Asymmetric Sigmoidal Activation Function for Feed-Forward Artificial Neural Networks

Ruchi Sehrawat, Pravin Chandra, Udayan Ghose

Abstract*: Artificial neural networks of the feed – forward kind, are an established technique under the supervised learning paradigm for the solution of learning tasks. The mathematical result that allows one to assert the usefulness of this technique is that these networks can approximate any continuous function to the desired degree. The requirement imposed on these networks is to have non-linear functions of a specific kind at the hidden nodes of the network. In general, sigmoidal non-linearities, called activation functions, are generally used. In this paper we propose an asymmetric activation function. The networks using the proposed activation function are compared against those using the generally used logistic and the hyperbolic tangent activation function for the solution of 12 function approximation problems. The results obtained allow us to infer that the proposed activation function, in general, reaches deeper minima of the error measures and has better generalization error values.

Keywords* : Sigmoidal activation function, transfer function, squashing function, feed-forward artificial activation function.

I. INTRODUCTION

Artificial neural networks (ANNs) have been used to solve diverse types of learning tasks [1]. Among the different types of architectures for ANNs, the feed-forward artificial neural network (FFANN) has generated considerable interest for the solution of learning tasks in the supervised learning paradigm. The success of FFANNs to solve complex learning tasks can be (rightly) attributed to the universal approximation results for these networks (see [1-3] and the references therein). These results impose the following two conditions on the architecture of these networks:

1. There should be at least one hidden layer of nodes (called hidden nodes).
2. The output function (activation function) of these hidden nodes should not be a polynomial of the total input to these nodes.
3. There should be sufficient number of these nodes.

The most popular activation function in literature has been the functions that belong to the so-called sigmoidal class of functions which may be defined as (adapted from [4]):

Definition 1: A sigmoid is a continuous, differentiable, bounded and monotonically increasing, map s: $\Re \rightarrow \Re$, if for all $x \in \Re$ (where $\Re$ is the set of real numbers) it satisfies the following relations:

$$\lim_{x\to\infty} s(x) = \gamma \quad (1)$$

$$\lim_{x\to-\infty} s(x) = \delta \quad (2)$$

where

$$\gamma, \delta \in \Re \text{ and } \gamma > \delta \quad (3)$$

In general $\gamma = 1$ and $\delta = 0$ or $\delta = -1$, though other values satisfying eq. (1) and (2) shall also be called a sigmoidal function provided $\gamma > \delta$.

The hyperbolic tangent function, which is an anti-symmetric (odd) function and the logistic / log-sigmoid function, which is an asymmetric activation function are the two most popular activation functions satisfying the requirements of the Definition 1, in literature. The hyperbolic tangent function is defined as:

$$s_1(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4)$$

And it's derivative is:

$$\frac{ds_1(x)}{dx} = sech^2(x) = (1 - (s_1(x))^2) \quad (5)$$

The function s_1 has a range equal to (-1,1) and its domain of definition is (-∞,∞) (the set of all real numbers). Its limits for ±∞ are:

$$\lim_{x\to\infty} s_1(x) = 1 \quad (6)$$

$$\lim_{x\to-\infty} s_1(x) = -1 \quad (7)$$

And, the value of the function at x=0 is 0 (zero). The limits of the derivative of the activation function s_1 for x tending to ±∞ are:

$$\lim_{x\to\pm\infty} \frac{ds_1(x)}{dx} = 0 \quad (8)$$

And, the maximum value equal to one, of the derivative of s_1, occurs when x = 0.

Similarly, the logistic / log-sigmoid activation function is defined as:

$$s_2(x) = \frac{1}{1+e^{-x}} \quad (9)$$

And it's derivative is:

$$\frac{ds_2(x)}{dx} = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) = s_2(x)(1 - s_2(x)) \quad (10)$$

The function s_2 has a range equal to (0,1) and its domain of definition is (-∞,∞) (the set of all real numbers). Its limits for ±∞ are:

$$\lim_{x\to\infty} s_2(x) = 1 \quad (11)$$

$$\lim_{x\to-\infty} s_2(x) = 0 \quad (12)$$

And, the value of the function at x=0 is 1/2. The limits of the derivative of the activation function s_2 for x tending to ±∞ are:

$$\lim_{x\to\pm\infty} \frac{ds_2(x)}{dx} = 0 \quad (13)$$

And, the maximum value equal to ¼, of the derivative of s_2, occurs when x = 0.

These two functions variation with the independent argument is shown in Fig. 1.

