2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A New Weight Initialization Method for Sigmoidal Feedforward Artificial Neural Networks

Sartaj Singh Sodhi, Pravin Chandra and Sharad Tanwar

***Abstract*—Initial weight choice has been recognized to be an important aspect of the training methodology for sigmoidal feedforward neural networks. In this paper, a new mechanism for weight initialization is proposed. The mechanism distributes the initial input to output weights in a manner that all weights (including thresholds) leading into a hidden layer are uniformly distributed in a region and the center of the region from which the weights are sampled are such that no region overlaps for two distinct hidden nodes. The proposed method is compared against random weight initialization routines on five function approximation tasks using the Resilient Backpropagation (RPROP) algorithm for training. The proposed method is shown to lead to about twice as fast convergence to a pre-specified goal for training as compared to any of the random weight initialization methods. Moreover, it is shown that at least for these problems the networks reach a deeper minima of the error functional during training and generalizes better than the networks trained whose weights were initialized by random weight initialization methods.**

I. Introduction

WEIGHT initialization in sigmoidal feedforward artificial neural networks (FFANN) have been shown to have an affect on the training speed of the network [1], [2], [3], [4], [5], [6], [7]. Usually, weights are initialized to small uniform random values; though efforts have been made to design weight initialization mechanisms based on the structure of computation in these networks and the dynamics of weight evolution during training [3], [4], [5], [6], [7].

For the solution of a learning task, more than one network starting from different initial points in the weight space are trained; and the FFANN with the minimum error is utilized as the solver of the task. This entails that a large number of networks are trained, using substantial time and resources. Thus, any mechanism that may allow for faster and better training of FFANNs is worth investigation. The term "faster training" is used with the meaning that a network can be trained (using a specified training mechanism) faster if it reaches a pre-specified goal / error of training in lesser number of training epochs, as compared to other training mechanisms. While the phrase "better training" is used in two contexts, namely:

1) The network, during training, reaches a lower minimum of the error functional (used for measuring the quality of training) in equal number of epochs as compared to when trained using other training mechanism, and
2) The network thus trained has better generalization capability (error on data not used for training is lower) as compared to networks trained by other training mechanisms for equal number of epochs.

The training mechanism for a FFANN (primarily[1]) involves the following choices:

1) Architectural Choice: The question of how many hidden layers and the number of nodes in each hidden layer must be answered.
2) Activation Function Choice: The choice of activation functions used at hidden layers' nodes and the output layer node must be answered.
3) Training Algorithm Choice: The choice of the training algorithm must be made.
4) Weight Initialization Choice: The procedure / methodology for the choice of the initial weights must be specified.

Some of the procedures for speeding up the training (or reducing the time devoted to training) that have been reported in literature range from efficient choice of initial weights [3], [4], [5], [6], [7], choice of activation function [8], stopping training or multiple restart training mechanisms [9], etc.

These questions are detailed and the appropriate choices made, in context to the work reported in this paper, in Section II. In this paper a proposal for a new mechanism for weight initialization is made. The weight initialization routine proposed is demonstrated to be equivalent to if not better than the random weight initialization scheme on a set of 5 function approximation problems.

The paper is organized as follows: Section II describes the design of experiments including the proposed weight initialization method. Results of the experiments are presented and discussed in Section III while conclusions are presented in Section IV.

II. Design of Experiments

Neural networks have been shown to be able to approximate a continuous function arbitrarily well [10], [11], [12] (Universal Approximation Property (UAP)). The design of

S. S. Sodhi and P. Chandra are with the University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University, Dwarka, Sector 16C, New Delhi (INDIA) - 110078 (email: {sartaj, pchandra}@ipu.ac.in); sartajsodhi@yahoo.com; chandra.pravin@gmail.com. S. Tanwar is with Deloitte Consulting India Private Ltd., Udhyog Vihar Phase 4, Gurgaon Haryana (INDIA) - 122015 (email: sharad_tanwar@outlook.com)

[1]Other than the choices enumerated, there are other questions that must be answered for the complete detailing of the training mechanism like the choice of error functional used for measuring the mismatch between the desired and the obtained response from the network for any input set, the pre-proccesing of data before training, etc.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

a neural network for a function approximation task involves the following:

A. Architecture

The number of hidden layers used has to be decided. The UAP requires that minimum one hidden layer with sigmoidal nodes be present [10], [11], [12] Thus, in this paper, one hidden layer networks are used. The number of inputs in the input layer and the number of outputs in the output layer are decided by the number of independent variables and the number of dependent variables, respectively. Moreover, a vector function (a function with more than one dependent variable), may be treated as a set of functions with the same set of independent variable. Thus, in the following, we will treat functions with one dependent variables only, thereby implying that the networks used have only one node in the output layer. The decision of how many sigmoidal nodes are used in the hidden layer (for the sake of brevity, these nodes will be called hidden nodes), is decided by 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.

B. Activation Function

Hidden nodes output functions are required to be sigmoidal in nature (see [8] and [13] for a survey of activation functions). The generally used activation functions are the anti-symmetric *hyperbolic–tangent* function:

$$\sigma_1(x) = \tanh(x) = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}} \tag{1}$$

and the asymmetric *logistic* or the *log-sigmoid* function:

$$\sigma_2(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{2}$$

Preference is made for anti-symmetric activation [14], [15], thus in this work, the activation function used at the hidden nodes is the hyperbolic tangent function (1).

