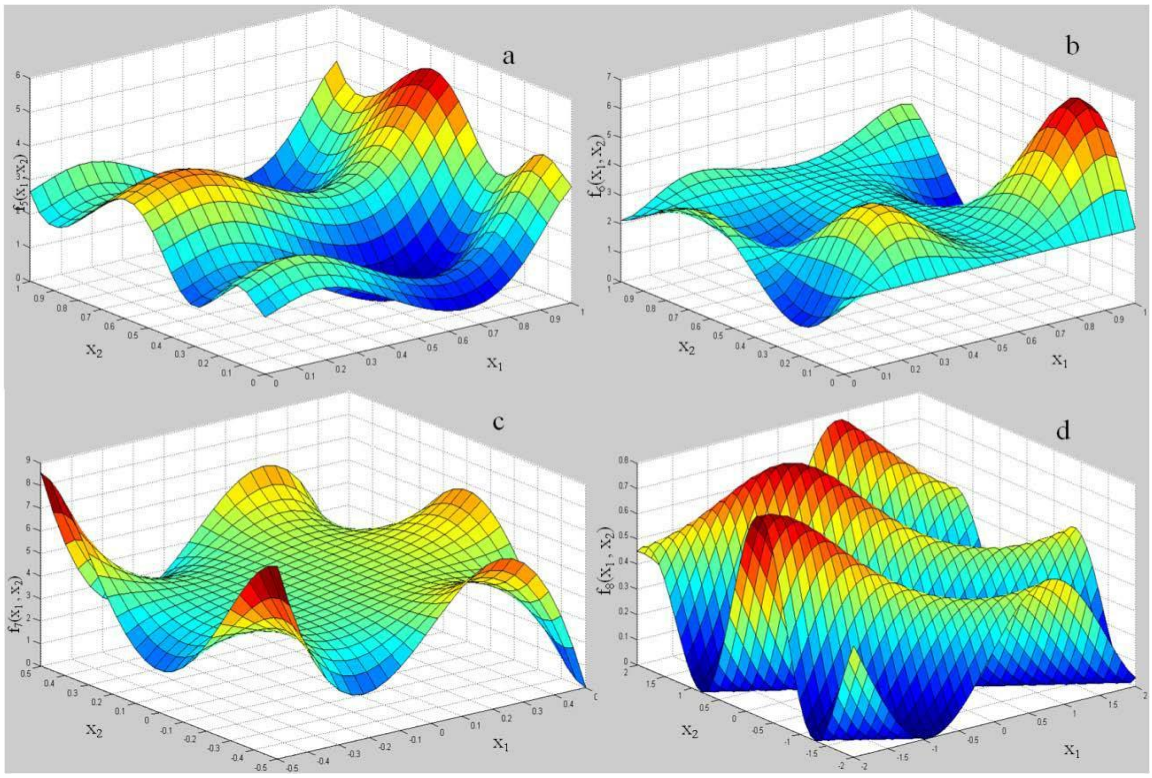

Fig. 3: *Functions f_5 to f_8.*

Validation Data Set

This data set is used to minimize over fitting. Because the validation set is not being us directly to adjust the weights of the network, therefore, a good error for the validation and also the test set indicates that the network predicts well for the training set exemplars. Also, it is expected to perform well when new patterns are presented to the networks which were not used in the training process before. If the accuracy over the training data set increases and the accuracy over then validation data set stay the same or decreases, then over fitting of the neural network is taking place and training should therefore be stopped. In the experiments, out of the 2200 data points, 1000 are used for validation purposes.

Testing Data Set

This data set is used only for testing the final solution in order to confirm the actual predictive power of the network. The remaining 1000 data points are used to test the neural networks.

Normalization of the input and output data is an important criteria in training the neural network. This is achieved by shifting the input and output data such that the mean of the data over the training set is close to or equal to zero. This reduces convergence time and improves the learning of the network.

All the data, i.e. the inputs and the outputs are normalized within the range of [–1, 1] for training and further reporting.

Architecture of SFFANN Used

The architecture of the FFANNs used for each task was obtained by exploratory experiments, where the number of hidden nodes was varied from 2 to 30 in steps of unity to determine the number of nodes required at the hidden layer for the neural network to converge satisfactorily. The architecture thus decided is given in Table 1.

Table 1: *Network Size Summary.*

S. No.	Function	Inputs	Hidden Layer Size	Outputs
1	f1	1	8	1
2	f2	2	15	1
3	f3	2	12	1
4	f4	2	10	1
5	f5	2	10	1
6	f6	2	12	1
7	f7	2	15	1
8	f8	2	25	1
9	f9	4	25	1
10	f10	6	10	1

Weight Initialization Mechanisms

Seven weight initialization methods are being used for all the experiments reported in this manuscript.

