A Comparative Approach to Accelerate Backpropagation Neural Network Learning using different Activation Functions

Dr. Hussein Rady
El-Shorouk Academy,
Higher Institute for Computer & Information Technology
Tel: 0106093311
dr_hussein_rady@yahoo.com

Abstract: *Slow convergence and long training times are still the disadvantages often mentioned when neural networks are compared with other competing techniques. One of the reasons of slow convergence in Backpropagation learning is the diminishing value of the derivative of the commonly used activation functions as the nodes approach extreme values, namely, 0 or 1. In this paper, we propose eight activation functions to accelerate learning speed by eliminating the number of iterations and increasing the convergence rate. Mathematical proving of the errors for the output and hidden layers using these activation functions are concluded. Statistical measures are also obtained using these different activation functions. Through the simulated results, these activation functions are analyzed, compared and tested. The analytical approach indicates considerable improvement in training times and convergence performance.*

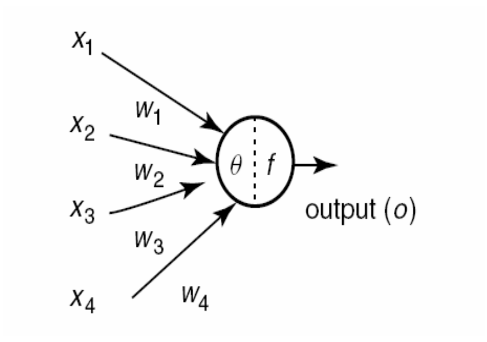
Keywords: Neural Networks, Neural Network Learning, Backpropagation, Activation Functions, Convergence speed.

1. Introduction

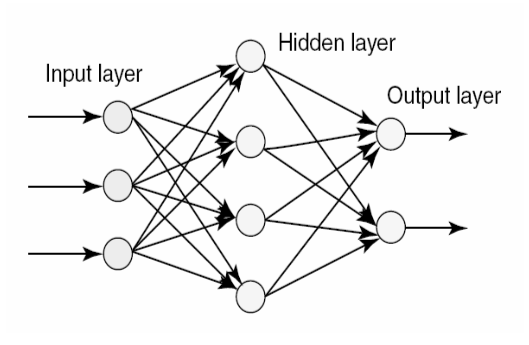
Artificial neural networks (ANN) have been developed as generalization of mathematical models of biological nervous systems [13]. They are known as the "universal approximators" and "computational models" with particular characteristics such as the ability to learn or adapt, to organize or to generalize data [3]. They are widely applied to solving a variety of problems such as information processing, data analysis, system identification, control etc. under structural and parametric uncertainty[2]. Neural networks have been very successful in solving different signal processing and pattern recognition. They have shown great promise in identifying complex nonlinear systems [24].

Artificial Feedforward Neural Networks (FNNs) have been widely used in many application areas in recent years and have shown their strength in solving hard problems in Artificial Intelligence. Although many different models of neural networks have been proposed, multilayered FNNs are the most common. FNNs consist of many interconnected identical simple processing units, called neurons. Each neuron calculated the dot product of the incoming signals with its weight, adds the bias to the resultant, and passes the calculated sum through its activation function [21]. Figure(1) shows the architecture of an artificial neuron and a multilayered neural networks.

Since the Feedforward neural networks are usually too slow for most applications [6], the multilayer Backpropagation (BP) neural networks based on gradient descent are the most common use of neural network models [21].



(a) Constitution of a neuron



(b) b- Multilayered artificial neural network

Figure1: Architecture of an Artificial Neuron and a Multilayer Neural Network.

The multilayer Backpropagation neural networks uses a most famous algorithm known as the Backpropagation algorithm [15]. Training is usually carried out by iterative updating of weights based on minimizing the mean square error. In the output layer, the error signal is the difference between the desired and the output values. Then the error signal is fed back through the steepest descent algorithm to the lower layers to update the weights of the network. The weights of the network are adjusted by the algorithm such that the error is decreased along a descent direction. Traditionally, two parameters, called learning rate and momentum factor, are used for controlling the weight adjustment along the descent direction and for dampening oscillations. However, the convergence rate of the BP algorithm is relatively slow, especially for networks with more than one hidden layer. The reason for this is the saturation behavior of the activation function used for the hidden and output layers. Since the output of a unit exists in the saturation area, the corresponding descent gradient takes a very small value, even if the output error is large, leading to very little progress in the weight adjustment [15].

Our objective is to improve the convergence of the learning process of the Backprobagation algorithm. Basically two different approaches have been developed for improving convergence of the learning process: modification of the activation function and modification of the activation function slope calculation for error propagation [23]. Convergence of the learning process can be improved by changing how the error propagates back through the network. With a standard sigmoid activation function, only a small error propagates back when the neuron is in a maximally wrong state. This is a consequence of using the steepest gradient method for calculating the weight adjustments. For the purpose of error propagation, the slope is calculated from the line connecting the output value with the desired value rather than the derivative of the activation function at the output value [23].

