A Non-Polynomial, Non-Sigmoidal, Bounded and Symmetric Activation Function for Feed – Forward Artificial Neural Networks

Apoorvi Sood, Pravin Chandra, Udayan Ghose

Abstract*: Feed-forward artificial neural networks are universal approximators of continuous functions. This property enables the use of these networks to solve learning tasks. Learning tasks in this paradigm are cast as function approximation problems. The universal approximation results for these networks require at least one hidden layer with non-linear nodes, and also require that the non-linearities be non-polynomial in nature. In this paper a non-polynomial and non-sigmoidal non-linear function is proposed as a suitable activation function for these networks. The usefulness of the proposed activation function is shown on 12 function approximation task. The obtained results demonstrate that the proposed activation function outperforms the logistic / log-sigmoid and the hyperbolic tangent activation functions.

Keywords* : Activation function, transfer function, squashing function, feed-forward artificial neural networks.

I. INTRODUCTION

Feed-forward artificial neural networks (FFANNs) have been demonstrated to be a method for the solution of complex learning tasks (see [1] for a comprehensive introduction). The justification for the usage of the FFANNs to solve a particular learning task can be attributed to the possession of the Universal Approximation Property (UAP) by these networks (see [2] for a survey of these results). The UAP for FFANNs assert that a network with one hidden layer of non-linear nodes is capable of approximating any continuous function to any desired degree of approximation, provided that the hidden layer has sufficient number of hidden nodes. The condition on the non-linearities at the hidden nodes is that the function should not be a polynomial [2,3]. These non-linear functions are also known as activation functions, transfer functions or squashing functions. The class of *S* shaped functions also known as sigmoidal functions have been used mostly as activation functions in the reported literature. These functions are also not a polynomial of the independent argument. A sigmoidal function can be defined as [4]:

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

$$\lim_{x\to\infty} \sigma(x) = \alpha \quad (1)$$

$$\lim_{x\to-\infty} \sigma(x) = \beta \quad (2)$$

where

$$\alpha, \beta \in \Re \text{ and } \alpha > \beta \quad (3)$$

In general $\alpha = 1$ and $\beta = 0$ or $\beta = -1$

The generally used sigmoidal functions are the logistic or the log-sigmoid function defined as:

$$\sigma_1(x) = \frac{1}{1+e^{-x}} \quad (4)$$

with the derivative:

$$\frac{d\sigma_1(x)}{dx} = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) = \sigma_1(x)\,(1-\sigma_1(x)) \quad (5)$$

While the other activation function generally used is the hyperbolic tangent function defined as:

$$\sigma_2(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = tanh(x) \quad (6)$$

with the derivative:

$$\frac{d\sigma_2(x)}{dx} = (1-(\sigma_2(x))^2) = sech^2(x) \quad (7)$$

These functions cannot be represented as a sum of finite number of powers of the independent variables, that is, these functions are non-polynomial in nature. The variation of these functions and their derivatives is shown in Fig. 1.

These activation functions are continuous, bounded, monotonically increasing and are differentiable and satisfy the requirements of the Definition 1. The sigmoidal functions have been the activations of choice in the feed-forward neural network literature [1]. Though, the UAPs for these networks do not require that the activation function be sigmoidal, not much attention has been given to activations which are non-sigmoidal [5,6]. It has been said that in the limit of infinite number of nodes at the hidden layer, all activations may be treated as equivalent, but for finite sized networks, activations play an important role in the speed of convergence during training of the networks and also for the generalization error [7]. In this paper, we propose the usage of a non-sigmoidal function that is also not a polynomial, as an activation function at the hidden layer nodes of a feed-forward activation function.

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.

Revised Manuscript Received on October 05, 2019.

***** Correspondence Author

Apoorvi Sood*, Research Scholar, University School of Information Communication and Technology, Guru Gobind Singh Indrapratha University, N.Delhi, India and Assistant Professor, Netaji Subhash University of Technology, N. Delhi, India. Email: soodapoorvi@yahoo.com

Pravin Chandra, University School of Information Communication and Technology, Guru Gobind Singh Indrapratha University, N.Delhi, India. Email: chandra.pravin@gmail.com, pchandra@ipu.ac.in.