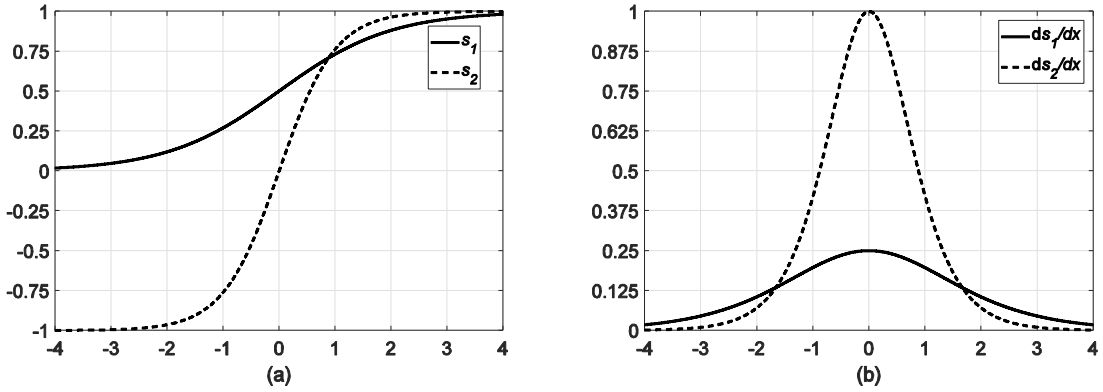

Fig. 1. (a) The logistic activation function, s_1 and the hyperbolic tangent activation function, s_2 (b) the derivatives of the activation functions s_1 and s_2.

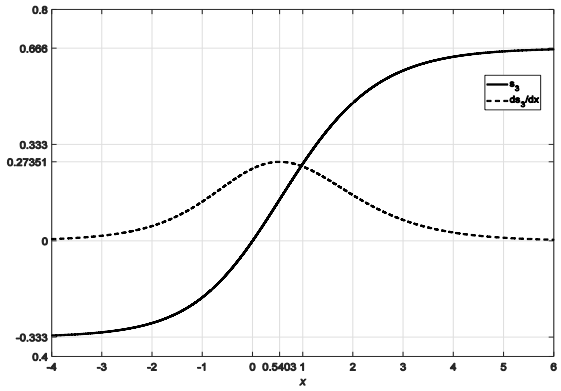

Fig. 2. The proposed activation function and its derivative.

From the Fig. 1, the continuous, bounded, monotonically increasing nature of these functions is easily seen.

The importance of these non-linear activations can be understood from the fact that with linear nodes at the hidden layers, only linear learning tasks can be solved [1-5]. The important role of the activation function in the FFANN literature has been recognized. It is these non-linear functions that allow the FFANNs to possess the universal approximation capability [2,3,5]. These mathematical results, do not provide any insight about which particular activation function is most suitable for FFANNs. This has fueled research for identification of activation functions that may provide better / faster convergence during training and better generalization capability (lessor error on data that has not been used for training / building the FFANN) [5-11]. The proposals for new activation functions have all been concentrated in the effort to propose functions whose range is either (-1,1) or (0,1) [5-11]. That is, the activation functions are either symmetric in range (equal negative and positive part of the range) or only positive definite functions with range equal to (0,1). In this paper, we propose an activation function whose range is (-1/3, 2/3). The proposed function is sigmoidal in nature and satisfies the requirement of Definition 1.

The paper is organized as follows: Section II describes the proposed activation function and discusses its satisfaction of the conditions to act as an activation function in feed-forward artificial neural networks. Section III describes the design of the experiments conducted. Section IV discusses the experimental results obtained. While the conclusions are presented in Section V.

II. Proposed Activation Function

The activation function proposed is defined as:

$$s_3(x)\begin{cases}=\frac{1}{1.5}\frac{1}{1+e^{-x}}\left(0.5+\frac{1}{1+e^{-x}}\right)-\frac{1}{3}\\=\frac{1}{1.5}s_2(x)(1+s_2(x))\ -\frac{1}{3}\end{cases} \quad (14)$$

with the derivative as:

$$\frac{ds_3(x)}{dx}\begin{cases}=\ \frac{1}{1.5}\frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right)\left(0.5+\frac{2}{1+e^{-x}}\right)\\=\frac{1}{1.5}s_2(x)(1-s_2(x))(0.5+2s_2(x))\end{cases} \quad (15)$$

The variation of the independent argument of the proposed function and its derivative is shown in Fig. 2. The properties of the proposed activation function and its derivatives may be enumerated as follows:

Property 1: The function $s_3(x)$ is continuous.

Proof: The proof follows from the continuity of the exponential function and the fact that the denominator of the terms in eq. (18) are not zero for any real value of x.

Property 2: The derivative of $s_3(x)$ is always positive.

Proof: The function $s_2(x)$, is always positive, the product $s_2(x)$ (1 - $s_2(x)$) is positive definite, hence proved.

Property 3: The function $s_3(x)$ is monotonically increasing.

Proof: The derivative of $s_3(x)$ is positive for all values of x, hence the function $s_3(x)$ is monotonically increasing.

Property 4: The value of the function $s_3(x)$ for $x = 0$ is 0.

Proof: Easily verified by substitution $x = 0$ in eq. (14).

Property 5: The function $s_3(x)$ is bounded.

Proof: The function $s_3(x)$ is monotonically increasing and its limits for $\pm\infty$ are:

$$\lim_{x\to\infty} s_3(x) = \frac{2}{3} \quad (16)$$

$$\lim_{x\to-\infty} s_3(x) = -\frac{1}{3} \quad (17)$$

Thus, the function $s_3(x)$ is bounded with a range of (-1/3, 2/3). Also for this function $\gamma = 2/3$ and $\delta = -1/3$.

Property 6: The function $s_3(x)$ is neither odd nor even or is asymmetric.

Proof: Easily seen from Fig. 2.

Property 7: The derivative of the function $s_3(x)$ is continuous.