Base (Random Weight Initialization)

In this method, 50 sets of weights are initialized randomly from a uniform distribution in the range [–0.5, 0.5].

Bottou

This method initializes the neural network weights for single layer network with one hidden layer, where the activation functions to be used at each of the hidden nodes is the tan hyperbolic function. This routine uniformly initializes the weights from the input to the hidden nodes in the range $[-a/\sqrt{d_{in}}, a/\sqrt{d_{in}}]$ where *a* is the weight variance corresponding to the maximal inflection point of the curve of the activation function and d_{in} is the fan-in of the node [8].

Smieja

This technique uses uniformly distributed weights which are normalized and adjusted to the magnitude $2/\sqrt{d_{in}}$ (d_{in} being the distance between the hyper-planes) for each node [5]. The thresholds of the hidden units are then initialized to a random value in the interval $[-\sqrt{d_{in}}/2, \sqrt{d_{in}}/2]$ and the thresholds of the output nodes are set to zero.

Yam 2001

The weights are uniform random numbers between $-w_{max}$ and w_{max} [7]. The equation for maximum value of the weight is:

$$w_{max} = \frac{8.72}{D_{in}} \sqrt{\frac{3}{N}}$$

SCAWI

This technique makes the weights uniformly distributed over the interval [−a, a], with $a=1.3/(1+E[x_2])-0.5$ (where x is the input) for the input layer and $a=1.3=(1+0.3d_{in})-0.5$ (d_{in} is the fan-in of a neuron) for the output layer, and restricted the number of neurons with absolute activations greater than 0.9 [4].

NW

This method uses a multilayer perceptron with piecewise linear activation functions as an approximation of a network with logistic activation functions. Based on this simplification, it calculates an optimal length of $\sqrt[d_{in}]{N_2}$ (where d_{in} is the fan-in of a node) for the randomly initialized weight vectors and an optimal bias range of $\left[-\sqrt[d_{in}]{N_2}, \sqrt[d_{in}]{N_2}\right]$ for neurons in the hidden layer, where N2 is the number of hidden nodes [3]. The weights of the neurons in the output layer are randomly initialized in the interval [-0.5, 0.5], without any justification given.

RAN 001

In the proposed weight initialization routine, for the input to hidden layer weights, a set of weights equal in number to the number of inputs in the cube of I dimensions with an edge size of S = 2*G/H are generated.

G is the point of maximum inflection of the curve of the derivative of the activation function used, i.e. the hyperbolic tangent function. By definition, the inflection point is a point on a curve where the curvature changes its sign from positive to negative or vice versa. The maximum point of inflection then implies the point on the curve where there is maximum change in the sign of curvature.

In the proposed method, the edge size S is approximately equal to 5% of the peak of the derivative of the hyperbolic tangent function, which is peaked at unity. This is done so that the derivative does not contribute much to the weight updation factor and thus learning can proceed effectively.

It is known that the tan hyperbolic function is approximately linear with a slope of unity when the variable for which it is calculated lies between –1 and 1. Further, the function saturates to unity when that variable becomes large in magnitude. The hidden layer nodes are responsible for internal representations of the function approximation task at hand. Each of these nodes, if made to operate in a separate region of the activation function used, will increase the quality of the learning of the neural network. When each hidden layer node is responsible for a small region of the tan hyperbolic function, we can imply that learning

is being achieved through piece-wise linear approximation.

Likewise, in the proposed method, the weights of the input to hidden nodes are shifted in the interval [–S/2, S/2] from a uniform distribution of random numbers generated in the range [0, 1]. The hidden layer node thresholds are also initialized in the same fashion. Further, a normalization parameter for each hidden layer node is calculated as the length of the weights at that particular hidden node. This parameter is then used to normalize the weights and thresholds feeding into that hidden node. Normalization results in the weights and thresholds of the hidden layer nodes occupying different regions of the activation function and can be thought of as being contained in different sized spheres in the increasing order of sizes. Thus, each of the nodes will respond differently to the input patterns presented during training thereby improving the learning of the network. This is the main principle behind the proposed algorithm. An algorithmic form of the same is given in Algorithm 1.