Udayan Ghose, University School of Information Communication and Technology, Guru Gobind Singh Indrapratha University, N.Delhi, India. Email: udayan@ipu.ac.in.

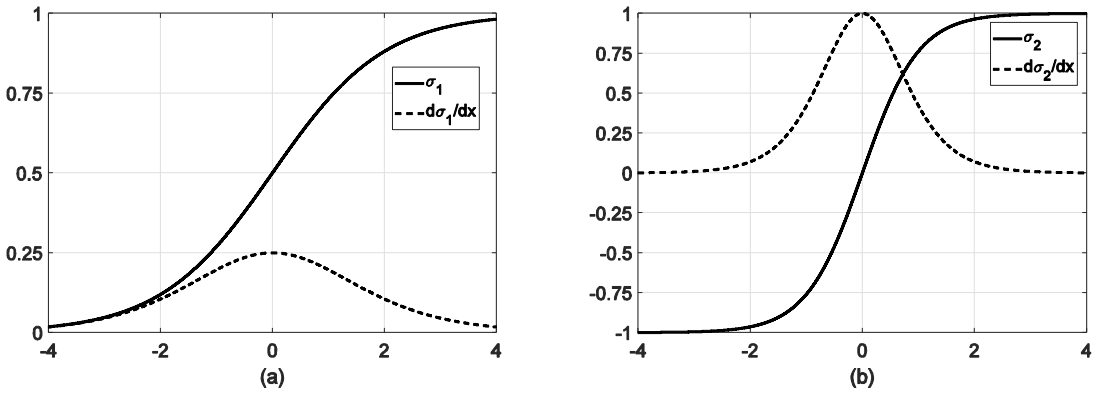

Fig. 1. (a) The logistic activation function (4) and its derivative, (b) the hyperbolic activation function (6) and its derivative.

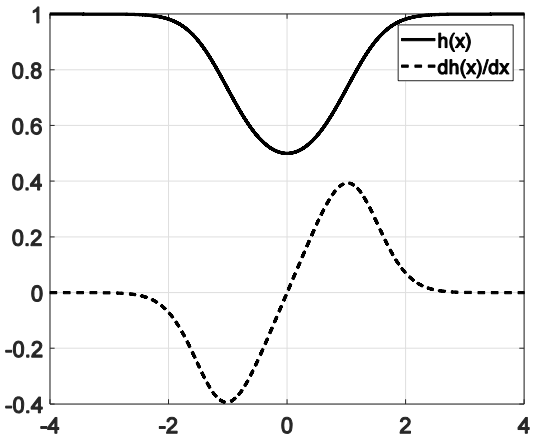

Fig. 2. The proposed activation function and its derivative.

II. PROPOSED ACTIVATION FUNCTION

The activation function proposed is defined as:

$$h(x) = \frac{1}{1+e^{-x^2}} \quad (8)$$

with the derivative as:

$$\frac{dh(x)}{dx} = 2x\left(\frac{1}{1+e^{-x^2}}\right)\left(1-\frac{1}{1+e^{-x^2}}\right) = 2xh(x)(1-h(x)) \quad (9)$$

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 $h(x)$ is continuous.

Proof: The proof follows from the continuity of the exponential function and the fact that the denominator of the eq. (8) is not zero for any real value of x.

Property 2: The function $h(x)$ is bounded.

Proof: The extremum values of the function are at ±∞ and at zero. The maximum occurs for the limits to ±∞ and the value is 1 while the minimum of the function occurs at zero where the value of the function is ½. It is also asserted that the function is non-monotonic in nature.

Property 3: The function $h(x)$ is symmetric about the line x=0 or it is an even function.

Proof: Easily verified by substituting $-x$ for x.

Property 4: The function $h(x)$ is not a sigmoidal function.

Proof: Since by Property 2, the function is not monotonic, hence it does not satisfy the requirements of the Definition 1.

Property 5: The derivative of the function $h(x)$ is continuous.

Proof: follows from the continuity of the function $h(x)$ and the linear function and the property that the product of two continuous functions is continuous.

Property 6: The derivative of the function $h(x)$ is bounded.