The remainder of this paper is organized as follows: in section 2, general aspects of Backpropagation is discussed. Section 3, shows a description of the Backpropagation neural networks. The role of activation functions is presented in section 4. Deriving the errors using the proposed activation functions is discussed in section 5. The simulation results are shown in section 6. Finally section 7 concluded this paper.

2. General Aspects of Backpropagation

The iterative process of adjustment of weights of a neural network is called training. The aim is to derive a neural network response that approximates the underlying data set with a maximum quality. There exist a variety of different methods to train a neural network. In this study the focus is on the Backpropagation algorithm as one form of supervised error-correction learning. This algorithm requires training data in form of input-output pairs that are obtained by repeatedly applying the computational model for structural analysis. The Backpropagation algorithm may be described with the following three steps, which have to be applied several times in an iteration.

1- Forward computation of input signal of training sample and determination of neural network response
2- Computation of an error between desired response and neural network response.
3- Backward computation of the error and calculation of corrections to synaptic weights and biases

The initial values of the synaptic weights are obtained using random numbers. After the computation of the neural network response to an input signal the response error is determined and used to compute adequate changes of the synaptic weights with the aim of improving the quality of the neural network response in the next step. By applying these corrections to the weights it is attempted to minimize the error surface. Backpropagation is based on a standard gradient method. For the computation of the gradient the activation function needs to be differentiable. The correction of the synaptic weights may be expressed by the following Eq. (1). Within the Backpropagation algorithm two different modes, viz. incremental and batch can be applied [4].

(weight correction) = (learning-rate parameter)*(local gradient)*(input signal) (1)

Incremental Mode The incremental mode, also known as single, on-line or sequential mode, applies a weight correction after each presentation of one sample of training data. The training sample is chosen randomly out of the training data, leading to a stochastic nature of the search for the minimum of the error surface. This maximizes the probability of finding the global minimum. Furthermore, the incremental mode is resistant against redundant training data. As the weight correction is applied after each sample, the exact error of the specific sample is used.

Batch Mode The batch mode applies a weight correction only once after each epoch. During one epoch every sample of the training data is presented to the neural network. The weight correction according to each training sample needs to be stored and the weights are updated after the presentation of the whole training data using all stored weight changes. The training samples are not chosen randomly which assures a convergence to a (local) minimum. Further, redundant training data is disadvantageous, because it yields longer computation time.

3. Description of the Backpropagation neural networks

One of the most commonly used supervised ANN model is backpropagation network that uses backpropagation learning algorithm. Backpropagation algorithm is one of the well-known algorithms in neural networks. The backpropagation neural network is essentially a network of simple processing elements working together to produce a complex output. These elements or nodes are arranged into different layers: input, middle and output. The output from a backpropagation neural network is computed using a procedure known as the forward pass [7]:

- The input layer propagates a particular input vector's components to each node in the middle layer.
- Middle layer nodes compute output values, which become inputs to the nodes of the output layer.
- The output layer nodes compute the network output for the particular input vector.

The forward pass produces an output vector for a given input vector based on the current state of the network weights. Since the network weights are initialized to random values, it is unlikely that reasonable outputs will result before training. The weights are adjusted to reduce the error by propagating the output error backward through the network. This process is where the backpropagation neural network gets its name and is known as the backward pass:

- Compute error values for each node in the output layer. This can be computed because the desired output for each node is known.
- Compute the error for the middle layer nodes. This is done by attributing a portion of the error at each output layer node to the middle layer node, which feed that output node. The amount of error due to each middle layer node depends on the size of the weight assigned to the connection between the two nodes.
- Adjust the weight values to improve network performance using the Delta rule.
- Compute the overall error to test network performance.

The training set is repeatedly presented to the network and the weight values are adjusted until the overall error is below a predetermined tolerance. Since the Delta rule follows the path of greatest decent along the error surface, local minima can impede training. The momentum term compensates for this problem to some degree.

4. The role of Activation Functions

The activation function $z_i = f(x, w_i)$ connects the weights w_i of a neuron i to the input x and determines the activation or the state of the neuron[10]. The behavior of an ANN depends on both the weights and the activation function that is specified for the units. Because the standard BP algorithm descends along the gradient of the error surface, the use of any activation function that has a larger gradient than that of the sum of the squared error at higher energy values would make for faster training. Modification of the Backpropagation algorithm using different methods of determining the slope of the activation function enhances the convergence of the learning procedure.

The great benefit of using neural networks (NNs) is their nonlinear behavior, which allows them to approximate nearly every type of function. Nonlinearities are introduced in NNs by means of the activation function. Ideally, any differentiable function can be used as an activation function [19].