C. Training Algorithm

The algorithms used for adjusting the weights, so as to minimize the error (the mismatch between the desired output from a network and the obtained output, for a given input tuple), are based on local nonlinear optimization methods [16], [17], [15] , though global optimization techniques have also been utilized for the training of sigmoidal FFANNs (for example, see [18]) . The local nonlinear optimization methods based training algorithms range from gradient descent based algorithms like the standard backpropagation algorithm [16] to second order algorithms using the Hessian [15]. In this paper we use the resilient backpropagation algorithm (RPROP) [17], [19]. RPROP [17], [19] is a fast convergent first order method that has lesser memory requirements than the second order methods.

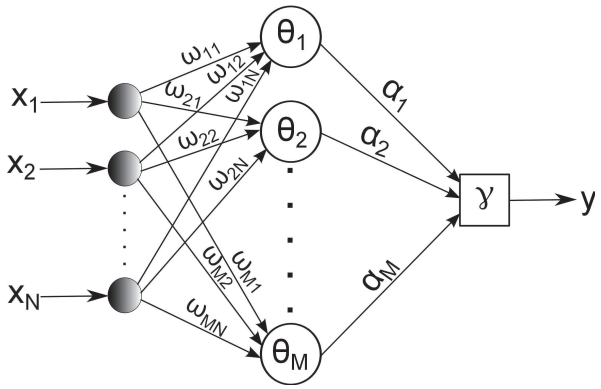


Fig. 1: The schematic diagram of a single hidden layer network.

D. Weight Initialization

The connection weights between the nodes and the thresholds of the nodes are collectively known as the weights of the network. Usually 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, $\lambda \in \{0.25, 0.50, 0.75, 1.00\}$. Random weight initialization is done so as to break the weight symmetry during training. If weights are initialized to equal values, they move/evolve during training in tandem/groups [16]. The RPROP algorithm works in the batch-mode, that is, the weights are updated based on the error values corresponding to the complete error over the training set. If the pth output desired is $t^{(p)}$ and the output obtained from the network is $y^{(p)}$ for the pth input set, and if there are P number of training exemplars, then the error is measured by the mean squared error[2] (MSE) defined as:

$$\mathrm{MSE} = \frac{1}{P}\sum_{p=1}^{P}(t^{(p)} - y^{(p)})^2 \tag{3}$$

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 as w (the weight between the ith hidden node and the jth input node is w_{ij}), the threshold of the ith hidden node is θ_i, and for any one input tuple, the net input to the ith hidden node is:

$$n_i = \sum_{j=1}^{N} w_{ij}x_j + \theta_i \tag{4}$$

The output from the node is $h_i(x_1, \ldots, x_N) = \tanh(n_i)$. The weight between the ith hidden node and the output node is represented by α_i, where $i \in 1, 2, \ldots, M$ and the threshold of the output node is γ. Then the output from the network

[2]During training, the MSE is usually defined as an equivalent quantity up-to a factor of 2 as MSE' = MSE / 2.

is:

$$y = \sum_{m=1}^{M} \alpha_m h_m + \gamma \tag{5}$$

That is, the network output function uses a linear node. The experiments were done using Matlab version 2013a [20] on a 64-bit Intel i7 based Microsoft Windows 7 system with 6GB RAM.

E. Function Approximation Tasks

The following 5 function approximation tasks are used for the experiments:

1) One dimensional input function taken from MATLAB sample file $humps.m$.

$$f_1(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.4} - 6 \tag{6}$$

where $x \in [0, 1]$. See Fig. 2a.

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

$$\begin{aligned} f_2(x_1, x_2) = & 3(1-x_1)^2 \mathrm{e}^{(-x_1^2-(x_2+1)^2)} \\ & -10(x_1/5 - x_1^3 - x_2^5)\mathrm{e}^{(-x_1^2-x_2^2)} \\ & -(1/3)\mathrm{e}^{(-(x_1+1)^2-x_2^2)} \end{aligned} \tag{7}$$

where $x_1 \in [-3, 3]$ and $x_2 \in [-3, 3]$. See Fig. 2b.

3) Two dimensional input function from [21], [22], [23].

$$f_3(x_1, x_2) = \sin(x_1\, x_2) \tag{8}$$

where $x_1 \in [-2, 2]$ and $x_2 \in [-2, 2]$. See Fig. 2c.

4) Two dimensional input function from [21], [22], [23].

$$f_4(x_1, x_2) = \mathrm{e}^{(x_1 \sin(\pi\, x_2))} \tag{9}$$

where $x_1 \in [-1, 1]$ and $x_2 \in [-1, 1]$. See Fig. 2d.

5) Two dimensional input function from [24], [22], [23].

$$\begin{aligned} f_5(x_1, x_2) = & 1.3356(1.5(1-x_1) + \\ & \mathrm{e}^{2x_1-1}\sin(3\pi(x_1-0.6)^2) + \\ & \mathrm{e}^{3(x_2-0.5)}\sin(4\pi(x_2-0.9)^2)) \end{aligned} \tag{10}$$

where $x_1 \in [0, 1]$ and $x_2 \in [0, 1]$. See Fig. 2e.