Proof: The extremum values of the derivative of $h(x)$ occurs at $x = \pm 1.021581$. The maxima is at the +ve x value while the minima occurs at the –ve x value, also these points correspond to the points of inflection of the function $h(x)$. The maximum/minimum values of the derivative are ±0.39355.

Property 7: The derivative of the function $h(x)$ is an odd function.

Proof: Easily verified by substituting $-x$ for x.

III. EXPERIMENT DESIGN

We first describe the learning tasks used in the experiments, followed by the architecture of the FFANNs used in the experiments, and the section is concluded by the description of the experiment methodology.

A. Learning tasks

Feed-forward artificial neural networks implement the supervised learning paradigm [1]. In the supervised learning paradigm, the network is trained to minimize the errors on a set of data points. These data points can be thought of as being produced by the sampling of the input domain of a function and the function value at these points (in general considered as random samples). Thus, all supervised learning tasks can be considered as function approximation tasks.

In this paper, we have used 12 function approximation tasks to verify the veracity of the networks using the proposed activation function as compared to the logistic function and the hyperbolic tangent function. The first two function approximation tasks are taken from the sample functions of MatLab 2017b, while the rest 10 functions are taken from literature [8-13]. The functions are:

$$f_1(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.4} - 6 \quad (10)$$

where $x \epsilon (0.1,1)$.

$$f_2(x,y) = \begin{cases} 3(1-x)^2 e^{-x^2-(1+y)^2} \\ -10(x/5 - x^3 - y^5) \\ -(1/3)e^{-y^2-(1+x)^2} \end{cases} \quad (11)$$

where $x, y \epsilon (-3,3)$.

$$f_3(x,y) = sin(xy) \quad (12)$$

where $x, y \epsilon (-2,2)$.

$$f_4(x,y) = e^{x\,sin(\pi y)} \quad (13)$$

where $x, y \epsilon (-1,1)$.

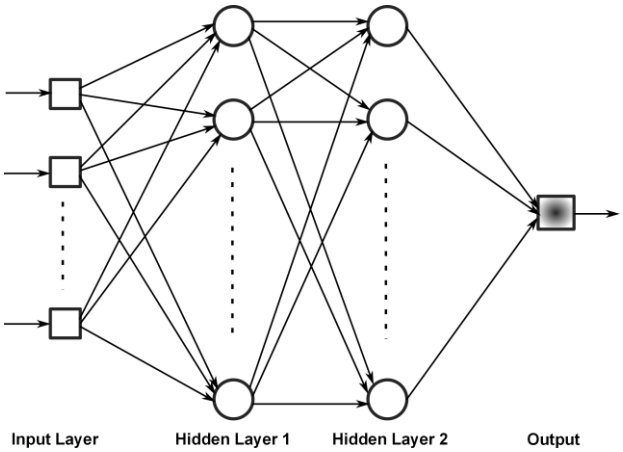

Fig. 3. The schematic diagram of a two hidden layer FFANN.

$$f_5(x,y) = \frac{1+sin(2x+3y)}{3.5+sin(x-y)} \quad (14)$$

where $x, y \epsilon (-2,2)$.

$$f_6(x,y) = \begin{cases} 42.659(0.1 + \\ x(0.05 + x^4 - 10x^2y^2 + 5y^4)) \end{cases} \quad (15)$$

where $x, y \epsilon (-0.5,0.5)$.

$$f_7(x,y) = \begin{cases} 1.3356\{1.5(1-x) + \\ e^{2x-1}sin(3\pi(x-0.6)^2) + \\ e^{3(y-0.5)}sin(4\pi(y-0.9)^2)\} \end{cases} \quad (16)$$

where $x, y \epsilon (0,1)$.

$$f_8(x,y) = \begin{cases} 1.9(1.35 + \\ e^x sin(13(x-0.6)^2)e^{-y}sin(7y)) \end{cases} \quad (17)$$

where $x, y \epsilon (0,1)$.

$$f_9(x,y) = sin\left(2\pi\sqrt{x^2+y^2}\right) \quad (18)$$

where $x, y \epsilon (-1,1)$.