Algorithm 1: PROPOSED WEIGHT INITIALIZATION ALGORITHM (RAN001)

Input: I = number of inputs, H = number of hidden nodes
Output: IW = input to hidden node weights, HO = hidden to output node weights, HT = hidden node thresholds, OT = output node thresholds

```
//Calculate G as the maximum inflection point of the derivative of the activation function used
at the hidden layer
if (activation function is tanh)
  G = 2.1783; //for tanh
else
  G = 4.3565; //for logsig
end if;
//Calculate S
S = (2*G)/H;
//Generate a uniform random number array with values in the range (-S/2, S/2) of size
HWT=(I+1)H and assign it to array W
MAX = (I + 1)*H;
//Uniform random numbers between (0, 1) are generated using the RAND function of the array size
specified
W = RAND (1: MAX);
//Shift by S/2 to get the weights in the interval [-S/2, S/2]
W = W*S – S/2;
//Initialize the input weights and thresholds of the hidden nodes
for i = 1 to H do
  //The index in W from where the assignment of the i^th hidden node input weights are to start
  K = (i – 1)*(I + 1) + 1;
  //The index in W from where the assignment of the i^th hidden node input weights are to end
  M = (i – 1)*(I + 1) + I;
    //Input weight initialization for the i^th hidden node from the uniform
  IW (i, :) = W (1, K: M);
    //Initialization of the i^th hidden node threshold
  HT (i, 1) = W (1, M + 1);
    //Calculate X for normalization of the weights where X is the length of the weights
  X = sqrt( sum( (IW (i,:))^2 + HT(i,1)^2 ) )
  //Normalize the input to hidden layer weights and the hidden layer node thresholds
  IW (i, :) = IW (i,:) * S * i / X
  HT (i, 1) = HT (i,1) * S * i / X
end for;
//Generate a uniform random number array with values in the range (-S, S) of size H and assign
it to hidden to output node weights
W (1, 1: H) = RAND (1, H);
W (1, 1: H) = 2*W (1, 1: H)*S – S;
HO (1, :) = W (1, 1: H);
//Set the output node threshold to zero
OT (1, 1) = 0;
```

Experimental Procedure

For each of the 10 problems and for each of the seven weight initialization procedures under consideration for analysis and comparison, 50 different networks are being trained. The RPROP algorithm is used for training these networks for 1000 epochs. When using the LMA as the training algorithm, the maximum number of epochs is set to 100 to prevent overtraining of the networks. The activation function to be used at the hidden layer is the tan hyperbolic activation function and the logarithmic sigmoidal function. The output node activation is linear in nature. Batch training is used for all the experiments. In batch mode, all the inputs in the training set are applied to the network before the weights are updated. For most problems, when using the Neural Network Toolbox software, batch training is significantly faster and produces smaller errors than incremental training [16].

The performance over the training and the test data sets is measured, for each network, using the mean squared error, where the desired output for the p^{th} input pattern is t(p) and the correspondingly obtained output is y(p), and the number of input patterns is P. The mean MSE (MMSE) values (over all the networks for a specific problem) for the training and test data sets are reported as the indicators of the average behavior of the weight initialization routines. This allows us to measure the goodness of training, i.e., the value of the error achieved during training, with lower values indicating better training, when the measurement is made over the training data set. The error over the test set measures the generalization capability of the network. Generalization capability means how well the neural network thus trained can respond to input patterns or data sets never seen before during training.

Also, the standard deviations of the MSEs across the networks, the median of the MSE, the minimum of the MSE and the maximum of the MSE are reported for an ensemble of network for each task, over the training data set and the test data set.

A two-sampled t-test is further carried out to check the null hypothesis that the weight initialization methods are equivalent in the sense that these techniques lead to the same distribution of MSEs achieved during training and for generalization, for a specific function approximation task and a specific training algorithm. All the t-tests are performed at 5% significance level.

A ranking scheme is used to analyze the data parameters obtained for each method. In this scheme, a method is ranked highest (rank 7) if the parameter being considered is lowest for all the methods. Thus, a method with rank 1 will have the highest value for that parameter. The average rank is then computed for each of the data parameters. Also, the average rank for the problems with one or two dimensions (problem 1 to problem 8) is also reported. The various parameters are (in order of indices): training MMSE, test MMSE, median of MSE for training data, median of MSE for test data, standard deviation for training data, standard deviation for test data, minimum of MSE for training data, minimum of MSE for test data, maximum of MSE for training data and maximum of MSE for test data.