1) Data sets: For each of the function approximation tasks, a set of of 1200 input points are generated by uniform random sampling of the input domain of the function and the corresponding outputs calculated. This set is partitioned into a set of 200 points represented as $\mathcal{TR}_i$ for the ith function approximation task ($i \in \{1, 2, \ldots, 5\}$) and is used for training the networks (the training set). The next partition of size 1000 is called the test set (represented by $\mathcal{T}_i$); where $i \in \{1, 2, \ldots, 5\}$.

All data (both input and output) are scaled to [-1,1] for training and further reporting.

2) Architecture of FFANN 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 one. The architecture thus fixed is summarized in Table I.

TABLE I: NETWORK SIZE SUMMARY.

Sr.No.	Function	Inputs	Hidden Layer Size	Outputs
1.	f_1	1	8	1
2.	f_2	2	15	1
3.	f_3	2	12	1
4.	f_4	2	10	1
5.	f_5	2	10	1

F. Weight Initialization Mechanisms

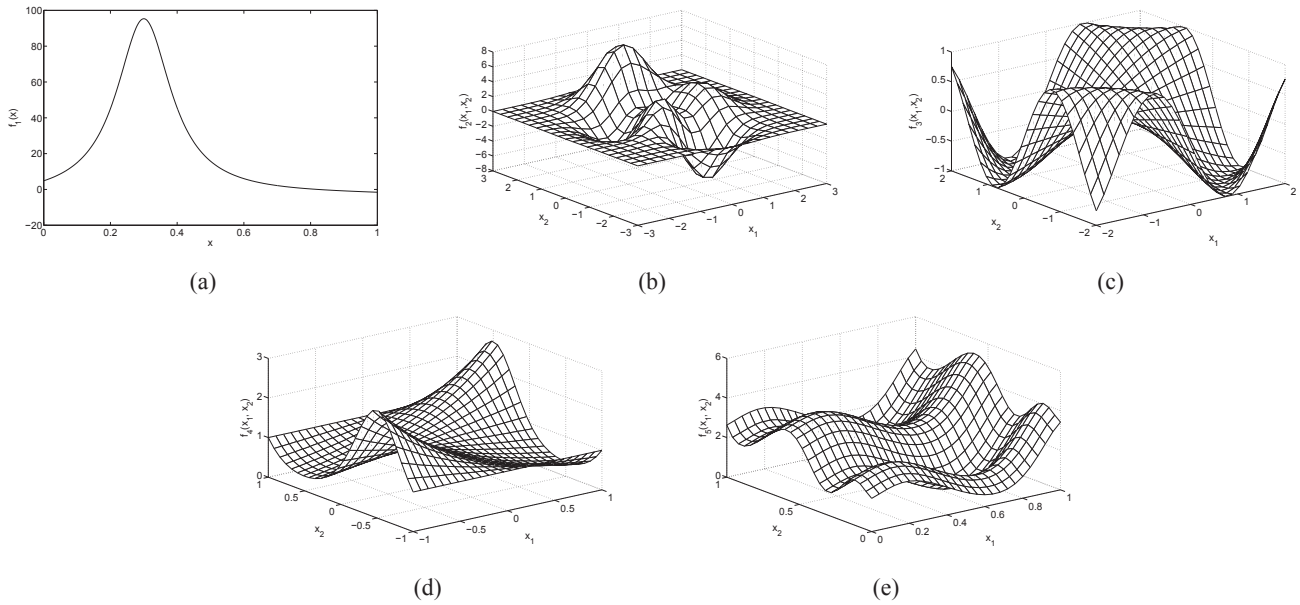
1) Random Weight Initialization Routine(s): Four classes of random weight initialization routines labeled as $WTR_i, i \in \{1, 2, 3, 4\}$ are used. For each i thirty sets of weights are generated by uniform random numbers between $[-\lambda, \lambda]$, where $\lambda \in \{0.25, 0.50, 0.75, 1.00\}$. That is all weights and thresholds are initialized by uniform random numbers in the N-dimensional cube $[-\lambda, \lambda]^N$.

2) Proposed Weight Initialization Routine: The weight initialization routines of class WTR distribute the weights (including the thresholds) leading into the hidden layer uniformly in a cube $[-\lambda, \lambda]^N$, centered at the origin, with each edge of size 2λ. The proposed weight initialization routine uses a portion of this cube for the weights leading into the hidden nodes from the the inputs. For the ith hidden node, these weights are initialized to a cube of size given by $[-S/2, S/2]^N$, where $S = 2\lambda/(M-1)$ and $\lambda \in \{0.25, 0.50, 0.75, 1.00\}$, with the center of the cube being at the point $(-\lambda + S(i-1), -\lambda + S(i-1), \ldots, -\lambda + S(i-1))$ in an N-dimensional space. Effectively, this implies that any of the inputs to the ith hidden node is multiplied by a connection strength weight belonging to the interval $[-\lambda + S(i-1) - S/2, -\lambda + S(i-1) + S/2]$. The threshold of the ith hidden node is initialized to the value $S(i-1)$. This mechanism guarantees that like the WTR's, the weights in these routines are also initialized to values within $[-\lambda, \lambda]$ across hidden nodes. But, unlike random weight initialization routines (WTRs), where the the maximum and the minimum value of the weights leading in to one node defines a region/interval of the interval $[-\lambda, \lambda]$ which may overlap the similar region defined by the maximum and the minimum weights leading into some other hidden node. The term weight region represents the interval defined by the minimum and the maximum weight value leading into the node; for the proposed weight initialization method, the interval defined by the maximum and the minimum value of the weights leading into two distinct hidden nodes will be non-overlapping.