$$f_{10}(x_1,x_2,x_3,x_4) = e^{2x_1 sin(\pi x_4) + sin(x_2 x_3)} \quad (19)$$

where $x_1, x_2, x_3, x_4 \epsilon (-0.25,0.25)$.

$$f_{11}(x_1,x_2,x_3,x_4) = \begin{cases} 4(x_1-0.5)(x_4-0.5) \\ \times\ sin(2\pi\sqrt{x_2^2+x_3^2}) \end{cases} \quad (20)$$

where $x_1, x_2, x_3, x_4 \epsilon (-1,1)$.

$$f_{12}(x_1,\dots,x_6) = \begin{cases} 10sin(\pi x_1 x_2) + \\ 20(x_3-0.5)^2 + \\ 10x_4 + 5x_5 + 0x_6 \end{cases} \quad (21)$$

where $x_1, x_2, x_3, x_4, x_5, x_6 \epsilon (-1,1)$.

The domain of definition of the functions are sampled at 1000 discrete random points which are chosen from a uniform distribution. These input points together with the function values at the input points constitute the data set for the experiments. Out of these 1000 data points for each of the function approximation problems, 500 are used to train the networks and are called the training data set while the other set of 500 data points shall be used to find the generalization error of the trained FFANNs and shall be called the test data set.

B. FFANN Architecture

The schematic architecture of a two hidden layer FFANN is shown in Fig.3. The networks used for the solution of the function approximation tasks was decided by exploratory experiments in which the number of layers was varied between one and two while the number of hidden nodes in the network was varied between 1 and 50. The smallest sized network that gave a satisfactory error was selected for the experiments. The number of inputs to the network and the number of outputs is determined by the learning task. Table I summarizes the architecture of the FFANNs used for a specific task.

Table- I: The architectural summary of the FFANNs used. I is the number of inputs, NHL is the number of hidden layers, H is the number of nodes in the hidden layer while O is the output layer size

S.No.	Task	I	NHL	H	O
1.	f_1	1	1	(10)	1
2.	f_2	2	2	(14,9)	1
3.	f_3	2	2	(10,5)	1
4.	f_4	2	2	(8,5)	1
5.	f_5	2	2	(15,5)	1
6.	f_6	2	2	(16,5)	1
7.	f_7	2	2	(12,7)	1
8.	f_8	2	2	(15,7)	1
9.	f_9	2	2	(14,5)	1
10.	f_{10}	4	2	(8,3)	1
11.	f_{11}	4	2	(6,3)	1
12.	f_{12}	6	2	(15,5)	1

C. Experiment Methodology

All the data set variables, for a specific learning task, are scaled to the interval [-1,1]. All experiments are conducted using the scaled variables. And, the results are reported over the scaled variables.

For each learning task, an ensemble set of 50 weights and thresholds is created. All weights and thresholds are chosen as uniform random numbers in the interval [-0.25,0.25]. For each task, the FFANNs use the logistic activation, the hyperbolic tangent activation function and the proposed activation functions at the hidden layer nodes. Thus, in all 50 × 3 × 12 =1800 networks are trained.

For the trained network we measure the performance error (or, the mean squared error over the training data set). This is an ensemble of 50 training data set mean squared errors for FFANN using one of the activation functions at the hidden nodes and for one specific task. It is the statistics over these ensemble values that constitute the training data set statistics. Similarly, we can define the generalization error ensembles as the ensemble of 50 values of the mean squared errors over the test data set.

We report the ensemble mean (MMSE), the ensemble standard deviation (STD), the minimum of the ensemble (MiMSE), and recognizing that the median is a more robust estimator of the central tendency [14]. We also report the ensemble median (MeMSE). This is done for both the training data set error ensembles and the test data set error ensembles.

To compare whether the mean and medians obtained for the different activation function using FFANNS are statistically different or not, we use the Student's t-test and the Wilcoxon rank-sum test [14,15]. The one-sided version of the tests are used at the significance level of 0.05, to check whether the mean / median of the ensemble's mean squared errors for a particular activation function are significantly lesser or not as compared to another

activation function.