RESULTS

This section deals with results and analysis of the experiments done by taking seven weight initialization methods under consideration. Experiments are conducted by implementing all these methods in MATLAB. After all the experiments, several data parameters are generated for each of the 50 networks and every method and problem under consideration. Four sets of experiments are conducted where the training algorithm is chosen as either RPROP or LMA and the activation function used at the hidden layer as either the tan hyperbolic function or the logarithmic sigmoidal activation function. The amount of data thus generated is huge. Due to the bulk of data produced as a result of four experiments, each for 50 networks, seven methods and 10 problems, this section reports only summary data for the sake of brevity. Each weight initialization method is given an index as follows (Table 2):

Table 2: *Indexing of Weight Initialization Methods.*

Index	Method/Function
1	BASE.m
2	BOTTOU.m
3	SMIEJA.m
4	YAM2001.m
5	NW.m
6	SCAWI.m
7	RAN001.m

Experiment I

In the first experiment, the training algorithm used is RPROP and the activation function at the hidden layer is the tan hyperbolic function. The average of the MMSE over all the ten function approximation problems for the training and generalization data sets is given in Table 3.

Table 3: *Average MMSE Over 10 Problems (Exp I).*

M/D	TR	TS
1	0.013268	0.028718
2	0.007298	0.01706
3	0.005571	0.024967
4	0.004804	0.014677
5	0.003732	0.012857
6	0.012767	0.028977
7	0.003815	0.013987

In the table given above, TR= Training Data, TS= Testing Data, M= Method index and D= Data set used.

The ranking of the methods over all the problems and over the first eight problems is tabulated in Tables 4 and 5 respectively. In these tables, P= Parameter index, and M= Weight initialization method index.

From these tables and detailed data (not reported here for the sake of brevity), the following may be inferred that:

1. The values of the MSEs achieved on an average for the weight initialization methods under consideration are entirely different for the different approximation tasks. The proposed weight initialization method has an average MMSE that is comparable to that of Nguyen Widrow weight initialization. Nguyen Widrow weight initialization is by far the most popular weight initialization technique used to train neural networks. The average MMSE for all the problems of this method and the proposed method are very close to each other (Nguyen Widrow's method having a slightly lesser average MMSE). Thus, it may be inferred that the proposed method is as good as the Nguyen Widrow weight initialization in this case.

a. The basic random weight initialization method and the SCAWI technique have the worst average MMSE among other methods for both, training and generalization.

b. Yam-Chows's method is found to be better than Smieja's method in terms of average MMSE, which in turn is better than LeCun-Bottou-Orr-Müller's. However, these methods show significant increase in average MMSE during generalization than during training. This shows that these methods learn the problem well but are not able to generalize with the same efficiency.

c. From the average ranks computed for each technique, it can be seen that Nguyen Widrow's weight initialization method has the highest rank followed by the new weight initialization method proposed in this manuscript. However, the difference in their ranks is insignificant and it may thus be inferred that the proposed method is comparable in performance to Nguyen Widrow's method.

d. Moreover, if the function approximation tasks with one or two dimensions are considered for rank computation, it is found that the proposed method now bears the highest rank. Thus, it supersedes the Nguyen-Widrow method.

e. The two-sampled t-test is performed over the training and test data sets for a 5% significance level. A matrix of dimensions 7×7 (i.e. 7 rows and 7 columns) is generated for each problem, both for training and generalization. Thus, the output of this hypothesis test is a structure containing 10 such matrices. Each matrix cell may contain either a zero or a one. Zero in a cell indicates that the means of both the methods specified by their indices in the row and column are same which the null hypothesis, is in this case. Likewise, a one in a cell indicates rejection of the null hypothesis. Through analysis, it is observed that the basic random weight initialization and SCAWI techniques are found to be equivalent. Both these methods are equivalent for 9 out of 10 problems (it is not the case only in problem 4) during testing and in seven out of 10 problems during training. Similarly, the proposed method and Nguyen-Widrow's method are found to be equivalent for many problems both during training and generalization.

Experiment II

The training algorithm used is RPROP and the activation function at the hidden layer is the logarithmic sigmoidal function. The average of the MMSE over all the 10 function approximation problems for the training and generalization data sets is given in Table 6.

The average rank for the problems and for problem 1 to problem 8 is reported in Tables 7 and 8 respectively.