Proof: Follows from the continuity of the function $s_2(x)$ and the addition and subtraction or multiplication by a constant of a continuous function results in a continuous function and the property that the product of two continuous functions is continuous.

Property 8: The derivative of the function $s_3(x)$ is bounded.

Proof: The extremum values of the derivative of $s_3(x)$ occurs at:

$$x = log\left(\frac{\sqrt{21}+4}{5}\right) \approx 0.5403 \quad (18)$$

Where the derivative value is ≈ 0.27351, and the limits for $\pm\infty$ are:

$$\lim_{x\to\pm\infty} \frac{ds_3(x)}{dx} = 0 \quad (19)$$

Property 9: The function $s_3(x)$ is not a polynomial.

Proof: The presence of the term e^{-x}, in the expression for $s_3(x)$ ensures that the function $s_3(x)$ cannot be expressed as a finite degree polynomial in x, and hence is not a polynomial function.

Property 10: The derivative function $s_3(x)$ is not a polynomial.

Proof: Similar to the proof of Property 9.

Thus, the proposed function is continuous, differentiable, bounded and monotonically increasing and satisfies the requirements of the Definition 1 to be a sigmoidal function with $\gamma = 2/3$ and $\delta = -1/3$.

III. EXPERIMENT DESIGN

Experiments are conducted for 12 function approximation problems. The goal of the experiments is that given a sample of the data from the input domain of the functions, and the corresponding function values at these points, find an approximation to the function using FFANNs. The approximate realization of the functions using the FFANNs should be such that even on data from the input domain, that was not used to find the approximating FFANN, the FFANNs error (*vis- à -vis* the desired actual function value) is low, or the FFANN generalizes well. This is also a short description of the supervised learning paradigm [1]. With this description, it can be understood, that all learning tasks in the supervised learning paradigm can be considered as function approximation problems. In this paper, we use 12 function approximation tasks. First these tasks are described, followed by the architecture of the FFANNs used for the solution of the learning tasks and the section is concluded with a description of the experimental methodology.

A. Learning tasks

The 12 function approximation tasks have been taken from literature [12-19]. The functions can be characterized by number of independent arguments. The tasks include one single independent argument function, 8 functions with two independent arguments, two functions with 4 independent arguments and one function with six independent arguments. All functions used have only one dependent argument. The functions shall be represented by g while the independent variable shall be represented by z.

The first two function approximation tasks are also sample functions of MatLab 2017b, while the rest 10 functions are taken from literature [13-19]. The functions are:

$$g_1(z) = \frac{1}{(z-0.3)^2+0.01} + \frac{1}{(z-0.9)^2+0.4} - 6 \quad (20)$$

where $z \in (0.1,1)$. This function in MatLab 2017b is called `humps.m`.

$$g_2(z_1,z_2) = \begin{cases} 3(1-z_1)^2 e^{-z_1^2-(1+z_2)^2} \\ -10(z_1/5 - z_1^3 - z_2^5) \\ -(1/3)e^{-z_2^2-(1+z_1)^2} \end{cases} \quad (21)$$

where $z_1, z_2 \in (-3,3)$. This function in Matlab 2017b is called `mypeaks.m`.

$$g_3(z_1,z_2) = sin(z_1\, z_2) \quad (22)$$

where $z_1, z_2 \in (-2,2)$.

$$g_4(z_1,z_2) = e^{z_1\, sin(\pi\, z_2)} \quad (23)$$

where $z_1, z_2 \in (-1,1)$.

$$g_5(z_1,z_2) = \frac{1+sin(2z_1+3z_2)}{3.5+sin(z_1-z_2)} \quad (24)$$

where $z_1, z_2 \in (-2,2)$.

$$g_6(z_1,z_2) = \begin{cases} 42.659(0.1 + \\ z_1(0.05 + z_1^4 - 10z_1^2z_2^2 + 5z_2^4)) \end{cases} \quad (25)$$

where $z_1, z_2 \in (-0.5,0.5)$.

$$g_7(z_1,z_2) = \begin{cases} 1.3356\{1.5(1-z_1) + \\ e^{2z_1-1}sin(3\pi(z_1-0.6)^2) + \\ e^{3(z_2-0.5)}sin(4\pi(z_2-0.9)^2)\} \end{cases} \quad (26)$$

where $z_1, z_2 \in (0,1)$.

$$g_8(z_1,z_2) = \begin{cases} 1.9(1.35 + e^{z_1}sin(13(z_1-0.6)^2) \\ \times e^{-z_2}sin(7z_2)) \end{cases} \quad (27)$$

where $z_1, z_2 \in (0,1)$.

$$g_9(z_1,z_2) = sin\left(2\pi\sqrt{z_1^2+z_2^2}\right) \quad (28)$$

where $z_1, z_2 \in (-1,1)$.

$$g_{10}(z_1,z_2,z_3,z_4) = e^{2z_1 sin(\pi z_4)+sin(z_2 z_3)} \quad (29)$$

where $z_1, z_2, z_3, z_4 \in (-0.25,0.25)$.

$$g_{11}(z_1,z_2,z_3,z_4) = \begin{cases} 4(z_1-0.5)(z_4-0.5) \\ \times\ sin(2\pi\sqrt{z_2^2+z_3^2}) \end{cases} \quad (30)$$

where $z_1, z_2, z_3, z_4 \in (-1,1)$.

$$g_{12}(z_1,\dots,z_6) = \begin{cases} 10sin(\pi z_1 z_2) + \\ 20(z_3-0.5)^2 + \\ 10z_4 + 5z_5 + 0z_6 \end{cases} \quad (31)$$

where $z_1, z_2, z_3, z_4, z_5, z_6 \in (-1,1)$.