Activation functions for the hidden units are needed to introduce nonlinearity into the network. Without nonlinearity, hidden units would not make nets more powerful than just plaint perceptions (which do not have any hidden units, just input and output units). The reason is that a linear function of linear functions is again a linear function. However, it is the nonlinearity (i.e., the capability to represent nonlinear functions) that makes multilayer networks so powerful. Almost any nonlinear function does the job, except for polynomials. For BP learning, the activation function is differentiable, and it helps if the function is bounded; the sigmoidal functions such as logistic and tanh and the Gaussian function are the most common choices. Functions such as tanh or arctan that produce both positive and negative values tend to yield faster training that functions that produce only positive values such as logistic [18], because of better numerical conditioning[14, 23].

For hidden units, sigmoid activation functions are usually preferable to threshold activation functions. Networks with threshold units are difficult to train because the error function is stepwise constant, hence the gradient either does not exist or is zero, making it impossible to use BP or more efficient gradient-based training methods. Even for training methods that do not use gradients – such as simulated annealing and genetic algorithms – sigmoid units are easier to train than threshold units. With sigmoid units, a small change in the weights will usually produce a change in the outputs, which makes it possible to tell whether that change in the weights is good to bad. With threshold units, a small change in the weights will often produce no change in the outputs [23].

In this paper, the following nonlinear activation functions are investigated that is:
the sigmoid function, the hyperbolic tangent function, the trigonometric sine function, the arctangent function, the Cauchy distribution function, the logistic sigmoid function, the trigonometric cosine function and the Exponential function.

5. Deriving the errors using the proposed activation functions

5.1. The sigmoid function

$$F(v) = \frac{1}{1-e^{-v}} \qquad -\infty < v < \infty \qquad \rightarrow \quad (2)$$

$$F'(v) = \frac{e^{-v}}{[1+e^{-v}]^2}$$

$$= F(v)\big(1-F(v)\big) \qquad \rightarrow \quad (3)$$

Using this energy function, the activation level is between 0 and 1.

For a neuron j located in the output layer

$$\delta_j = O_j(1 - O_j) \quad \rightarrow \quad (4)$$

For a neuron j located in the hidden layer

$$\delta_j = O_j(1 - O_j)\sum_k \delta_k W_{jk} \quad \rightarrow \quad (5)$$

where neuron j is hidden

5.2. The hyperbolic tangent function

$$F(v) = \tanh(v) = \frac{exp(v) - exp(-v)}{exp(v) + exp(-v)} \quad \rightarrow \quad (6)$$

The limiting values of this function are -1 and +1.

The derivative of $F(v)$ w.r.t. to v is

$$F'^{(v)} = \mathrm{sech}^2(v) \qquad \tanh(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}}$$

$$= [1 - \tanh^2(v)]$$
$$= [1 - F^2(v)]$$
$$= [1 - F(v)][1 + F(v)] \quad \rightarrow \quad (7)$$

For a neuron j located in the output layer

$$\delta_j = (1 - O_j)(1 + O_j)(T_j - O_j) \quad \rightarrow \quad (8)$$

For a neuron j located in the hidden layer

$$\delta_j = (1 - O_j)(1 + O_j)\textstyle\sum_k \delta_k w_{jk} \quad \rightarrow \quad (9)$$

5.3. The trigonometric sine function

$$F(x) = \sin x \quad \rightarrow \quad (10)$$

$$\begin{aligned} F'(x) &= \cos x \\ &= \sqrt{1-\sin^2 x} \\ &= \sqrt{1-F^2(x)} \\ &= \sqrt{[1-F(x)][1+F(x)]} \end{aligned} \quad \rightarrow \quad (11)$$

For a neuron j located in the output layer

$$\delta_j = \sqrt{(1-O_j)(1+O_j)(T_j-O_j)} \quad \rightarrow \quad (12)$$

For a neuron l located in the hidden layer

$$\delta_j = \sqrt{(1-O_j)(1+O_j)\Sigma_k \delta_k w_{jk}} \quad \rightarrow \quad (13)$$

5.4. The arctangent function

$$F(v) = y = \tan^{-1} v \quad \rightarrow \quad (14)$$

$$\therefore F'(v) = \frac{1}{1+\tan^2 F(v)} \quad \rightarrow \quad (15)$$

For a neuron j located in the output layer

$$\delta_j = \frac{1}{1+\tan^2 O_j}(T_j - O_j) \quad \rightarrow \quad (16)$$

For a neuron j located in the hidden layer

$$\delta_j = \frac{1}{1+\tan^2 O_j}\Sigma_k \delta_k w_{jk} \quad \rightarrow \quad (17)$$

where neuron j is hidden.