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

Sr.No.	Task	Statistics	Activation Functions		
			σ_1	σ_2	$h(x)$
1.	f_1	MMSE	0.32027	0.45464	0.24865
		STD	0.54472	0.54951	0.68867
		MiMSE	0.0032	0.02608	0.00089
		MeMSE	0.02743	0.21653	0.01032
2.	f_2	MMSE	0.22754	0.26158	0.16089
		STD	0.20386	0.16039	0.09447
		MiMSE	0.05257	0.0944	0.05902
		MeMSE	0.18184	0.20379	0.13881
3.	f_3	MMSE	2.81901	1.55823	1.26216
		STD	3.38019	2.65821	1.30102
		MiMSE	0.21747	0.19724	0.25102
		MeMSE	1.28824	0.86274	0.76342
4.	f_4	MMSE	1.25994	0.20675	0.22294
		STD	6.54638	0.1049	0.08907
		MiMSE	0.06488	0.05792	0.10774
		MeMSE	0.23147	0.17495	0.20144
5.	f_5	MMSE	1.76848	3.44335	1.066
		STD	2.35718	5.16153	0.86481
		MiMSE	0.21442	0.46429	0.12805
		MeMSE	1.2078	2.20869	0.87084
6.	f_6	MMSE	0.74044	0.59276	0.42589
		STD	0.65322	0.33304	0.30602
		MiMSE	0.10773	0.10248	0.15435
		MeMSE	0.5862	0.54087	0.31628
7.	f_7	MMSE	0.5799	0.21907	0.7439
		STD	0.35473	0.19595	0.50276
		MiMSE	0.06115	0.02486	0.06602
		MeMSE	0.51107	0.14642	0.66978
8.	f_8	MMSE	0.57464	0.34316	0.31562
		STD	0.78996	0.14413	0.14387
		MiMSE	0.09185	0.12106	0.1498
		MeMSE	0.35198	0.33578	0.28562
9.	f_9	MMSE	4.4495	4.99545	2.45815
		STD	6.86942	3.3888	2.04287
		MiMSE	0.80194	2.18627	0.46707
		MeMSE	2.59198	4.16064	1.76815
10.	f_{10}	MMSE	1.40039	0.14203	0.13102
		STD	5.91966	0.17185	0.15809
		MiMSE	0.06859	0.0245	0.02607
		MeMSE	0.44645	0.06161	0.07431
11.	f_{11}	MMSE	13.3871	10.76487	11.79746
		STD	2.85788	2.85001	2.24071
		MiMSE	7.018	5.7315	7.29438
		MeMSE	13.02989	10.30997	12.25259
12.	f_{12}	MMSE	0.19647	0.17605	0.1977
		STD	0.06978	0.06306	0.06233
		MiMSE	0.04786	0.06866	0.09271
		MeMSE	0.18422	0.17169	0.19447

The networks are trained using the improved variant of the resilient backpropagation algorithm [16,17] as reported in [18]. This algorithm (iRProp$^+$) is a first order optimization algorithm that has been shown to have a very fast convergence property and scales linearly with the size of the problem [18]. The networks are trained for 2000 epochs.

IV. RESULTS

The summary data for the training of FFANNs is shown in Table II. From the table, on comparison we see that the activation function σ_1 is better than the activation function σ_2 in 4 tasks, that is, in 8 tasks the activation σ_2 is achieving lower average error as compared to σ_1. Thus, on the training data set, across learning tasks, on the basis of the ensemble mean of errors we can rate σ_2 as a better activation function as compared to σ_1. Similarly, if we compare the MMSE values across tasks, we find that h is achieving lower MMSE