For creation of the data sets for the experiments, the functions are sampled at 1000 points which are chosen from a uniform random distribution in the space of the domain of definition of the functions. The functions are evaluated at these points. These input-output pairs are the data sets used in the experiments. The first 500 pairs of these inputs-output are used for training the FFANNs and are called the training data set while the rest 500 tuples are used to find the generalization errors of the trained FFANNs, and are called the test data set.

B. FFANN Architecture

The architecture of the FFANN can be represented as layers of nodes which are connected to the previous and the succeeding layer nodes. A schematic representation is shown in Fig. 3. The leftmost layer is called the input layer, this layer receives the inputs and transfers it to the next layer nodes without processing. The connections shown in-between the inputs and the hidden layer nodes has a strength that is known as the synaptic strength or the weight of the connections. The inputs leading into a hidden node is multiplied by the corresponding weight and is summed at the node where a threshold is also applied. The weights between the kth layers ith node and the (k-1)th layers jth node is represented by w_{ij}^k, and the threshold of the kth layers ith node is represented by θ_i^k. Then the net input to the kth layers ith node can be represented as:

$$n_i^k = \sum_{j=1}^{S_{k-1}} w_{ij}^k\, o_j^{k-1} + \theta_i^k \quad (32)$$

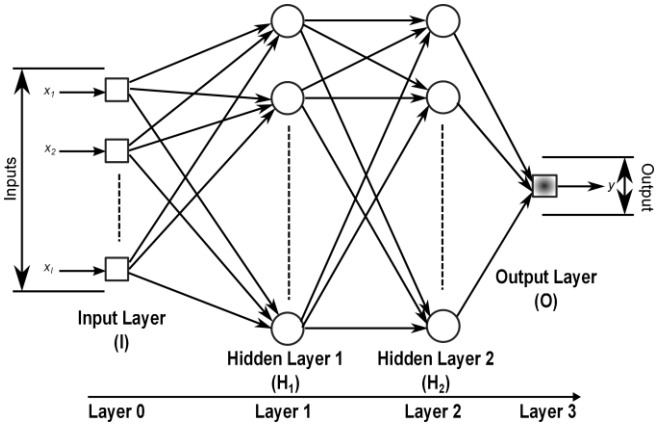

Fig. 3. The schematic diagram of a two hidden layer FFANN.

where *k* takes one of the values {1,2,3} for two hidden layer FFANNs or takes one of the values from {1,2} for single hidden layer networks, and o_j^{k-1} is the output of the *j*th node of the layer preceding the *k*th layer. The output of the *l*th layer's *p*th node can be represented as:

$$o_p^l = \begin{cases} x_p\,; \text{ the pth input if } l = 0 \text{ (input layer)} \\ s\left(n_p^l\right); \text{ the pth node of } l\text{th hidden layer)} \\ n_p^l; \text{ if } l \text{ is the output layer} \end{cases} \quad (33)$$

where *s* is one of the activation functions defined.

The networks in this paper have either one or two hidden layers. Though the universal approximation results for FFANNs only require minimum one hidden layer, experimental and theoretical results have been presented which assert that for finite sized networks, upto two hidden layer networks may be beneficial [20-25]. Therefore exploratory experiments, varying the number of nodes in the hidden layers between 1 to 50 and the number of hidden layers between 1 and 2, was conducted (in the exploratory experiments, the activation function used at the hidden layer(s) was the logistic / log-sigmoid activation function). The smallest sized networks in terms of the number of weights and thresholds, that gave a satisfactory error was chosen as the architecture to be used in the experiments. The summary of the architecture is shown in Table I, for the different function approximation tasks.

C. Experiment Methodology

Since the independent variables of the function approximation tasks as well as the dependent variables of these functions have different ranges, all variables are scaled to a common interval. The interval used in this work for scaling is [-1,1].

The training of the FFANNs on the training data set is performed by using an iterative minimization algorithm. The iterative algorithms of non-linear optimization (for minimization) require that the weights and thresholds be initialized to small random values. In this paper we initialize the weights and thresholds for all tasks to uniform random values in the range [-0.25,0.25].

The minimization algorithm used is a variant of the Resilient Backpropogation algorithm (RProp) [26,27] known as the iRProp$^+$ algorithm [28]. This algorithms' complexity has been shown to be a linearly scaling (with the number of parameters to be optimized which is the number of weights and thresholds in the specific case) algorithm [28].

Table- I: The summary of the Architecture of FFANNs. Number of output nodes in every case is one.