5.5. The Cauchy distribution function

$$F(v) = 0.5 + \frac{1}{\pi}\tan^{-1} v \quad \rightarrow \quad (18)$$

The formula for the cumulative distribution function for the Cauchy distribution is:

$$\therefore F(v)' = \frac{1}{\pi}\left[\frac{1}{1+\tan^2 F(v)}\right] \quad \rightarrow \quad (19)$$

For a neuron j located in the output layer

$$\delta_j = \frac{1}{\pi(1+\tan^2 O_j)}(T_j - O_j) \quad \rightarrow \quad (20)$$

For a neuron j located in the hidden layer

$$\delta_j = \frac{1}{\pi(1+\tan^2 O_j)} \sum_k \delta_k w_{jk} \rightarrow (21)$$

5.6. Logistic sigmoid $(x) = \frac{2}{1+e^{-x}} - 1$ **function F**

$$\therefore F(v) = \frac{2}{1+e^{-v}} - 1 \qquad -\infty < v < \infty \rightarrow (22)$$

$$\therefore F'(v) = \frac{2e^{-v}}{[1+e^{-v}]^2}$$

$$= 2F(v)(1-F(v)) \rightarrow (23)$$

using this energy function, the activation level is between 0 and 1.

For a neuron j located in the output layer

$$\delta_j = 2\,O_j(1-O_j)(T_j - O_j) \rightarrow (24)$$

For a neuron j located in the hidden layer

$$\delta_j = 2\,O_j(1-O_j)\sum_k \delta_k w_{jk} \rightarrow (25)$$

where neuron j is hidden

5.7. Trignometric cosine function

$$F(v) = \cos v \rightarrow (26)$$

$$F'(v) = -\sin v$$

$$= -\sqrt{1-\cos^2 v}$$

$$= -\sqrt{1-F^2(v)}$$

$$= -\sqrt{[1-F(v)][1+F(v)]}$$

For a neuron j located in the output layer

$$\delta_j = -\sqrt{(1-O_j)(1+O_j)(T_j - O_j)} \rightarrow (27)$$

For a neuron j located in the hidden layer

$$\delta_j = -\sqrt{(1-O_j)(1+O_j)\sum_k \delta_k w_{jk}} \rightarrow (28)$$

5.8. Exponential function

$$F(v) = e^{-v} \quad \rightarrow \quad (29)$$

$$\therefore F'(v) = -e^{-v}$$

$$\therefore F'(v) = -F(v) \quad \rightarrow \quad (30)$$

For a neuron j located in the output layer

$$\delta_j = -O_j(T_j - O_j) \quad \rightarrow \quad (31)$$

For a neuron j located in the hidden layer

$$\delta_j = -O_j \sum_k \delta_k W_{jk} \quad \rightarrow \quad (32)$$

where neuron j is hidden

6. Simulation Results

The activation function of neurons allows non-linearity to be introduced into neural network training and determines the elasticity of weight changes. It therefore can improve convergence in training. The sigmoid function is, anyway, the most widely used activation function. It is a smooth function, which returns nearly proportional outputs for intermediate values, while smoothing out values at the extremes of the spectrum. The hyperbolic function is mostly similar to the sigmoid function. The hyperbolic function is negatively oriented, tending to force extreme values of the distribution to either 1 or – 1.

While any of the described functions can be implemented in NNs, there is no clear rule on how to select the most appropriate activation function. The use of sigmoid functions for classification problems, and of hyperbolic functions for forecasting problems, that is, when learning about deviations from the average is involved. A different function can ideally be used for each computational unit in the NN. While the usual NN models found in the literature employ the same activation function for all units, examples can also be found of NNs in which a different function is selected for the output units. Sigmoid functions are mostly used in the input and hidden layers, while there is no agreement on what activation function should be employed for the output units [19].

In this section, and on simulated data sets, we discuss some of the figures and tables showing the behavior resulting from the practical implementation as a comparison between the different eight activation functions.

Figure (2) shows the number of iterations for each activation function. In this figure, we find that using the trigonometric sine activation function is the best one, and the multilayer perception trained with the BP algorithm learn faster in terms of the number of iterations.

