Comparison of Activation Functions in Feedback Neural Networks for Recognition of Hand Written Numbers

Chandan M S, Student, MVJ College of Engineering

email id : mschandan96@gmail.com

***Abstract*—The idea of Neural networks is gaining more and more recognition as we are heading towards the age of Artificial Intelligence. This paper introduces to the concept of handwritten number recognition using the model of neural networks which is trained by several datasets of handwritten numbers. The paper concentrates on the mathematical derivation of error correction and the study of various activation functions along with their efficiency to optimize the network through back propagation of errors and update the weights of the network for the smooth and precise output.**

Index Terms* – *artificial Intelligence, activation functions ,back propagation, neural machine learning, recurrent neural network.

I. Introduction

The concept of Neural Networks is based on the function of neurons in the human brain. These networks try to mimic the computation method of the brain by effectively grouping the set of neurons to achieve the given task. The network is trained and optimized by giving a large amount of datasets which comprises of variations in the problem domain to be solved. Traditional technique involves processing of data sequentially without any ambiguity or fuzziness where as neural networks works quite opposite to that of the above. It processes the data in parallel and fuzziness is the key feature of computation. The major advantage is the computational speed of computers which is a lot more than that of a brain which works in slow rhythms to process the data.

For any neural networks to be truly efficient, it should be fed with large amount of datasets and appropriate correction of weights allocated to the network. This is obtained by assigning the mathematical functions which involves basic differentiation and integration along with matrix calculations.

A. Purpose

The purpose of this paper is to establish the power of neural networks in image processing and recognition of hand written characters and comparative study of activation functions to be used. This is demonstrated by the effective implementation of neural networks in recognizing the hand written numbers up to the accuracy of ninety seven percent.

B. Outline

In section II, the origin, classification of neural networks and the capabilities of the same is described. The section III explains the detailed working of feedback neural networks and involvement of activation functions. Section IV outlines the mechanism of handwritten digit dataset recognition and performance evaluation. Section V gives a comparative study of efficiency in various activation functions.

II. Origin and classification of neural networks

Neural networks are a computational approach used in computer science and other research disciplines, which is based on a large collection of neural units (artificial neurons), loosely mimicking the way a biological brain solves problems with large clusters of biological neurons connected by axons. Each neural unit is connected with many others, and links can be enforcing or inhibitory in their effect on the activation state of connected neural units. Each individual neural unit may have a summation function which combines the values of all its inputs together. There may be a threshold function or limiting function on each connection and on the unit itself, such that the signal must surpass the limit before propagating to other neurons. The concept of artificial neuron was first introduced by Warren McCulloch and Walter Pitts in 1943[1]. The basic structure of an artificial neuron is shown in the Fig. 1.

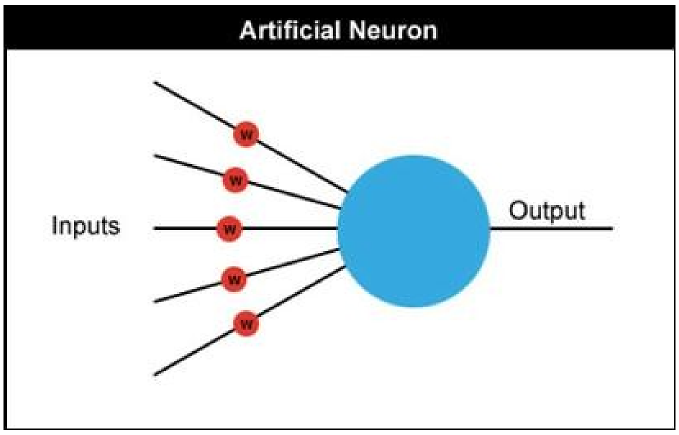

Fig. 1. The basic structure of an artificial neuron along with weights , inputs and outputs.

A network of neurons take several input and gives the outputs which are then evaluated accordingly according to the activation function.

Activation function is a function that helps to refine the outputs using linear regression of inputs and customize the weights of the nodes. The network of neurons are essentially made of numerous neurons and respective activation functions to compute the data in parallel.

There are different types of Artificial Neural Networks depending on the task to be achieved. Main types of Artificial Neural Networks are :

a) Feedback Neural Networks - In these type of ANN, the output goes back into the network to achieve the best-evolved results internally. The networks are used by the internal system error corrections. Fig describes the feedback neural network of three layers.

b) Feed forward Neural Networks – In this type, the information flow is unidirectional. A unit sends information to other unit from which it does not receive any information. They are used in pattern generation/recognition/classification. They have fixed inputs and outputs.