S. No.	Task	Input Dimension	No. of Hidden Layers	Nodes in Hidden Layers
1.	g_1	1	1	10
2.	g_2	2	2	14 and 9
3.	g_3	2	2	10 and 5
4.	g_4	2	2	8 and 5
5.	g_5	2	2	15 and 5
6.	g_6	2	2	16 and 5
7.	g_7	2	2	12 and 7
8.	g_8	2	2	15 and 7
9.	g_9	2	2	14 and 5
10.	g_{10}	4	2	8 and 3
11.	g_{11}	4	2	6 and 3
12.	g_{12}	6	2	15 and 5

The iRporp$^+$ algorithm, even though being a gradient based (first order) method has been demonstrated to be comparable, if not better than second order methods like Levenberg-Marquardt, scaled conjugate gradient etc. which estimate the Hessian of the error functional to be minimized [12].

For the comparison of the activation functions efficiency in the FFANNs for approximation tasks, we create an ensemble of 50 weights and thresholds for every learning task. And on the architectures decided in Section IIIB or Table I, at the hidden layer nodes, the two established activation functions are used as well as the proposed activation function is used.

Thus, for a specific function approximation task, for every activation function we measure errors over the training data set and the test data set on completion of 2000 epochs or iteration of the training algorithm. As a result we obtain 50 values of the training and the generalization errors over the weight ensemble corresponding to the task.

It is the statistics over these ensemble values that constitute the statistics measured for comparison of the activation functions. Similarly, we can define the generalization error ensembles as the ensemble of 50 values of the mean squared errors over the test data set for networks using the three activation functions.

The mean squared error values for the ensemble corresponding / identified by the three activation functions are analysed and the results are reported.

The statistics that we report are the mean / average of the ensemble (mean squared errors) over both the training data set (MMSE of training) and the test data set (MMSE of generalization). We report the standard deviation of the ensemble errors (represented by STD) over both the training data set and the test data set. We report the minimum value of the ensemble error (represented as MiMSE) over both the training data set and the test data set. We report the maximum value of the ensemble error (represented as MaMSE) over both the training data set and the test data set. MiMSE and MaMSE difference and the STD values are an assessment of the spread of the errors while MMSE is a measure of the central tendency as the expected value of the errors. In contrast to the mean / average, the median is considered to be less sensitive to outlier values and is considered a robust measure of central tendency [29]. Thus, we also report the ensemble error's median value over both the training data set as well as the test data set (MeMSE).

Table- II: Training data set summary. All values $\times 10^{-3}$.

Sr.No.	Task	Statistics	s_1	s_2	s_3
1.	g_1	MMSE	0.3935	0.25259	0.4477
		STD	0.42672	0.49655	1.69064
		MiMSE	0.02917	0.00279	0.01892
		MaMSE	1.72795	1.6750	11.94084
		MeMSE	0.2046	0.02365	0.10008
2.	g_2	MMSE	0.26962	0.18924	0.23957
		STD	0.13545	0.15608	0.11665
		MiMSE	0.0906	0.0579	0.07469
		MaMSE	0.73091	0.91042	0.54591
		MeMSE	0.23122	0.14861	0.22103
3.	g_3	MMSE	1.65844	2.8548	1.06201
		STD	2.38629	2.8920	0.79714
		MiMSE	0.20803	0.27527	0.24714
		MaMSE	11.92121	10.3425	3.19118
		MeMSE	0.92233	1.6093	0.75443
4.	g_4	MMSE	0.24543	0.27827	0.2416
		STD	0.14605	0.18227	0.17901
		MiMSE	0.06127	0.05771	0.0651
		MaMSE	0.89825	0.94827	0.85875
		MeMSE	0.18605	0.23106	0.18356
5.	g_5	MMSE	4.9327	2.69143	3.39203
		STD	7.93962	4.12645	5.1395
		MiMSE	0.44849	0.25576	0.56495
		MaMSE	34.83707	19.83857	34.48774
		MeMSE	2.28995	1.29585	1.79155
6.	g_6	MMSE	0.61728	0.63938	0.67811
		STD	0.34069	0.51557	0.30789
		MiMSE	0.11805	0.11747	0.05852
		MaMSE	2.06565	3.03118	1.20581
		MeMSE	0.58381	0.52961	0.65944
7.	g_7	MMSE	0.26278	0.57359	0.42167
		STD	0.222	0.40931	0.38288
		MiMSE	0.03366	0.05949	0.05221
		MaMSE	1.16457	1.75666	1.53342
		MeMSE	0.18888	0.3889	0.28401
8.	g_8	MMSE	0.37623	0.53034	0.39412
		STD	0.18883	0.61089	0.20394
		MiMSE	0.09203	0.08787	0.13188
		MaMSE	1.07102	3.34983	0.92412
		MeMSE	0.33887	0.35046	0.33873
9.	g_9	MMSE	4.51539	3.26388	9.34649
		STD	3.45084	2.27786	16.8135
		MiMSE	1.48761	1.14418	2.03864
		MaMSE	23.5513	13.63053	109.6422
		MeMSE	3.67045	2.42486	3.86256
10.	g_{10}	MMSE	0.17653	1.88877	0.09528
		STD	0.1898	6.50085	0.11401
		MiMSE	0.02758	0.0533	0.02335
		MaMSE	0.73724	32.29336	0.50327
		MeMSE	0.07153	0.43499	0.05733
11.	g_{11}	MMSE	10.91603	13.40552	11.10894
		STD	2.63467	3.28105	2.60393
		MiMSE	7.29785	7.56883	6.44409
		MaMSE	18.3686	19.26698	17.62098
		MeMSE	10.47797	13.54886	11.12072
12.	g_{12}	MMSE	0.18587	0.19999	0.19909
		STD	0.05533	0.06843	0.06479
		MiMSE	0.10053	0.09053	0.0947
		MaMSE	0.30679	0.3811	0.43013
		MeMSE	0.17834	0.1993	0.18945