From these tables, the following may be inferred that:

1. The values of the MSEs achieved on an average for the weight initialization methods under consideration are entirely different for the different approximation tasks.
2. The proposed weight initialization method has an average MMSE that is comparable to that of Nguyen Widrow weight initialization in this case as well. Nguyen Widrow weight initialization is by far the most popular weight initialization technique used to train neural networks. The average MMSE for all the problems of this method and the proposed method are very close to each other (Nguyen Widrow's method having a slightly lesser average MMSE). Thus, it may be inferred that the proposed method is as good as the Nguyen Widrow weight initialization in this case.
3. The basic random weight initialization method, the SCAWI technique and LeCun-Bottou-Orr-Müller's method have the worst average MMSE among other methods for both, training and generalization.
4. Yam-Chows's method is found to be better than Smieja's method in terms of average MMSE.
5. From the average ranks computed for each technique, it can be seen that Nguyen Widrow's weight initialization method has the highest rank followed by the new weight initialization method proposed in this manuscript. However, the difference in their ranks is insignificant and it may thus be inferred that the proposed method is comparable in performance to Nguyen Widrow's method.
6. Moreover, if the function approximation tasks with one or two dimensions are considered for rank computation, it is found that the proposed method now bears the highest rank. Thus, it supersedes the Nguyen-Widrow method.
7. The two-sampled t-test is performed over the training and test data sets for a 5% significance level. Through analysis, it is observed that the basic random weight initialization, SCAWI and Bottou's techniques are found to be equivalent in most problems, both for training and generalization. Similarly, the proposed method and Nguyen-Widrow's method are found to be equivalent for many problems both during training and generalization.

Table 4: *Ranking For 10 Problems (Exp I).*

M/P	1	2	3	4	5	6	7	8	9	10	Avg.
1	1.7	1.9	1	1.6	1	1.9	2.1	2.1	2.2	3.3	1.88
2	3.3	3.6	3.6	4	3.6	4.3	3.6	4.3	3.2	4.3	3.78
3	4.8	4.4	4.8	4.5	4.5	3.9	4.9	4.7	4	3.1	4.36
4	4.3	4.3	4.3	4.3	5.3	5.5	4	3.8	5.1	5.2	4.61
5	5.9	5.4	6	5.5	5	4.4	6	5.5	5.1	4.5	5.33
6	2.1	3.1	2.3	2.9	3.2	3.6	2.1	2.3	3.4	3.5	2.85
7	5.9	5.3	6	5.2	5.4	4.4	5.3	5.3	5	4.1	5.19

Table 5: *Ranking Over Problems 1 To 8 (Exp 1).*

M/P	1	2	3	4	5	6	7	8	9	10	Avg.
1	1.38	1.50	1.00	1.00	1.00	1.00	1.50	1.63	2.00	2.63	1.4625
2	3.25	3.38	3.50	3.88	3.50	4.38	3.50	3.75	3.38	4.25	3.675
3	5.38	5.13	5.25	5.13	5.00	4.63	5.50	5.63	4.38	3.63	4.9625
4	4.50	4.75	4.38	4.88	5.63	5.88	4.13	4.25	5.50	5.88	4.975
5	5.63	5.38	5.88	5.38	4.63	4.63	5.88	5.25	4.75	4.50	5.1875
6	2.00	2.13	2.13	2.13	3.13	2.88	2.00	1.88	3.38	2.75	2.4375
7	5.88	5.75	5.88	5.63	5.13	4.63	5.50	5.63	4.63	4.38	5.3

Table 6: *Average MMSE over 10 Problems (Exp II).*

M/D	TR	TS
1	1.15E-02	3.00E-02
2	0.011526	0.027364
3	0.005561	0.020625
4	0.005061	0.015875
5	0.004181	0.014282
6	0.011574	0.029859
7	0.004226	0.014916

Table 7: *Ranking for 10 Problems (Exp II).*

M/P	1	2	3	4	5	6	7	8	9	10	Avg.
1	3.1	2.5	2.9	2.6	2.9	3.4	3.1	2.8	3.4	3.7	3.04
2	2.3	3.7	2.4	4.1	2.8	3.3	3	2.7	2.9	3	3.02
3	3.9	3.6	4.7	3.8	4	3.6	4.1	4.2	4.4	4.3	4.06
4	4.3	4.6	4	4.1	4.1	3.5	4.7	4.8	4.4	4.7	4.32
5	6.1	5.8	5.3	5.3	5.9	5.5	5.5	5.7	5.2	4.9	5.52
6	2.5	2.9	2.7	3.1	2.7	2.7	3	3	3.1	3.4	2.91
7	5.8	4.9	6	5	5.6	6	4.6	4.8	4.6	4	5.13

Table 8: *Ranking Over Problems 1 To 8 (Exp II).*

M/P	1	2	3	4	5	6	7	8	9	10	Avg.
1	3.3	2.5	3.0	2.6	3.3	3.0	3.3	2.9	3.8	4.1	3.2
2	2.3	3.0	2.4	3.5	2.9	3.1	2.9	2.1	2.5	2.3	2.7
3	4.3	4.3	5.3	4.5	4.4	4.3	4.5	5.0	4.9	5.0	4.6
4	4.5	4.8	4.0	4.1	3.8	3.6	5.0	4.9	4.6	4.8	4.4
5	6.0	5.6	5.1	5.0	6.0	5.8	5.3	5.6	4.9	4.8	5.4
6	2.1	2.6	2.4	2.9	2.1	2.1	2.9	2.9	3.1	3.3	2.6
7	5.6	5.3	5.9	5.4	5.6	6.1	4.3	4.6	4.3	3.9	5.1