III. Working of Feedback neural networks and activation functions

A feedback neural network is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feed forward neural networks, they can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition or speech recognition. The first development of backpropogation goes back to the work of goes back to the work of Paul Werbos.[2][3]

Fig. 2 shows the basic structure of feedback neural network with three layers.

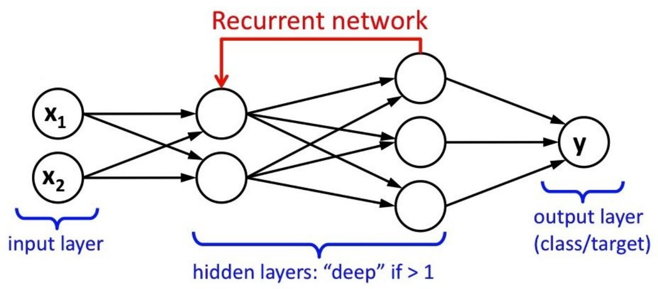

Fig. 2. The feedback or recurrent neural network with three layers.

A network comprises of three layers:

a) Input layer – The input layer is the layer where various inputs in the form of datasets are given.

b) Hidden layer – The hidden layer is the layer where the major computation takes place. There can be more than one hidden layer.

c) Output layer – The output layer is the layer where the output is shown.

Activation function - A function used to transform the activation level of a neuron into an output signal. Traditionally, there is a certain threshold for the function where only when this threshold is breached, the output is computed within the layer.

Weights – Each connection between nodes is given a certain weight. These weights are responsible for the variation in values of activation function. Hence the threshold can be manipulated by adjusting the weights of the connection through which we can also adjust the output. This is where the error correction via feedback comes into picture. The Fig. 3 demonstrates the a three layered feedback network along with weights of each connection.

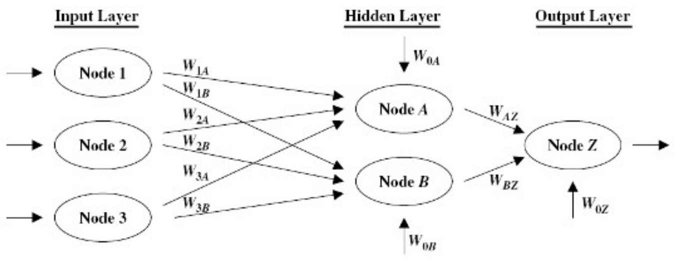

Fig. 3. The three layered network along with the representation of weights

A. Calculation of outputs of nodes

From Fig. 3, let's consider the input in the form of matrix I.

$$I = \begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} \quad (1)$$

The weights between the input and hidden layers are given in the form of matrix W_{inhi}.

$$W_{inhi} = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{pmatrix} \qquad (2)$$

The combined moderated input into the hidden layer is given by the matrix multiplication of inputs and weights between the input and hidden node.

$$X_{hi} = W_{inhi} \cdot I \qquad (3)$$

The X_{hi} can also be represented in the form of matrix as given below

$$X_{hi} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \qquad (4)$$

The nodes apply the activation function for every value in X_{hi} to compute the output from hidden nodes which is denoted by O_{hi}. The O_{hi} is computed as follows where A_{funct} is the activation function.

$$O_{hi} = A_{funct}(X_{hi}) \qquad (5)$$

The output from the hidden nodes after the normalization through activation function can be represented in the form of matrix as given below

$$O_{hi} = \begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} \qquad (6)$$

Similarly, the input for the output layer is given by O_{hi} and is multiplied with the weights between hidden and output nodes before passing it to activation function. The X_{out} is given as follows.

$$X_{out} = W_{hiout} \cdot O_{hi} \qquad (7)$$

Thus, the final output is represented by O_{out} and can be obtained by passing the X_{out} through the activation function.

$$O_{out} = A_{funct}(X_{out}) \qquad (8)$$

B. Backpropogation of errors

The fundamental characteristic of a feedback network is the backpropogation of errors. The goal is to update each of weights so that they cause actual the actual output to be nearer to the target output, hence minimizing the error for each output and the network as a whole.