The comparison of these statistics for the FFANNs using the different activation functions (s_1, s_2 or s_3) shall allow us to assess which activation function leads to lower values of the training and/or generalization errors across the function approximation tasks. But to assess whether these differences are statistically significant or not we use the one sided Student's t-test (to check for the difference in means) and the Wilcoxon rank-sum test (to check for the difference in medians) [30]. Both these tests are conducted at a level of significance of 0.1. And, the tests are conducted for the ensemble errors over both the training data set and the test data set.

Table- III: The comparison matrix for the training data set on the basis of the mean of the ensemble errors (MMSE).

Activations	s_1	s_2	s_3
s_1	--	4	5
s_2	8	--	7
s_3	7	5	--

Table- IV: The comparison matrix for the training data set on the basis of the median of the ensemble errors (MeMSE).

Activations	s_1	s_2	s_3
s_1	--	5	5
s_2	7	--	5
s_3	7	7	--

IV. RESULTS

The summary of the statistics for the training data set is shown in Table II. Table III and IV represent in how many cases / tasks, the row label activation function was better than (had a lower value than) the column label activation function, on the basis of the mean (MMSE) and median (MeMSE), respectively, over the training data set (Tables like this shall be called the comparison tables). From the Tables II to IV, it is clear that the activation function s_1 outperforms s_2, s_3 outperforms s_2 and on the basis of MMSE comparison, for the training set data, though s_1 is better than s_3 for 7 cases out of 12, but a closer inspection of the Table II, MaMSE values shows the presence of instances where the networks may not have converged and the same is reflected when the comparison is done on the basis of the median (MeMSE) whereas 7 cases s_3 is better than s_2 out of 12 tasks. Thus, on comparison on the basis of the statistics reflected in Table II, III and IV, for the training set data, we may assert that s_3 is better than s_2 and s_1, while s_2 is better than s_1.

The actual test of the success of any modelling mechanism is on the basis of its generalization performance. The model that has the least generalization error is to be preferred [16]. Table V shows the summary statistics for the test data set (the generalization error statistics). The comparison matrix on the basis of the means and medians of the ensemble errors are shown in Table VI and VII respectively, for the test data set. These tables also allow us to assert that some outliers are present (leading to the difference in Table VI and VII). On the basis of the results obtained, we can assert that s_3 has the best performance behavior as compared to s_1 and s_2 while s_1 was better than s_2.

The comparison matrix on the basis of the one sided Student's t-test reflects in how many tasks the row label activation function is better (in mean) than the column label activation function at a significance level of 0.10. The training data set comparison matrix is shown in Table VIII while the test data set comparison matrix, in this case is shown in Table IX. From these tables we see that s_2 is the worst while s_1 and s_3 are comparable. Bust, since this analysis is for the means of the ensemble errors, it may be affected by outliers.

The comparison matrix based on the one sided Wilcoxon rank-sum test at a significance level of 0.10, are shown in Table X and XI for the training and the test data set, respectively. And, from these tables, especially on the basis of the test data set results, we may infer that s_3 is better than both s_1 and s_2 while s_1 is better than s_2.

Table- V: Test data set summary. All values $\times 10^{-3}$.