Experiment III

The training algorithm used is LMA and the activation function at the hidden layer is the tan hyperbolic function. The average of the MMSE over all the ten function approximation problems for the training and generalization data sets is given in Table 9. A similar ranking scheme is used in this experiment as well. The average rank for the problems and for problem 1 to problem 8 is reported (Tables 10 and 11).

From these tables, the following may be inferred that:

1. The values of the MSEs achieved on an average for the weight initialization methods under consideration are entirely different for the different approximation tasks.
2. The proposed weight initialization method has the lowest average MMSE among all other methods for training data over the ten problems. However, for the test data set, SCAWI and random weight initialization have shown better performance over all other methods.
3. Smieja's method has the worst average MMSE among other methods for both, training and generalization.
4. From the average ranks computed for each technique, it can be seen that LeCun-Bottou-Orr-Müller's weight initialization method has the highest rank for the 10 problems, very closely followed by the new weight initialization method proposed. Moreover, the difference in their ranks is insignificant and it may thus be inferred that the proposed method is comparable in performance to Bottou's method.
5. Moreover, if the function approximation tasks with one or two dimensions are considered for rank computation, it is found that the proposed method now bears the highest rank. Thus, it supersedes the LeCun-Bottou-Orr-Müller's method as well.
6. The two-sampled t-test is performed over the training and test data sets for a 5% significance level. Through analysis, it is observed that the basic random weight initialization and SCAWI techniques are found to be equivalent in most problems, both for training and generalization.

Experiment IV

The training algorithm used is LMA and the activation function at the hidden layer is the logarithmic sigmoid function. The average of the MMSE over all the 10 function approximation problems for the training and generalization data sets is given in Table 12. The average rank for the problems and for problem 1 to problem 8 is reported in Tables 13 and 14.

Table 9: *Average MMSE Over 10 Problems (EXP III).*

M/D	TR	TS
1	5.89E-04	5.30E-03
2	5.25E-04	6.11E-03
3	2.02E-03	2.25E-02
4	5.86E-04	9.35E-03
5	9.03E-04	9.61E-03
6	5.78E-04	5.12E-03
7	4.57E-04	7.12E-03

Table 10: *Ranking For 10 Problems (Exp III).*

M/P	1	2	3	4	5	6	7	8	9	10	Avg.
1	4	4.7	3.9	4.7	3.6	4.4	5.4	5.6	4.9	5.5	4.67
2	5.7	5.2	5.9	5.3	5	5.1	4.9	4.9	5.2	4.7	5.19
3	2	2.4	1.8	2	1.6	1.7	2.1	2.3	2.2	2.4	2.05
4	3.9	3.8	3.6	3.7	3.8	4.6	3.6	3.8	3.7	3.8	3.83
5	2	1.8	3.2	2.2	4.4	3	1.8	1.9	1.8	1.7	2.38
6	4.6	5.1	4	5.1	4.3	4	5.2	4.8	5	5.2	4.73
7	5.8	5	5.6	5	5.3	5.2	5	4.7	5.2	4.7	5.15

Table 11: *Ranking Over Problems 1 To 8 (Exp III).*

M/P	1	2	3	4	5	6	7	8	9	10	Avg.
1	3.8	4.4	3.5	4.3	3.3	3.8	5.3	5.5	4.9	5.4	4.4
2	6.0	5.6	6.1	5.8	4.9	5.1	5.4	5.1	5.6	5.0	5.5
3	2.3	2.8	2.0	2.3	1.8	1.9	2.4	2.6	2.5	2.8	2.3
4	4.1	4.0	4.0	4.0	4.0	4.8	3.8	4.0	3.9	4.0	4.1
5	1.8	1.5	3.0	1.8	4.3	3.1	1.5	1.6	1.5	1.4	2.1
6	4.1	4.6	3.5	4.8	4.0	3.8	4.8	4.5	4.5	4.8	4.3
7	6.0	5.1	5.9	5.3	5.9	5.6	5.0	4.6	5.1	4.8	5.3

Table 12: *Average MMSE over 10 Problems (Exp IV).*

M/D	TR	TS
1	0.001115	0.005763
2	0.000601	0.005418
3	0.000841	0.01748
4	0.000641	0.006702
5	0.000411	0.00598
6	0.001092	0.006455
7	0.000483	0.00586