The hidden nodes to the output node weights are initialized to uniform random values in the interval $[-0.5, 0.5]$ and the output node threshold is initialized to zero. The four classes of weight initialization routines are labeled as NEW_i, where $i \in \{1, 2, 3, 4\}$. The weight initialization method is given in an algorithmic form in Algorithm 1.

G. Experiment Procedure

For each of the problem and for each weight initialization procedure(s) of the class WTR and NEW, 30 different networks are trained for 1000 epochs of training using the



(a) (b) (c)

(d) (e)

Fig. 2: Functions f_1 (6) to f_5 (10).

TABLE II: TRAINING GOAL (MSE) FOR MEASURING THE SPEED OF CONVERGENCE.

Sr.No.	Function	MSE desired
1.	f_1	0.0145
2.	f_2	0.0278
3.	f_3	0.0262
4.	f_4	0.0049
5.	f_5	0.0083

RPROP algorithm. The performance over the training and the test data sets is measured (for each network trained) using the mean squared error (where the desired output for the pth input pattern is $t^{(p)}$ and the correspondingly obtained output is $y^{(p)}$ and the number of input patterns is P).

We report the mean MSE (MMSE) values (over all the networks for a specific problem) for the training and the test data set as the indicator of the mean / average behavior of the weight initialization routines. We also report the standard deviation (St.Dev.) of the MSE's across the networks, the Median of the MSE, the minimum (Min.) MSE, and the maximum (Max.) MSE for an ensemble of network for each task, over the training data and the test data set.

We also report on the number of epochs required by a network initialized by a routine, to reach a pre-specified MSE. The goal (MSE) to be achieved during training was taken as the worst MMSE among the WTR initialization routines of the previous experiment to which was added three times the standard deviation of the MSEs of the networks ensemble associated with the worst MMSE (that is obtained using the data of Table II), during training, for each function approximation task. The goals of training is specified in Table II.

A particular network may not converge in 1000 epochs to the specified goal, in this case the value of the epochs required to achieve the goal is arbitrarily, replaced with the maximum epoch, that is 1001. This may lead to a small bias in the reported result. The number of networks that do not converge is also reported.

III. RESULTS

The summary data for the first experiment conducted for a fixed number of epochs (1000) for each task and every weight initialization method, for the training data and the test data, are shown in Table III and IV, respectively. From the summary data of Table III – Table IV, we may infer the following:

1) The four WTR's are not equivalent for training / generalization. That is the value of the MSE achieved on an average for these weight initialization routines are entirely different for the different approximation tasks. Moreover, no one weight initialization method is better in terms of the MMSE for all the five tasks, among the WTRs.
2) The four NEW's are also not equivalent for training / generalization. That is the value of the MSE achieved on an average for these weight initialization routines are entirely different for the different approximation tasks. But the method NEW_4 is better than the other three methods of this class on all five tasks.
3) The NEW_4 weight initialization routine reaches lower error functional minima, on an average, as compared to the 4 random weight initialization routines used in the experiments for both the training and the generalization experiments.
4) The network with the minimum MSE on the training set may not belong to the network ensemble corresponding to the lowest MMSE value. This can be seen

Algorithm 1: PROPOSED WEIGHT INITIALIZATION ALGORITHM (NEW).

Input: N: Number of inputs, M: Number of hidden nodes, λ: Weight initialization parameter.

Output: ω: Input to hidden weights, θ: Hidden nodes threshold. α: Hidden to output node weights, γ: Output node threshold.

// Calculate S

$S = \frac{2\lambda}{M-1}$;

// Generate a uniform random number array with values in the range $(-S/2, S/2)$ of size $HWT = (N+1)M$ and assign it to the array W.

$MAX = N \times M$;

// uniform random numbers between (0,1), $RAND$ generates uniform random numbers between (0,1) of the array size specified.

$W = RAND(1 : MAX) \times S$;

// Shift by $S/2$, to get the weights in the interval $[-S/2, S/2]$

$W = W - \frac{S}{2}$;

// Shift by λ, so that the weight matrix is shifted such that the interval of weights is $[-\lambda - S/2, -\lambda + S/2]$

$W = W - \lambda$;

// Initialize the input weights and threshold of the hidden nodes.

for *i=1* **to** M **do**

  // The index in W from where the assignment of the ith hidden nodes input weights are to start.

  $K = (i-1) \times N + 1$;

  // The index in W till where the assignment of the ith hidden nodes input weights are to end.

  $M = i \times N$;

  // Input weight initialization for the ith hidden node. For the ith node the effective weight interval shifts to $[-\lambda - S/2 + S \times (i-1), -\lambda + S/2 + S \times (i-1)]$

  $\omega_i = W(K : M) + S \times (i-1)$ // Initialization of the ith hidden node threshold.

  $\theta_i = S \times (i-1)$

// Generate a uniform random number array with values in the range $(-0.5, 0.5)$ of size M and assign it to α.