Sr.No.	Task	Statistics	s_1	s_2	s_3
1.	g_1	MMSE	0.38245	0.25155	0.42915
		STD	0.42202	0.49461	1.58054
		MiMSE	0.02741	0.00272	0.01933
		MaMSE	1.73684	1.67898	11.1514
		MeMSE	0.19514	0.02481	0.0958
2.	g_2	MMSE	0.70976	0.8097	0.58746
		STD	0.64084	0.74644	0.32918
		MiMSE	0.20404	0.20042	0.15729
		MaMSE	3.71538	4.7524	1.62946
		MeMSE	0.53997	0.58312	0.51948
3.	g_3	MMSE	3.12629	5.24423	3.0096
		STD	2.74022	3.6136	2.38073
		MiMSE	0.62601	0.53708	0.32348
		MaMSE	12.20692	17.30332	9.53182
		MeMSE	2.14691	4.12658	2.24178
4.	g_4	MMSE	0.34888	0.42744	0.33672
		STD	0.23636	0.28834	0.27598
		MiMSE	0.09207	0.08766	0.08733
		MaMSE	1.44831	1.27413	1.36204
		MeMSE	0.25213	0.31578	0.24509
5.	g_5	MMSE	6.0612	5.26068	4.6369
		STD	8.38447	8.0013	6.15703
		MiMSE	0.5703	0.44555	0.82604
		MaMSE	35.81627	41.08461	39.60208
		MeMSE	3.30586	2.83964	3.01497
6.	g_6	MMSE	0.60992	0.87618	0.69666
		STD	0.29661	0.7089	0.29989
		MiMSE	0.15126	0.15989	0.05835
		MaMSE	1.77546	4.4163	1.23178
		MeMSE	0.5576	0.69252	0.68615
7.	g_7	MMSE	0.32042	0.8306	0.48834
		STD	0.2432	0.62243	0.39922
		MiMSE	0.04829	0.09131	0.07244
		MaMSE	1.31669	2.88539	1.6696
		MeMSE	0.24817	0.57718	0.34562
8.	g_8	MMSE	0.54236	1.0115	0.58049
		STD	0.26197	0.93937	0.25325
		MiMSE	0.12609	0.25628	0.15664
		MaMSE	1.76662	4.91325	1.18934
		MeMSE	0.51364	0.68361	0.50182
9.	g_9	MMSE	7.03208	108.6311	12.4308
		STD	4.46711	719.4845	18.31369
		MiMSE	2.78493	2.70616	3.51171
		MaMSE	29.4174	5094.282	117.6235
		MeMSE	6.23704	5.60803	5.98271
10.	g_{10}	MMSE	0.24716	2.28422	0.13044
		STD	0.28732	7.28173	0.14402
		MiMSE	0.0443	0.10684	0.02949
		MaMSE	1.53559	34.61244	0.67934
		MeMSE	0.11096	0.54844	0.08443
11.	g_{11}	MMSE	24.00049	41.09779	22.40957
		STD	6.43133	114.6675	5.87722
		MiMSE	12.73733	18.8069	13.92489
		MaMSE	46.66714	834.8001	42.78993
		MeMSE	24.36186	23.99648	20.59418
12.	g_{12}	MMSE	0.45546	0.4916	0.44265
		STD	0.14711	0.15516	0.1405
		MiMSE	0.23784	0.22954	0.19324
		MaMSE	0.8985	0.95488	0.75658
		MeMSE	0.42751	0.4705	0.41358

Table- VI: The comparison matrix for the test data set on the basis of the mean of the ensemble errors (MMSE).

Activations	s_1	s_2	s_3
s_1	--	2	1
s_2	10	--	5
s_3	11	7	--

Table- VII: The comparison matrix for the test data set on the basis of the median of the ensemble errors (MeMSE).

Activations	s_1	s_2	s_3
s_1	--	4	3
s_2	8	--	3
s_3	9	9	--

Table- VIII: The comparison matrix for the train data set on the basis of the one sided Student's t-test.

Activations	s_1	s_2	s_3
s_1	--	4	2
s_2	5	--	2
s_3	5	2	--

Table- IX: The comparison matrix for the test data set on the basis of the one sided Student's t-test.

Activations	s_1	s_2	s_3
s_1	--	1	0
s_2	6	--	3
s_3	8	2	--

Table- X: The comparison matrix for the train data set on the basis of the one sided Wilcoxon rank-sum test.

Activations	s_1	s_2	s_3
s_1	--	4	5
s_2	4	--	2
s_3	5	2	--

Table- IX: The comparison matrix for the test data set on the basis of the one sided Wilcoxon rank-sum test.

Activations	s_1	s_2	s_3
s_1	--	2	2
s_2	7	--	2
s_3	8	3	--

V. CONCLUSION

In this paper we have proposed a new sigmoidal function that is asymmetric and has a skewed derivative. The maxima of the derivative of the proposed activation function does not occur at the input value of zero, thus this helps in elevating the vanishing weight update problem. The properties of the proposed activation functions were analyzed and it was demonstrated that the function is suitable for the use as an activation function in FFANNs. The proposed activation function satisfies the requirement to be a sigmoidal function (as per Definition 1) and satisfies the conditions of the universal approximation conditions imposed on the activation function. On a learning task suite of 12 function approximation problems, the proposed activation function s_3 was compared with the hyperbolic tangent function (s_1) and with the logistic / log-sigmoid function (s_2), when used as activation functions at the hidden layers of feed-forward artificial neural networks. The results obtained indicated that the proposed activation is in general better in performance than the logistic as well as the hyperbolic tangent functions.

REFERENCES

1. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Inc., New Jersey, 1999.
2. A. Pinkus, Approximation theory of the MLP model in neural networks, *Acta Numerica*, vol. 8, 1999, pp. 143–195.