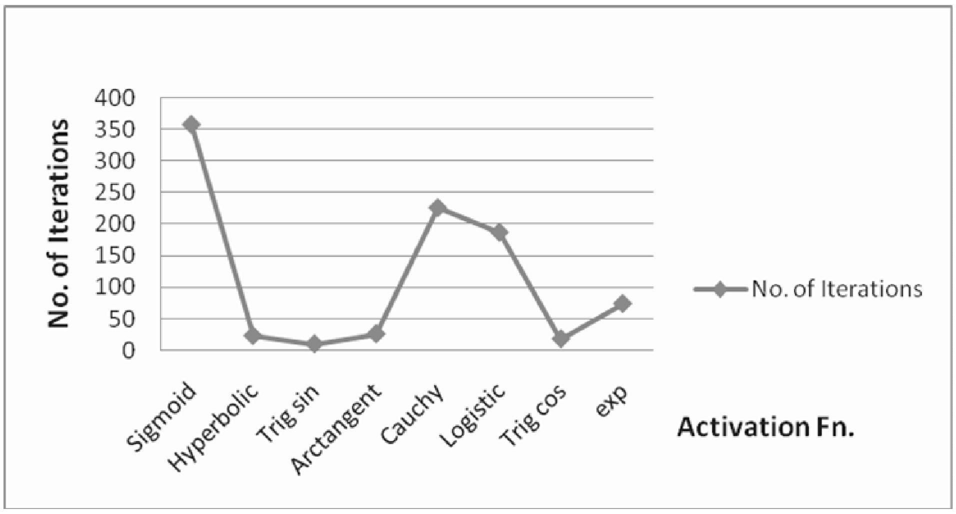


Figure 2 : The Number of Iterations For Each Activation Function.

The second function in terms of the number of iterations is the trigonometric cosine function. The third function is the hyperbolic tangent function, the fourth function is the arctangent activation function, the fifth function is the exponential activation function, the sixth function is the logistic activation function, the seventh function is the Cauchy distribution activation function, and finally the last one in terms of the number of iterations is the sigmoid activation function.

Table (1) shows the number of iteration, actual output, the sum of square error (SSE) and the mean square error (MSE) for each activation function given that the desired output = 0 and the learning rate = 0.5.

Table1: Shows the Number of Iteration, Actual Outputs, SSE, and MSE.

Activation Function	Number of Iterations	Actual Output	Sum of square error(SSE)	Mean Square error(MSE)
Sigmoid	358	0.1996718	0.01996718	0.00996721
Hyperbolic	23	0.184995643	0.01711169	0.00855584
Trig sin	10	0.1824833	0.0166500	0.0083250
Arctangent	26	0.1928384	0.0185933	0.0092966
Cauchy	226	0.1999554	0.01999108	0.0099955
Logistic	187	0.1997979	0.0199596	0.0099798
Trig cos	18	0.193509	0.018722	0.009361
Exp	74	0.158863	0.012618	0.006309

Figure (3) . Shows the Behavior of the MSE For All the activation Functions.

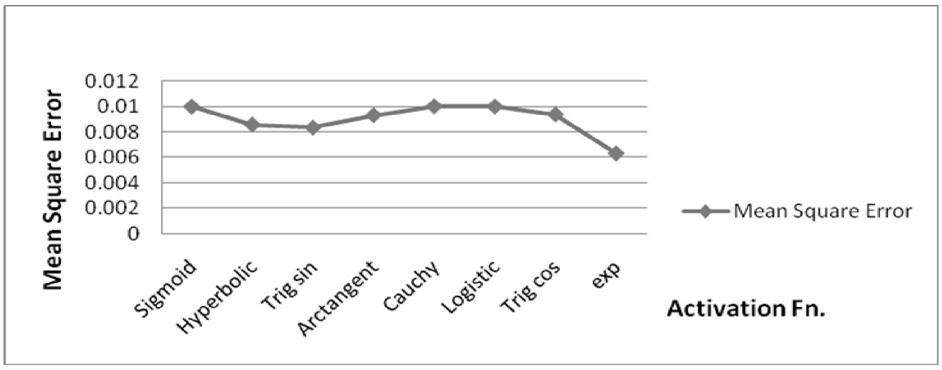


Figure 3 : Shows the MSE Using the Eight Activation Functions.

It shows that the graduation of the MSE from lowest value to the largest value is as follows: the exponential activation function, the trigonometric sine, the hyperbolic tangent, the arctangent, the trigonometric cosine function, the sigmoid function, the logistic function, and the Cauchy distribution activation function.

Table (2) shows the degree of convergence using each activation function. The table shows that the exponential activation function = 84.114 % is higher than the trigonometric sine activation function = 81.752 % which is higher than the hyperbolic tangent function = 81.5 % which is higher than the other functions which falls nearly on the same level.

Table (2) shows the degree of convergence using each activation function.

Activation Function	Number of Iterations	Degree of convergence
Sigmoid	358	80.033 %
Hyperbolic	23	81.500 %
Trig sin	10	81.752 %
Arctangent	26	80.716 %
Cauchy	226	80.004 %
Logistic	187	80.020 %
Trig cos	18	80.649 %
exp	74	84.114 %

Table (3) Shows Some Statistical Calculations Illustrating the Behavior of the actual outputs for the last nine iterations. The minimum statistic shows the last actual output for the stopping criteria = 0.01 and the desired output = 0. The table shows that the exponential activation function is the nearest one to the desired output. Following to this function is the trigonometric sine function, etc., but comparing the maximum and minimum for each activation function, we find that the trigonometric sine function is more convergent than any other function. This is shown also from the mean value. The standard deviation STD shows the deviation from the mean.

Table (3) shows the Min, Max, Mean, and STD for the actual output using the last nine iterations.