$\alpha = RAND(1 : M) - 0.5$;

// Output node threshold set to zero.

$\gamma = 0$;

from the training data (Table III), for example for the function f_1, among the WTR methods WTR_4 has the least MMSE while the lowest MSE is achieved by a network in the class WTR_2. While for the same function, among the methods NEW, the lowest MMSE is obtained for the method NEW_4 while the network with the minimum MSE is in the ensemble corresponding to the method NEW_2.

5) A similar inference can also be made for the generalization results (Table IV), wherein it can be seen that the network weight initialization method that leads to the lowest MMSE may not contain the network for which the generalization error is minimum across the experiments.
6) The method NEW_4 has the best MMSE value across the tasks for both training and generalization experiments.
7) We performed a two sample (2-tailed) t-test to check the null hypothesis: "*The weight initialization routines are equivalent in the sense that all weight initialization routines lead to the same distribution of MSEs achieved during training and for generalization, for a specific function task and a specific training algorithm*". The proposed weight initialization algorithm (NEW_4) was equivalent to the WTR_3 for the approximation of the function f_1 (for both the training as well as the test/generalization data). NEW_4 is equivalent to WTR_1 and WTR_2 for the approximation of the function f_2 for training but is equivalent to WTR_2 only for the generalization experiment. For the task of approximating the function f_3, from the training data, it was obtained that the method NEW_4 was better than any of the four WTR's, but from the generalization data, the method was equivalent to WTR_1 and WTR_4. For the function f_4 and f_5, the method NEW_4 was not equivalent to any of the WTR's in any case. This is in consonance with the values of the Table III. These results are at the 5% significance level.

8) On performing a one-tail t-test, it was observed that the either the results for NEW_4 is equivalent to the result for the best WTR's or is better in the sense of achieving deeper minima on an average. The details are on the same lines as above for the two-sided t-test.

TABLE III: SUMMARY OF TRAINING DATA FOR THE WEIGHT INITIALIZATION CLASSES OVER 30 NETWORKS FOR EACH TASK. ALL VALUES OF THE STATISTICS ARE REPORTED $\times 10^{-3}$.

Method	Statistic	f_1	f_2	f_3	f_4	f_5
WTR_1	MMSE	2·435	13·431	12·730	2·698	3·955
	St.Dev.	4·015	3·910	3·029	0·818	1·450
	Median	0·562	12·886	12·365	2·465	4·233
	Min.	0·102	7·759	6·784	1·377	1·203
	Max.	17·903	23·135	20·452	4·583	6·057
WTR_2	MMSE	0·886	13·079	14·570	2·881	3·842
	St.Dev.	1·551	3·772	3·863	0·661	1·261
	Median	0·387	12·157	13·741	2·812	4·051
	Min.	0·051	7·461	8·571	1·511	1·440
	Max.	7·552	21·502	23·653	5·044	5·906
WTR_3	MMSE	0·903	14·458	13·423	2·726	3·636
	St.Dev.	3·037	3·005	3·386	0·780	1·688
	Median	0·244	14·973	12·548	2·657	4·163
	Min.	0·068	8·317	7·394	1·497	1·172
	Max.	16·910	19·625	22·686	4·582	6·083
WTR_4	MMSE	0·399	14·735	12·931	2·684	3·759
	St.Dev.	0·368	4·347	2·007	0·796	1·521
	Median	0·262	14·863	12·674	2·671	3·955
	Min.	0·090	6·538	8·922	1·179	0·886
	Max.	1·851	24·834	17·262	4·351	6·067
NEW_1	MMSE	0·781	16·855	16·083	2·561	4·134
	St.Dev.	1·451	4·445	5·461	0·627	1·593
	Median	0·286	16·705	14·948	2·486	4·367
	Min.	0·083	8·300	8·227	1·330	1·371
	Max.	6·312	29·111	35·751	3·842	7·533
NEW_2	MMSE	0·603	14·144	13·310	2·265	3·506
	St.Dev.	1·096	2·970	3·439	0·474	1·258
	Median	0·218	14·021	12·843	2·251	3·459
	Min.	0·026	9·681	7·285	1·566	0·966
	Max.	5·783	19·935	21·634	3·538	5·918
NEW_3	MMSE	0·193	13·712	11·893	1·948	2·517
	St.Dev.	0·098	2·581	2·533	0·630	1·412
	Median	0·191	13·894	11·562	1·873	1·966
	Min.	0·028	6·411	6·495	0·940	0·836
	Max.	0·522	19·041	17·592	3·943	5·003
NEW_4	MMSE	0·166	12·264	10·211	1·675	2·156
	St.Dev.	0·078	2·968	2·194	0·533	1·932
	Median	0·170	12·437	10·565	1·657	1·146
	Min.	0·033	6·409	5·294	0·877	0·599
	Max.	0·342	18·809	13·990	3·118	7·622

The second set of experiments was performed to assess the speed of training for achieving the goals of training as specified in Table II. The summary of the obtained results is shown in Table V. From the table it can be seen that the proposed mechanisms are generally faster in convergence as compared to the methods of WTRs, except for the method NEW_1 for the function approximation of f_1, which is slower in convergence as compared to WTR_4. The method NEW_4 is faster than all other methods of weight initialization. As can be seen from the table, some of the networks for some of the function approximation tasks (specifically see data for f_1 and f_4) for the WTR class of weight initialization

TABLE IV: SUMMARY OF TEST DATA FOR THE WEIGHT INITIALIZATION CLASSES OVER 30 NETWORKS FOR EACH TASK. ALL VALUES OF THE STATISTICS ARE REPORTED $\times 10^{-3}$.