Table 13: *Ranking for 10 Problems (Exp IV).*

M/P	1	2	3	4	5	6	7	8	9	10	Avg.
1	3	3.6	2.5	3.3	2.2	3	3.5	4.8	3.6	4.6	3.41
2	4.7	4.7	4.4	4.6	4.7	4.7	4.3	4.5	4.3	4.3	4.52
3	2.7	2.8	3.4	3.2	3.9	3.8	2.9	2.7	2.2	2.7	3.03
4	3.6	3.1	4.3	3.6	4.5	4.2	3.3	3.2	3.6	3.1	3.65
5	6	5.1	5.9	5	5.7	5.7	4.9	3.9	5.4	4	5.16
6	3.4	3.8	3.1	3.5	2.9	2.7	3.9	3.3	3.6	4.1	3.43
7	4.6	4.9	4.4	4.8	4.1	3.9	5.2	5.6	5.3	5.2	4.8

Table 14: *Ranking over Problems 1 to 8 (Exp IV).*

M/P	1	2	3	4	5	6	7	8	9	10	Avg.
1	2.8	2.9	2.0	2.5	2.1	2.8	3.4	4.3	3.4	4.0	3
2	4.5	4.6	4.1	4.4	4.6	4.5	4.1	4.6	4.3	4.3	4.4
3	3.1	3.3	4.0	3.8	4.6	4.5	3.1	3.0	2.5	3.0	3.4875
4	3.6	3.4	4.4	4.0	4.8	4.6	3.0	3.1	3.5	3.1	3.75
5	6.0	5.4	6.0	5.4	5.6	5.9	5.4	4.1	5.6	4.1	5.35
6	3.1	3.3	2.8	2.9	2.3	1.6	3.5	3.0	3.4	4.0	2.975
7	4.9	5.3	4.8	5.1	4.0	4.1	5.5	5.9	5.4	5.5	5.0375

CONCLUSION

In this work, a new weight initialization method for sigmoidal feed-forward artificial neural networks has been proposed. The proposed method has been compared with six other popular weight initialization methods on 10 function approximation problems. The proposed method of weight initialization distributes the initial weights and thresholds in such a manner that they lie in different regions of the activation function used at the hidden layer. The comparison shows that the proposed method of weight initialization is better than the other methods under consideration and is comparable (and even found to be better in some cases) in performance to the most popular method i.e., the Nguyen-Widrow weight initialization technique when the RPROP algorithm is used as the learning mechanism (for both types of activation function) and also in the case where the LMA method is used for learning with the logarithmic sigmoidal activation function being used at the hidden layer. It is also observed that for the case where the activation function at the hidden layer is the tan hyperbolic function along with LMA, the proposed method is found to be comparable (and even found to be better in some cases) in performance to the weight initialization method given by LeCun, Bottou, Orr and Müller. Thus, it can be inferred that in all the experiments carried out in this work, the proposed method is found to work well and reach deeper minima and thus improve the learning of the neural network for both the popular versions of back propagation learning and utilizing the most widely used activation functions of the sigmoidal class at the hidden layer nodes.

Further, it is also inferred from the analysis of the null hypothesis that, for all the experiments, the basic random weight initialization and SCAWI techniques are similar in performance (in two of these experiments, the LeCun-Bottou-Orr-Müller method is equivalent to the basic and SCAWI methods). For three of the four experiments, the proposed method is found to be equivalent to the Nguyen-Widrow weight initialization technique. Further experimentation and comparison with other methods of weight initialization as proposed in literature may be required to draw a more confident statement concerning the efficiency and efficacy of the proposed method [4–6, 11, 17–26].

REFERENCES

1. Simon Haykin. *Neural Networks: A Comprehensive Foundation*. 2nd Edn. Pearson Education Inc.; 2004.
2. Kolen John F, Pollack Jordan B. *Back Propagation is Sensitive to Initial Conditions, Complex Systems*. Morgan Kaufmann Publishers; Ohio, USA; 1990; 860–867p.
3. Nguyen D, Widrow B. Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. *International Joint Conference on Neural Networks (IJCNN)*. 17–21 Jun 1990; 3: 21, 26p.
4. Drago G, Ridella S. Statistically Controlled Activation Weight Initialization (SCAWI). *IEEE Trans Neural Netw*. 1992; 3(4): 627–631p.
5. Thimm G, Fiesler E. High-Order and Multilayer Perceptron Initialization. *IEEE Trans Neural Netw*. 1997; 8(2): 349–359p.
6. Yam Y, Chow TW, Leung C. A New Method in Determining Initial Weights of Feed-Forward Neural Networks for Training Enhancement. *Neurocomput*. 1997; 16(1): 23–32p.
7. Yam Y, Chow TW. Feed-forward Networks Training Speed Enhancement by Optimal Initialization of the Synaptic