Statistics	Sigmoid	Hyper-bolic	Trig sin	Arctan	Cauchy	Logistic	Trig cos	exp
Min	0.19967	0.18499	0.18248	0.19284	0.19996	0.19979	0.19351	0.15886
Max	0.20641	0.73569	0.83682	0.48608	0.20584	0.20509	0.34337	0.30024
Mean	0.20304	0.26648	0.49036	0.43522	0.26792	0.16312	0.18924	0.22029
STD	0.00477	0.26648	0.28076	0.25931	0.14927	0.09009	0.12178	0.08249

Figure (4) illustrates the phenomena shown in the table. It shows that the trignometric sine function has a significant difference between the maximum value and the minimum value. Following this function is the hyperbolic tangent function. The sigmoid function which has a low convergence reaches to the stability in terms of we find that the Min value, the Max value, and the Mean value are approximately equal.

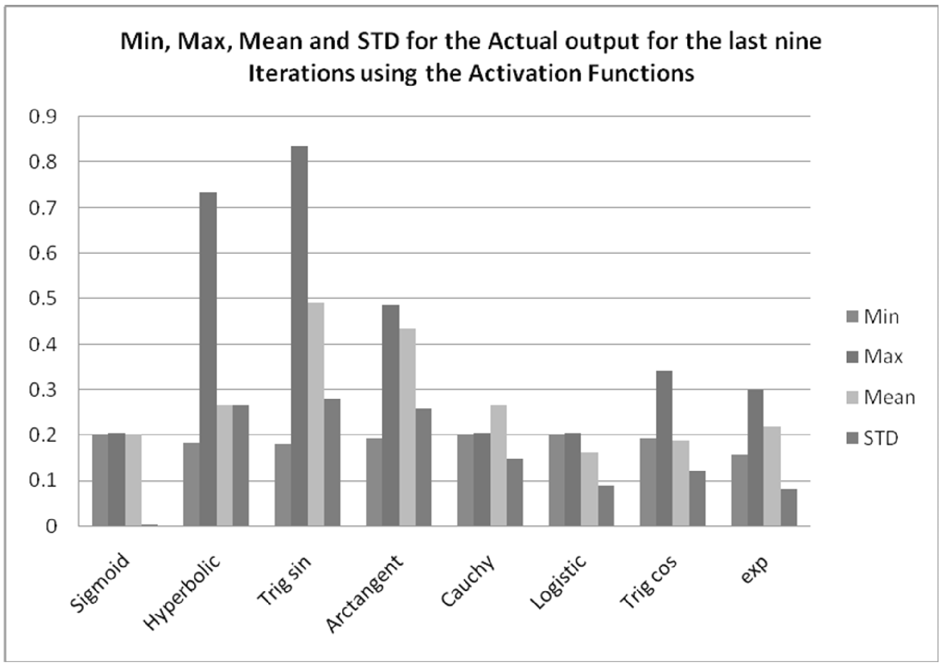
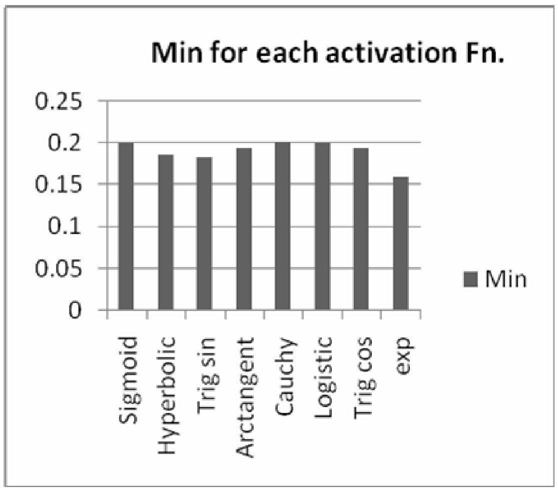
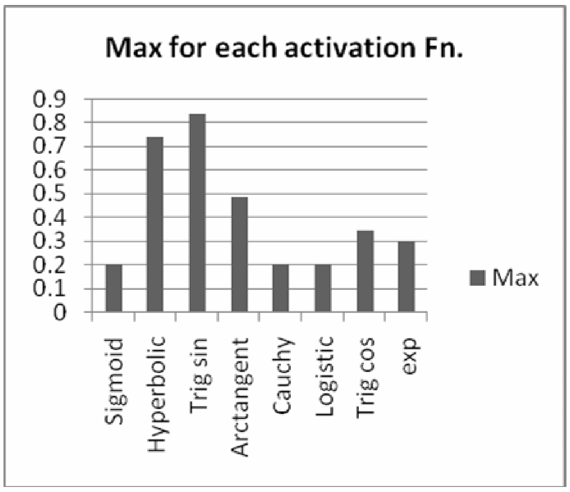


Figure 4 : Shows a Comparison Among the Eight Activation Functions (Min, Max, Mean, and STD)