Method	Statistic	f_1	f_2	f_3	f_4	f_5
WTR_1	MMSE	2·649	31·167	24·541	3·911	5·679
	St.Dev.	4·277	7·436	4·727	1·184	1·962
	Median	0·633	31·741	23·870	3·642	5·969
	Min.	0·106	13·621	13·105	2·058	1·772
	Max.	18·802	46·662	32·915	6·183	8·008
WTR_2	MMSE	0·983	27·837	27·614	4·177	5·837
	St.Dev.	1·668	6·891	6·060	0·938	1·973
	Median	0·444	29·748	26·929	3·951	6·285
	Min.	0·067	15·099	17·915	2·363	2·014
	Max.	8·074	38·422	40·580	6·782	9·266
WTR_3	MMSE	0·983	28·772	25·118	3·940	5·308
	St.Dev.	3·189	4·708	5·606	1·037	2·362
	Median	0·284	29·477	23·160	3·924	5·929
	Min.	0·070	18·007	16·408	2·126	1·775
	Max.	17·780	36·781	38·195	6·021	8·984
WTR_4	MMSE	0·459	26·254	24·587	3·854	5·610
	St.Dev.	0·415	5·809	3·966	1·124	2·352
	Median	0·307	27·345	24·605	3·922	5·985
	Min.	0·100	11·237	17·527	1·778	1·613
	Max.	2·125	38·322	33·684	6·014	10·057
NEW_1	MMSE	0·867	32·445	31·323	3·708	6·124
	St.Dev.	1·569	5·795	7·527	1·036	2·332
	Median	0·325	31·995	30·691	3·496	7·016
	Min.	0·110	21·350	18·718	1·783	2·052
	Max.	6·810	44·848	57·113	5·807	10·164
NEW_2	MMSE	0·679	31·117	25·925	3·136	5·616
	St.Dev.	1·206	5·217	5·475	0·810	1·950
	Median	0·250	31·161	25·101	3·041	5·286
	Min.	0·039	16·107	17·743	2·074	1·984
	Max.	6·384	39·588	35·772	5·794	8·913
NEW_3	MMSE	0·226	29·775	24·096	2·727	3·983
	St.Dev.	0·110	4·789	3·921	0·884	2·109
	Median	0·224	29·962	23·794	2·478	3·077
	Min.	0·039	13·882	13·310	1·399	1·621
	Max.	0·583	37·832	33·369	5·787	8·301
NEW_4	MMSE	0·193	26·177	22·425	2·451	3·540
	St.Dev.	0·088	5·878	4·628	0·808	2·635
	Median	0·196	27·243	23·989	2·320	2·281
	Min.	0·044	13·224	14·571	1·404	1·076
	Max.	0·381	36·012	30·547	4·485	10·610

routines do not converge in the maximum number of epochs (1000). Whereas, for the NEW set of methods, all networks converge to the desired value.

Moreover, from the data for the WTR_1 which is the best method in terms of the mean number of epochs for f_2, out of the 4 WTR methods, the network with the fastest convergence actually belongs to WTR_4 (see the Min. value). Such a pattern is not seen for the NEW set of methods.

The details of the speed comparision for the convergence is shown in Table VI. In the table "Average Speedup" refers to the average speed up (the ratio of the mean epochs for convergence of the best WTR class of weight initialization to the NEW_4 class of weight initialization method), while "best Speedup" refers to the ratio of the minimal epoch required by any network initialized by any of the WTR methods to the minimal epoch required by any network initialized by any of the NEW methods (which in all cases

is NEW_4).

The ratio "Average Speedup" ranges in the interval $[1.69, 3.62]$, and the average of these values is 2.92 (as an average over Average Speedup across tasks). And, on the basis of the best speed of convergence (the network that converges the fastest in any ensemble), the "Best Speedup" value lies in the interval $[1.88, 3.04]$, with the average of the ratio of convergence for the best initialization routine belonging to WTR and NEW is 2.54. Thus, we conclude that the the method of weight initialization NEW_4 is about two and half times faster than the WTR method of weight initialization.

TABLE V: TRAINING DATA EPOCHS SUMMARY REQUIRED TO ACHIEVE THE GOALS SPECIFIED IN TABLE II. NCN STANDS FOR NUMBER OF NON-CONVERGENT NETWORKS.