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

Sr.No.	Task	Statistics	Activation Functions		
			σ_1	σ_2	$h(x)$
1.	f_1	MMSE	0.31931	0.43612	0.27455
		STD	0.54188	0.53892	0.76904
		MiMSE	0.00314	0.02493	0.00087
		MeMSE	0.02765	0.20774	0.01004
2.	f_2	MMSE	1.13487	0.64459	0.26991
		STD	0.87686	0.51257	0.15098
		MiMSE	0.14848	0.21983	0.10983
		MeMSE	0.78695	0.48589	0.23028
3.	f_3	MMSE	5.47146	3.07216	1.49978
		STD	4.29032	2.46781	0.74386
		MiMSE	0.46573	0.80768	0.65658
		MeMSE	3.70816	2.39848	1.25631
4.	f_4	MMSE	1.70336	0.27949	0.29111
		STD	6.64881	0.13954	0.10742
		MiMSE	0.12975	0.07002	0.15179
		MeMSE	0.356	0.23807	0.27498
5.	f_5	MMSE	3.4262	4.59534	1.58763
		STD	4.64636	5.52123	1.43629
		MiMSE	0.38039	0.86009	0.1572
		MeMSE	2.03865	2.77631	1.24248
6.	f_6	MMSE	1.00216	0.59465	0.4679
		STD	0.92893	0.3141	0.35202
		MiMSE	0.15541	0.12215	0.15849
		MeMSE	0.7973	0.54481	0.31347
7.	f_7	MMSE	0.76615	0.26188	0.89981
		STD	0.46551	0.20954	0.56597
		MiMSE	0.10636	0.02916	0.08812
		MeMSE	0.64584	0.19204	0.79191
8.	f_8	MMSE	2.57514	0.49824	0.43345
		STD	8.17204	0.17539	0.19285
		MiMSE	0.15827	0.17526	0.20876
		MeMSE	0.71107	0.51093	0.39262
9.	f_9	MMSE	7.72942	7.65902	3.78958
		STD	8.79804	4.54249	2.69198
		MiMSE	2.06321	3.56519	1.00383
		MeMSE	5.16097	6.30379	2.87051
10.	f_{10}	MMSE	2.11716	0.21227	0.17479
		STD	7.07927	0.2432	0.17688
		MiMSE	0.12161	0.03499	0.05281
		MeMSE	0.62766	0.09401	0.1131
11.	f_{11}	MMSE	33.59421	22.23543	25.99474
		STD	48.08464	5.27652	4.81935
		MiMSE	15.48349	12.48596	15.701
		MeMSE	26.76708	22.33104	25.70219
12.	f_{12}	MMSE	0.51072	0.40765	0.41087
		STD	0.18488	0.14226	0.11232
		MiMSE	0.13105	0.15982	0.20818
		MeMSE	0.4911	0.40679	0.42504

values than σ_1 in 10 cases, while σ_1 is better in only 2 cases. Comparing σ_2 and h on the basis of MMSE, we observe that σ_2 is better in 4 cases but is worse in 8 cases. Thus on the basis of training errors MMSE, h out performs both σ_1 and σ_2. Similar, analysis when done for the ensemble median (MeMSE) values also allows us to assert the same, that is h is better than σ_2 in 7 cases while it is better than σ_1 in 10 cases.

The summary data for the test data sets is shown in Table III. From the MMSE values obtained, we infer that in 10 tasks σ_2 is better than σ_1, in 10 cases while it is worse only in two cases. Comparison of σ_1 with $h(x)$ on the basis of MMSE values across tasks, tells us that in 11 cases σ_1 is worse than h and is better in only one case. While comparison of σ_2 with h (on MMSE values) leads to the

conclusion that σ_2 is better than h in only 4 cases while it is worse than h in 8 cases.

Similarly when the comparison is performed on the basis of the MeMSE values, we find that h is

Table- IV: Each cell in the matrix shows in how many tasks for training data set, the row activation outperforms the column activation, as measured by Student's t-test at 0.05 significance level.

Activations	σ_1	σ_2	h
σ_1	--	1	1
σ_2	4	--	3
h	7	4	--

Table- V: Each cell in the matrix shows in how many tasks for test data set, the row activation outperforms the column activation, as measured by Student's t-test at 0.05 significance level.

Activations	σ_1	σ_2	h
σ_1	--	0	0
σ_2	7	--	2
h	8	6	--

better than σ_2 in 7 cases and is better than σ_1 in 11 cases while σ_2 is better than σ_1 in 9 cases.

Thus, on the basis of the MMSE and MeMSE, from both training errors and the generalization error, we observe that h outperforms both σ_1 and σ_2 while σ_2 is better than σ_1 in majority of the cases. The result that σ_2 is better than σ_1 in majority of the tasks is in consonance with the results reported in literature, where it has been suggested that anti-symmetric (odd) activation functions should be preferred [19].