3. M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a non-polynomial activation function can approximate any function, *Neural Networks*, vol. 6, 1993, pp. 861–867.
4. P. Chandra and Y. Singh, Feedforward sigmoidal networks – equicontinuity and fault–tolerance, IEEE Transactions on Neural Networks, vol. 15, no. 6, 2004, pp. 1350–1366.
5. W. Duch and N. Jankowski, Survey of neural network transfer functions, Neural Computing Surveys, vol. 2, 1999, pp. 163–212.
6. D. Elliott, A better activation function for artificial neural networks, *Tech. Rep. ISR Technical Report TR 93-8*, Institute for Systems Research, University of Maryland, College Park, MD 20742, 1993.
7. S. S. Sodhi and P. Chandra, Bi-modal derivative activation function for sigmoidal feedforward networks, *Neurocomputing*, vol. 143, 2014, pp. 182 – 196.
8. Y. Singh and P. Chandra, A class + 1 sigmoidal activation functions for FFANNs, *Journal of Economic Dynamics and Control*, vol. 28, 2003, pp. 183–187.
9. P. Chandra, Sigmoidal function classes for feedforward artificial neural networks, *Neural Processing Letters*, vol. 18, no. 3, 2003, pp. 205–215.
10. G. Gomes, T. Ludermir, and L. Lima, Comparison of new activation functions in neural network for forecasting financial time series, *Neural Computing and Applications*, vol. 20, 2011, pp. 417–439.
11. P. Chandra and S. Sodhi, A skewed derivative activation function for SFFANNs, in *Recent Advances and Innovations in Engineering (ICRAIE)*, May 2014, pp. 1–6.
12. S. S. Sodhi and P. Chandra, Bi-modal derivative activation function for sigmoidal feedforward networks, *Neurocomputing*, vol. 143, 2014, pp. 182 – 196.
13. A. Mishra, P. Chandra, U. Ghose, and S. S. Sodhi, Bi-modal derivative adaptive activation function sigmoidal feedforward artificial neural networks, *Applied Soft Computing*, vol. 61, no. Supplement C, 2017, pp. 983 – 994.
14. L. Breiman, The PI method for estimating multivariate functions from noisy data, *Technometrics*, vol. 3, no. 2, 1991, pp. 125–160.
15. V. Cherkassky, D. Gehring, and F.M¨ulier, Comparison of adaptive methods for function estimation from samples, *IEEE Transactions on Neural Networks*, vol. 7, no. 4, 1996, pp. 969–984.
16. V. Cherkassky and F. M¨ulier, *Learning from Data – Concepts, Theory and Methods*. New York: John Wiley,1998.
17. 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.
18. V. Cherkassky, Y. Lee, and H. Lari-Najafi, Self organizing network for regression: efficient implementation and comparative evaluation," in *IJCNN-91- Seattle International Joint Conference on Neural Networks*,vol. i, July 1991, pp. 79–84.
19. J. H. Friedman, Multivariate adaptive regression splines, *Ann. Statist*, vol. 19, 1991, pp. 1–141.
20. G. Brightwell, C. Kenyon, and H. Paugam-Moisy, Multilayer neural networks: One or two hidden layers?, in *Advances in Neural Information Processing Systems 9* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), MIT Press, 1997, pp. 148–154.
21. S. Tamura and M. Tateishi, Capabilities of a four-layered feedforward neural network: four layers versus three, *IEEE Transactions on Neural Networks*, vol. 8, 1997, pp. 251–255.
22. G.-B. Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, *IEEE Transactions on Neural Networks*, vol. 14, 2003, pp. 274–281.
23. T. Nakama, Comparisons of single- and multiple-hidden-layer neural networks, in *Advances in Neural Networks* { ISNN 2011 (D. Liu, H. Zhang, M. Polycarpou, C. Alippi, and H. He, eds.), Springer Berlin Heidelberg, 2011, pp. 270 – 279.
24. V. Kurkova and M. Sanguineti, Can two hidden layers make a difference?, in *Adaptive and Natural Computing Algorithms* (M. Tomassini, A. Antonioni, F. Daolio, and P. Buesser, eds.), Springer Berlin Heidelberg, 2013, pp. 30–39,
25. A. Thomas, M. Petridis, S. Walters, S. Gheytassi, and R. Morgan, Two hidden layers are usually better than one, in *Proc. of International Conference on Engineering Applications of Neural Networks* (G. Boracchi, L. Iliadis, C. Jayne, and A. Likas, eds.), Athens (Greece), Springer, 25-27 August 2017 pp. 279–290.
26. 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.
27. M. Riedmiller, Advanced supervised learning in multilayer perceptrons from backpropagation to adaptive learning algorithms, *Computer Standards & Interfaces*, vol. 16, no. 3, 1994, pp. 265 – 278.
28. C. Igel and M. H¨usken, Empirical evaluation of the improved RProp learning algorithms, *Neurocomputing*, vol. 50, 2003, pp. 105 – 123.
29. T. Hettmansperger and J. McKean, *Robust Nonparametric Statistical Methods*. Kendall's Library of Statistics: An Arnold Publication No. 5, Arnold, 1998.
30. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*, pp. 977–979. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.
31. A
32. T. Hettmansperger and J. McKean, *Robust Nonparametric Statistical Methods*. Kendall's Library of Statistics: An Arnold Publication No. 5, Arnold, 1998.
33. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*, pp. 977–979. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.
34. 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.
35. M. Riedmiller, Advanced supervised learning in multilayer perceptrons from backpropagation to adaptive learning algorithms, *Computer Standards & Interfaces*, vol. 16, no. 3, 1994, pp. 265 – 278.
36. C. Igel and M. H¨usken, Empirical evaluation of the improved RProp learning algorithms, *Neurocomputing*, vol. 50, 2003, pp. 105 – 123.
37. Y. LeCun, L. Bottou, G. B. Orr and K.-R. Muller, Efficient BackProp, in *Neural Networks: Tricks of the Trade*, G. B. Orr and K.-R. Muller (Eds.), Berlin: Springer, 1998, pp. 9–50.