Method	Statistic	Functions				
		f_1	f_2	f_3	f_4	f_5
WTR_1	MEAN	208·07	111·57	298·10	382·07	247·20
	St.Dev.	189·56	19·55	99·52	151·79	112·59
	Median	134·50	112·00	294·00	361·50	214·50
	Min.	84·00	81·00	158·00	163·00	112·00
	Max.	1001·00	173·00	506·00	838·00	606·00
	NCN	1·00	0·00	0·00	0·00	0·00
WTR_2	MEAN	119·50	127·93	272·30	385·30	227·60
	St.Dev.	47·97	53·54	150·68	163·98	70·62
	Median	105·00	111·50	247·00	339·50	216·50
	Min.	63·00	67·00	118·00	204·00	109·00
	Max.	248·00	308·00	813·00	1001·00	388·00
	NCN	0·00	0·00	0·00	1·00	0·00
WTR_3	MEAN	125·87	161·50	243·67	381·73	243·73
	St.Dev.	166·59	115·59	121·63	166·80	94·05
	Median	89·50	123·00	205·50	346·50	229·00
	Min.	69·00	60·00	98·00	143·00	120·00
	Max.	1001·00	573·00	664·00	862·00	485·00
	NCN	1·00	0·00	0·00	0·00	0·00
WTR_4	MEAN	85·13	265·80	205·60	337·60	218·27
	St.Dev.	20·49	201·58	90·05	137·14	80·61
	Median	82·50	194·50	196·50	328·50	202·00
	Min.	57·00	53·00	92·00	171·00	99·00
	Max.	129·00	723·00	452·00	817·00	421·00
	NCN	0·00	0·00	0·00	0·00	0·00
NEW_1	MEAN	101·07	57·07	163·73	205·20	143·27
	St.Dev.	31·18	11·89	151·80	74·10	58·24
	Median	93·50	57·00	105·00	184·00	129·50
	Min.	60·00	33·00	48·00	92·00	65·00
	Max.	176·00	94·00	707·00	388·00	331·00
	NCN	0·00	0·00	0·00	0·00	0·00
NEW_2	MEAN	85·53	41·97	99·53	130·77	115·13
	St.Dev.	34·23	9·04	54·68	36·67	21·73
	Median	77·00	41·00	85·50	127·50	113·00
	Min.	41·00	25·00	48·00	75·00	86·00
	Max.	190·00	68·00	302·00	218·00	167·00
	NCN	0·00	0·00	0·00	0·00	0·00
NEW_3	MEAN	55·57	39·33	73·73	98·77	109·70
	St.Dev.	13·37	9·14	27·14	36·30	26·78
	Median	52·50	38·00	63·50	94·00	109·00
	Min.	35·00	21·00	40·00	65·00	71·00
	Max.	99·00	60·00	145·00	199·00	167·00
	NCN	0·00	0·00	0·00	0·00	0·00
NEW_4	MEAN	50·27	30·97	57·47	93·37	104·57
	St.Dev.	11·79	8·51	13·83	33·37	62·92
	Median	48·00	30·00	58·00	86·00	83·00
	Min.	29·00	18·00	32·00	47·00	53·00
	Max.	79·00	54·00	93·00	202·00	373·00
	NCN	0·00	0·00	0·00	0·00	0·00

IV. CONCLUSIONS

In this work, we have proposed a new weight initialization methodology for sigmoidal feedforward artificial neural networks. The proposed method have been compared with four uniform random weight initialization methods on five function approximation methods. The proposed method of weight initialization distributes the initial input to hidden weights in such a manner as to target the output of each individual hidden node in a different region. The comparison shows that the proposed method / routine of weight initialization is better at achieving a deeper minima during training, generalizes better and converges faster at least as compared to the four random weight initialization methods. Moreover, out of the four instances of the proposed method, the method NEW_4 has the best result. Another observation that can be made is that the order of the four instances of the proposed algorithm for weight initialization NEW[3] is as follows: $NEW_1 > NEW_2 > NEW_3 > NEW_4$; that is NEW_4 demonstrates the best result. Thus, further experimentation is required that varies the value of λ in a range above 1 also for finding whether there is any dependence on the value of the parameter λ for better weight initialization or not. Further experimentation and comparison with other methods of weight initialization as proposed in literature [25], [3], [26], [27], [28], [4], [29], [30], [5], [6], [31], [32], [7], [33], [34], [35], [36] is required before a definite statement about the efficiency and efficacy of the proposed weight initialization method can be made. Moreover, the weight initialization routines should also be compared when training algorithms other than the RPROP training algorithm is used.

TABLE VI: SUMMARY OF TRAINING SPEED RATIO FOR THE TWO CLASS OF WEIGHT INITIALIZATION METHOD.

Sr.No	Fn.	Average Speedup	Best Speedup
1.	f_1	1.69	1.97
2.	f_2	3.60	2.94
3.	f_3	3.58	2.88
4.	f_4	3.62	3.04
5.	f_5	2.09	1.88

REFERENCES

[1] S. E. Fahlman, "An empirical study of learning speed in back-propagation networks," School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, Tech. Rep. CMU-CS-88-162, September 1988.

[2] J. F. Kolen and J. B. Pollack, "Back propagation is sensitive to initial conditions," in *Proc. of the 1990 conference on Advances in neural information processing systems 3*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 860–867.

[3] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *Proc. of International Joint Conference on Neural Networks*, vol. 3, 1990, pp. 21–26.

[4] G. Drago and S. Ridella, "Statistically controlled activation weight initialization (SCAWI)," *IEEE Transactions on Neural Networks*, vol. 3, no. 4, pp. 627–631, 1992.

[5] G. Thimm and E. Fiesler, "High-order and multilayer perceptron initialization," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 349–359, 1997.