Coefficients. *IEEE Trans Neural Netw*. 2001; 12(2): 430–434p.
8. LeCun Y, Bottou L, Orr GB, *et al*. Efficient Backprop. In *Neural Networks: Tricks of the Trade, Ser. LNCS:* 1524. Orr GB, Müller KR, editors. Berlin: Springer; 1998; 9–50p.
9. Smieja Frank J. *Hyper-plane 'Spin' Dynamics, Network Plasticity and Back-propagation Learning.* GMD, St. Augustin, Germany, GMD Rep. Nov 28, 1991.
10. Balázs Csanád Csáji. *Approximation with Artificial Neural Networks*. Faculty of Sciences, Eötvös Loránd University, Hungary. 2001.
11. Lehtokangas M, Saarinen J, Kaski K, *et al*. Initializing Weights of a Multilayer Perceptron Network by Using the Orthogonal Least Square Algorithm. *Neural Comput*. Sep 1995; 7(5): 982–999p.
12. Breiman L. The Π-Method for Estimating Multivariate Functions from Noisy Data. *Technometrics*. 1991; 33: 125–143p.
13. [Cherkassky V, Müller F. *Learning from Data: Concepts, Theory and Methods*. New York: John Wiley; 1998.
14. Cherkassky V, Gehring D, Müller F. Comparison of Adaptive Methods for Function Estimation from Samples. *IEEE Trans Neural Netw*. 1996; 7(4): 969–984p.
15. Maechler M, Martin D, Schimert J, *et al*. Projection Pursuit Learning Networks for Regression. In *Proc. of the 2nd International IEEE Conference on Tools for Artificial Intelligence*. 1990; 350–358p.
16. The MATLAB Documentation Centre. URL: http://www.mathworks.in/help/matlab/.
17. Kim Y, Ra JB. Weight Value Initialization for Improving Training Speed in the Back-Propagation Network. In *Proc. of IEEE International Joint Conference on Neural Networks*. 1991; 3: 2396–2401p.
18. Chen CL, Nutter RS. Improving the Training Speed of Three-Layer Feed-Forward Neural Nets by Optimal Estimation of the Initial Weights. In *Proc. of IEEE International Joint Conference on Neural Networks*. 1991; 3: 2063–2068p.
19. Wessels LFA, Barnard E. Avoiding False Local Minima by Proper Initialization of Connections. *IEEE Trans Neural Netw*. 1992; 5(6): 899–905p.
20. Weymaere N, Martens JP. On the Initialization and Optimization of Multilayer Perceptrons. *IEEE Trans Neural Netw*. 1994; 5(5): 738–751p.
21. Jamett M, Acuña G. An Interval Approach for Weight's Initialization of Feedforward Neural Networks. In *MICAI 2006: Advances in Artificial Intelligence, Ser. Lecture Notes in Computer Science*. Gelbukh A, Reyes-Garcia C, editors. Berlin, Heidelberg: Springer; 2006; 4293: 305–315p.
22. Xhang XM, Chen YQ, Ansari N, *et al*. Mini-Max Initialization for Function Approximation. *Neurocomput*. 2004; 57: 389–409p.
23. Shepanski JF. Fast Learning in Artificial Neural Systems: Multilayer Perceptron Training using Optimal Estimation. In *Proc. of IEEE International Conference on Neural Networks*. 1988; 1: 465–472p.
24. de Castro LN, Iyoda EM, Zuben FJV, *et al*. Feedforward Neural Network Initialization: An Evolutionary Approach. In *Proc. of Vth Brazilian Symposium on Neural Networks*. de Pdua Braga A, Ludermir TB, editors. IEEE Computer Society; 1998; 43–48p.
25. Fernandez-Redondo M, Hernandez-Espinosa C. Weight Initialization Methods for Multilayer Feedforward Networks. In *European Symposium on Artificial Neural Networks*. 2001; 119–124p.
26. Timotheou S. A Novel Weight Initialization Method for the Random Neural Network. *Neurocomput*. 2009; 73(13): 160–168p.

Cite this Article
Rajashree Chaurasia. A Novel Strategy for Weight Initialization in Sigmoidal Feed-forward Artificial Neural Networks. *Journal of Artificial Intelligence Research & Advances*. 2018; 5(1): 62–75p.