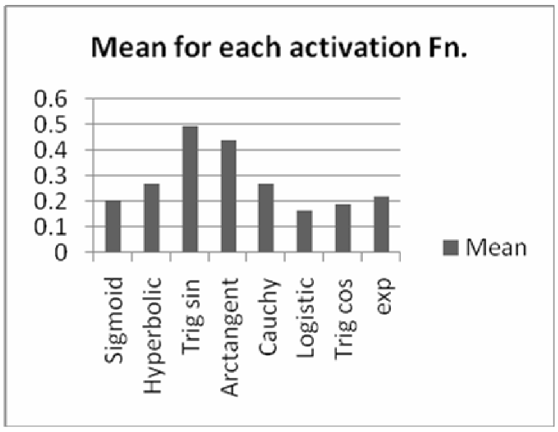
Figure (5) shows statistical results which indicat the behavior of the neural network learning for the last nine iterations using each activation function. In Figure (5-a) the actual output using the exponential function is the lowest i.e., it is the nearest to the desired output = 0. Following to it is the trignometric sine function, and so on as illustrated in the figure. The Max as shown in Figure (5 – b) means the value of the actual output for the ninth iteration from the last. The behavior is shown in the Figure. Figure (5 – c) the mean of the last nine iterations for the actual output coressponding to the desired output = 0. The Figure shows the stability of the logistic function over all the other functions. Figure (5 – d) shows standard deviation using each activation function. The Figure shows that the sigmoid function is the lowest in the convergence and the deviation from the mean is a smallest value compared to all the other functions.


Min for each activation Fn.
0.25
0.2
0.15
0.1
0.05
0
■ Min
Sigmoid
Hyperbolic
Trig sin
Arctangent
Cauchy
Logistic
Trig cos
exp

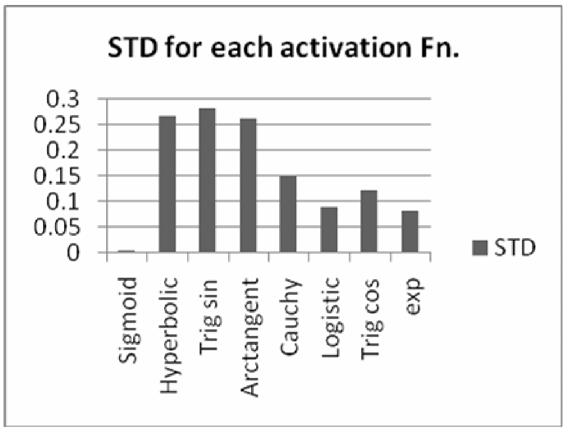
(a)


Max for each activation Fn.
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
■ Max
Sigmoid
Hyperbolic
Trig sin
Arctangent
Cauchy
Logistic
Trig cos
exp

(b)



(c)



(d)

Figure 5 : Shows The Statistics (Min, Max, Mean, and STD) For Tha Actual Output in the Last Nine Iterations.

7. **Conclusions**

The multilayer Backpropagation neural networks needs a lot of research to overcome the slow convergence and the long training times. To alleviate the need for eliminating the number of iterations, speeding the convergence rate and to increase the efficiency of BP-Learning, we developed an efficient set of activation functions which improves the convergence accuracy, speed and reduces the number of iterations using each activation function. Simulation results show that the proposed comparative approach achieves a good performance and considerable enhancements for the BP-learning. We conclude the following issues:

- Mathematical proving for the errors in the output and hidden layers are concluded.
- The number of iterations using the trigonometric sine activation function is the lowest, so this function is the best. The one that follows is the trigonometric cosine function. The hyperbolic tangent function is the third. The fourth function is the arctangent function. The fifth function is the exponential function in terms of the number of iterations. The sixth function is the logistic, the Cauchy function is the seventh and the last activation function is the sigmoid activation function.
- The convergence rate of the BP-learning increases using the exponential function. Following to this is the trigonometric sine function, and the hyperbolic tangent function lies in the third category. The arctangent function is the fourth one. The fifth function is the trigonometric cosine function. There is a little difference among the three last equations namely, sigmoid, logistic and Cauchy.
- Statistical results (Min, Max, Mean, and Standard deviation) show the behavior of the actual output for the last iterations of the neural network are conducted as shown in the simulation results.

The analysis illustrated in the present paper can be expanded by carrying out further research in several directions.

References

[1] Karma R., Bressler S., Perlovsky L., And Kamar G. “Advances In Neural Networks Research: An Introduction”. Advances in Neural Networks Research: International Joint Conference on Neural Networks. Volume 22, Issues 5-6, 2009.

[2] Bodyanskiy Y., Pliss I. And Slipchenko. “Growing Neural Networks Using Nonconventional Activation Functions”, Information Theories & Application International Journal Vol. 14, 2007.