[6] Y. Yam, T. W. Chow, and C. Leung, "A new method in determining initial weights of feedforward neural networks for training enhancement," *Neurocomputing*, vol. 16, no. 1, pp. 23 – 32, 1997.

[3]The ordering is on the basis of the mean epochs required to converge to the desired goal.

[7] J. Y. F. Yam and T. W. S. Chow, "Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients." *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 430–434, 2001.

[8] W. Duch and N. Jankowski, "Survey of neural transfer functions," *Neural Computing Surveys*, vol. 2, pp. 163–212, 1999. [Online]. Available: http://www.icsi.berkeley.edu/ jagota/NCS

[9] L. Feldkamp, D. Prokhorov, and C. Eagen, "Multiple-start directed search for improved nn solution," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, July 2004, pp. 991–996 vol.2.

[10] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989.

[11] K. Funahashi, "On the approximate realization of continuous mapping by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[12] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[13] P. Chandra, "Sigmoidal function classes for feedforward artificial neural networks," *Neural Processing Letters*, vol. 18, no. 3, pp. 205–215, 2003.

[14] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, ser. LNCS:1524, G. B. Orr and K.-R. Müller, Eds. Berlin: Springer, 1998, pp. 9–50.

[15] S. Haykin, *Neural Networks: A Comprehensive Foundations*. Englewood Cliffs, NJ: Prentice–Hall, 1999.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Volume 1: Foundations*, D. E. Rumelhart, J. L. McClelland, and The PDP Research Group, Eds. Cambridge: MIT Press, 1987, pp. 318–362.

[17] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. of IEEE conference on Neural Networks*, vol. 1, San Francisco, 2010, pp. 586–591.

[18] F. Han and J.-S. Zhu, "Improved particle swarm optimization combined with backpropagation for feedforward neural networks," *International Journal of Intelligent Systems*, vol. 28, no. 3, pp. 271–288, 2013.

[19] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons from backpropagation to adaptive learning algorithms," *Computer Standards & Interfaces*, vol. 16, no. 3, pp. 265 – 278, 1994.

[20] The MathWorks Inc., "Matlab version R2013a," 2013.

[21] L. Breiman, "The PI method for estimating multivariate functions from noisy data," *Technometrics*, vol. 3, no. 2, pp. 125–160, 1991.

[22] V. Cherkassky, D. Gehring, and F. Mülier, "Comparison of adaptive methods for function estimation from samples," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 969–984, 1996.

[23] V. Cherkassky and F. Mülier, *Learning from Data – Concepts, Theory and Methods*. New York: John Wiley, 1998.

[24] M. Maechler, D. Martin, J. Schimert, M. Csoppenszky, and J. Hwang, "Projection pursuit learning networks for regression," in *Proc. of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, 1990, pp. 350–358.

[25] J. F. Shepanski, "Fast learning in artificial neural systems: multilayer perceptron training using optimal estimation," in *Proc. of IEEE International Conference on Neural Networks*, 1988, pp. 465–472 vol.1.

[26] C.-L. Chen and R. S. Nutter, "Improving the training speed of three-layer feedforward neural nets by optimal estimation of the initial weights," in *Proc. of IEEE International Joint Conference on Neural Networks*, vol. 3, 1991, pp. 2063–2068.

[27] Y. Kim and J. B. Ra, "Weight value initialization for improving training speed in the backpropagation network," in *Proc. of IEEE International Joint Conference on Neural Networks*, vol. 3, 1991, pp. 2396–2401.

[28] L. F. A. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 899–905, 1992.

[29] N. Weymaere and J.-P. Martens, "On the initialization and optimization of multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 738 – 751, 9 1994.

[30] M. Lehtokangas, J. Saarinen, K. Kaski, and P. Huuhtanen, "Initializing weights of a multilayer perceptron network by using the orthogonal least squares algorithm," *Neural Computation*, vol. 7, no. 5, pp. 982–999, Sep. 1995.

[31] L. N. de Castro, E. M. Iyoda, F. J. V. Zuben, and R. R. Gudwin, "Feedforward neural network initialization: an evolutionary approach," in *Proc. of Vth Brazilian Symposium on Neural Networks*, A. de Pdua Braga and T. B. Ludermir, Eds. IEEE Computer Society, 1998, pp. 43–48.

[32] J. Y. F. Yam and T. W. S. Chow, "A weight initialization method for improving training speed in feedforward neural network," *Neurocomputing*, vol. 30, pp. 219 – 232, 2000.

[33] M. Fernández-Redondo and C. Hernández-Espinosa, "Weight initialization methods for multilayer feedforward," in *European Symposium on Artificial Neural Networks*, 2001, pp. 119–124.

[34] X. M. Zhang, Y. Q. Chen, N. Ansari, and Y. Q. Shi, "Mini-max initialization for function approximation," *Neurocomputing*, vol. 57, pp. 389–409, 2004.

[35] M. Jamett and G. Acuña, "An interval approach for weights initialization of feedforward neural networks," in *MICAI 2006: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, A. Gelbukh and C. Reyes-Garcia, Eds. Springer Berlin Heidelberg, 2006, vol. 4293, pp. 305–315.

[36] S. Timotheou, "A novel weight initialization method for the random neural network," *Neurocomputing*, vol. 73, no. 13, pp. 160 – 168, 2009.