The comparisons and the analysis above is on the basis of the differences in MMSE and MeMSE across the tasks and the activation functions. This gross difference provides an indication that the activation function proposed, h eq. (8), is better than σ_1 and σ_2.To measure if these differences are statistically significant or not we compare the means and the medians for the tasks and activation function using FFANNs using the one sided Student's t-test and the Wilcoxon rank-sum test respectively.

The summary of the t-test for training data set is shown in Table IV.

From Table IV it is clear that in majority of the tasks, where there is a significant difference, the proposed activation achieves lower average error as compared to both σ_1 and σ_2. While σ_2 is better than σ_1. Similar results for the test data set is obtained on the usage of the t-test and is shown in Table V.

From Table V it is clear that in majority of the tasks, where there is a significant difference, the proposed activation function achieves lower average error as compared to both σ_1 and σ_2. While σ_2 is better than σ_1.

Similar comparison matrix on the basis of Wilcoxon rank-sum test is shown in Table VI and VII for the training data set and test data set, respectively. And, the conclusions that can be drawn are similar, that is, the FFANNs using the activation function h outperforms both σ_1 and σ_2. While σ_2 is better than σ_1. Moreover, as the median is taken as the more robust estimator of the central tendency, we may assert that the proposed activation function h (eq. 8) majorly outperforms the logistic activation function σ_1 (eq. 4) as well as the hyperbolic tangent activation function σ_2 (eq. 6) even though the activation function h does not produce any negative value for any input in contrast to σ_2.

Table- VI: Each cell in the matrix shows in how many tasks for training data set, the row activation outperforms the column activation, as measured by Wilcoxon rank-sum test at 0.05 significance level.

Activations	σ_1	σ_2	h
σ_1	--	3	1
σ_2	5	--	3
h	9	5	--

Table- VII: Each cell in the matrix shows in how many tasks for test data set, the row activation outperforms the column activation, as measured by Wilcoxon rank-sum test at 0.05 significance level.

Activations	σ_1	σ_2	h
σ_1	--	3	0
σ_2	9	--	2
h	10	7	--

V. CONCLUSION

In this paper we have proposed a non-sigmoidal and non-polynomial function to be used as activation function at the nodes of the hidden layer of the feed forward neural networks. The usage of the proposed activation function in FFANNs was compared with the generally used activation functions, namely the logistic activation function (eq. 4) and the hyperbolic tangent function (eq. 8). On a suite of 12 function approximation tasks, the comparison was conducted, and it is observed that the proposed activation function provides lower errors in most of the tasks. Even though, the proposed activation function's range is (0,1), that is, it does not have any negative output value, the proposed activation function outperforms the hyperbolic tangent activation function also. We are working on a modification of the proposed activation that shall have a negative range of values also and shall be reported separately.

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. P. Chandra, U. Ghose, and A. Sood, A non-sigmoidal activation function for feedforward artificial neural networks, in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.
6. U. Ghose, P. Chandra, and A. Sood, On the feasibility of solving regression learning tasks with ffann using non-sigmoidal activation functions, *IEEE International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2015, pp. 495–500, 2015.
7. W. Duch and N. Jankowski, Survey of neural network transfer functions, Neural Computing Surveys, vol. 2, 1999, pp. 163–212.
8. L. Breiman, The PI method for estimating multivariate functions from noisy data, *Technometrics*, vol. 3, no. 2, 1991, pp. 125–160.

9. 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.
10. V. Cherkassky and F. M¨ulier, *Learning from Data – Concepts, Theory and Methods*. New York: John Wiley,1998.
11. 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.
12. 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.
13. J. H. Friedman, Multivariate adaptive regression splines, *Ann. Statist*, vol. 19, 1991, pp. 1–141.
14. T. Hettmansperger and J. McKean, *Robust Nonparametric Statistical Methods*. Kendall's Library of Statistics: An Arnold Publication No. 5, Arnold, 1998.
15. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*, pp. 977–979. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.
16. 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.
17. 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.
18. C. Igel and M. H¨usken, Empirical evaluation of the improved RProp learning algorithms, *Neurocomputing*, vol. 50, 2003, pp. 105 – 123.
19. 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.