[3] Kiranyaz S., Ince T., Yildirim A. And Gabbouy M. "Evolutionary Artificial Neural Networks By Multi-Dimentional Practical Swarm Optimization." Neural Networks, Volume 22, Issue 10, pages 1448-1462, 2009.

[4] Beyer W., Liebscher M., Beer M. And Graf W. "Neural Network Based Response Surface Methods – A Comparative Study." In Proceedings of the 5th German LS-DYNA Forum, 2006.

[5] Sun J. "Local Coupled Feedforward Neural Network." Neural Networks, volume 23, Issue 1, pages 108-113, 2009.

[6] Rajapandiar V. And Gunaseeli N. "Modified Standard Backpropagation Algorithm With Optimum Initialization For Feedforward Neural Networks." International Journal Of Imaging Science And Engineering, Vol. 1, No. 3, 2007.

[7] Shihab K. "A Backpropagatiion Neural Network For Computer Network Security." Jouranl Of Computer Science 2(9): 710-715, 2006.

[8] Tezel G. And Ozbay Y. "A New Neural Network With Adaptive Activator Function For Classification Of ECG Arrhythmias." Volume 19, Issue 6, Digital Signal Processing, 2009.

[9] Anilkumar K. And Tanprasert T. "A Subjective Scheduler Based On Backpropagation Scheduling Situation." International Journal Of Intelligent Systems And Technologies, 2008.

[10] Debes K. Koenig A. And Gross H. "Transfer Functions In Artificial Neural Networks, A Simulation-Based Tutorial." Journal of Brains, Minds and Media, Volume1, Number 1, 2005.

[11] Khalil R. "Digital Image Compression Enhancement Using Bipolar Backpropagation Neural Networks." Al-Rafidain Engineering, Volume 15 Number 4, 2007.

[12] Shrestha R., Theobald S. And Nestmann F. "Simulation Of Flood Flow In A River System Using Artificial Neural Networks." Hydrology And Earth System Science, 9(4), 313-321, 2005.

[13] Abraham A. "Artificial Neural Networks Handbook Of Measuring System Design", WILEY / Engineering & Materials Science / Electrical & Electronics Engineering, John Wiley & Sons, Ltd. 129: Artificial Neural Networks, 2005.

[14] Rady H. "Classification Of Multilayer Neural Networks Using Cross Entropy And Mean Square Errors." Journal of ACS, Vol. 2, May 2008.

[15] Rady H. "Different Aspects For Enhancing The Backpropagation Neural Networks." Journal of ACS, Vol. 1, May 2007.

[16] Kham A. Bandopadhyaya T. And Sharma S. "Genetic Algorithms Based Backpropagation Neural Network Performs Better Than Backpropagation Neural Network In Stock Rates Prediction." International Journal Of Computer Science And Network Security, Vol. 8 No. 7, 2008.

[17] Krissilov A., Krissilov D. And Oleshko N. "Application Of The Sufficiency Principle In Acceleration Of Neural Networks Training." International Journal "Information Theories & Applications" Vol. 10, 1997.

[18] Migo L., Aslanyan L., Gastellanos J., Diaj M., And Riajanov V. "Fourier Neural Networks: An Approach With Sinusoidal Activation Functions." Information Theories & Applications International Journal Vol. 11, 2003.

[19] Patuell R., Reggiani A., Nijkamp P. And Schanne N. "Neural Networks For Cross Sectional Employment Forecasts: A Comparison Of Model Specification For Germany.“ Library Networks of Western Switzerland. Scientific Commons, 2008.

[20] Joarder K. "Arctangent Activation Function To Accelerate Backpropagation Learning." IEICE Tranansactions Fundam. Electron Community Computer Science, Vol. E85-A; No. 10 page 2373-2376, 2002.

[21] Plagianakos V. And Vrahatis M. "Training Neural Networks With Threshold Activation Functions And Constrained Integer Weights." IEEE Computer Society, International Joint Conference on Neural Networks, Volume 5, 2000.

[22] Aran O. And Alpaydin E., "An Incremental Neural Network Construction Algorithm For Training Multilayer Perceptions." Institute for Graduate Studies in Science and Engineering, 2002.

[23] Donald T. "Why Use Activation Functions?", comp.ai..neural-nets FAQ, part 2: Learning , http://www.Faqs.Org/Faqs/Ai-Faq/Neural.Nets/. 2009.

[24] Torvik L. And Wilamowski B. "Modification Of The Backpropagation Algorithm For Faster Convergence." An International Conference on Computational Intelligence: Neural Networks, Fuzzy Systems, Evolutionary Programming and Virtual Reality. Proc. SPIE Vol 2204, Fifth Workshop on Neural Networks, 1993.

[25] Nechyba M. And Xu Y. "Neural Network Approach To Control System Identification With Variable Activation Functions." IEEE International Symposium on Intelligent Control, pp 358-363, 1994.