Let's consider the three layer network with each layer having two nodes as shown in the Fig. 4,

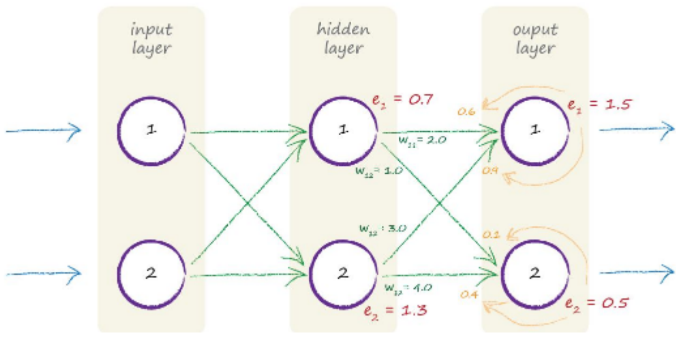

Fig. 4. The three layered network with each layer containing two nodes and errors denoted in each layer.

It is seen that the error from the output of hidden node is split into the weights between the hidden and output layers. This error is the sum of split errors multiplied with weights which can be put into mathematical form as follows

$e_{hidden,\,1}$ = sum of split errors on weights $w_{1,1}$ and $w_{1,2}$

where $e_{hidden,\,1}$ is the error from the node one of hidden layer and $w_{1,1}$ and $w_{1,2}$ are the weights between the hidden and output layer. The individual error constitutes as given below,

$$e_{h,1} = (e_{o,1} * w_{1,1}) + (e_{o,2} * w_{1,2}) \qquad (9)$$

$$e_{h,2} = (e_{o,1} * w_{2,1}) + (e_{o,2} * w_{2,2}) \qquad (10)$$

The (9) and (10) are represented in the form of matrix multiplication as follows

$$e_h = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix} \cdot \begin{pmatrix} e_{o1} \\ e_{o2} \end{pmatrix} \tag{11}$$

The matrix approach to backpropogate the error to hidden nodes can be shortly written as,

$$E_{hidden} = W_{hiout} \cdot E_{out} \tag{12}$$

C. Gradient descent and update of weights

The error propagated back is used as a guide to adjust the weights of the links to substantially increase the overall output of the network. In order to adjust the weights appropriately and in a trivial way, the mathematical approach used is called Gradient descent. Gradient descent is an optimization algorithm used to find the values of parameters of a function that minimizes the cost. It provides the power to calculate the minimum value without actually understanding the complex functions. This helps in smoothing out the error to absolute minimum by running enough iterations and determine the direction of error. The error at the output is given by

$$Error = Target - Actual \tag{13}$$

But taking the error in (13) will make the positive and negative errors cancel each other out. In order to correct this, the square of the difference is taken as the error. This serves the purpose of making the error function smooth and continuous and also minimizes the risk of overshooting the objective. Hence the error function (13) can be modified as

$$Error(E) = (Target - Actual)^2 \tag{14}$$

In order to find the minimum value of the error function of (14), the slope of the function to descent to the minimum value. This can be represented mathematically with respect to the weights as,

$$\frac{\partial E}{\partial w_{j,k}} = \frac{\partial (t_n - O_n)^2}{\partial w_{j,k}} \tag{15}$$

where t_n and O_n denotes target and output of a nth node respectively, $w_{j,k}$ denotes the weights between the j and k layers. The right hand side of (15) is the expansion of the error function as given in (14). After the application of chain rule of differentiation ,

$$\frac{\partial E}{\partial w_{j,k}} = \frac{\partial E}{\partial O_k} \cdot \frac{\partial O_k}{\partial w_{j,k}} \tag{16}$$

The error function can be tackled by the application of simple derivative of the square function.

$$\frac{\partial E}{\partial w_{j,k}} = -2(t_k - O_k) \cdot \frac{\partial O_k}{\partial w_{j,k}} \tag{17}$$

The output O_k is the result of outcomes of the activation function hence the variable can be replaced with the activation function which is does the summation of the weights and the nodes in between the j and k layers of the network itself. The (17) can be given as,

$$\frac{\partial E}{\partial w_{j,k}} = -2(t_k - O_k) \cdot \frac{\partial A_{funct}\left(\sum_j w_{j,k} \cdot O_j\right)}{\partial w_{j,k}} \tag{18}$$

From the (18), it is noted that the negative sign represents the direction of the slope which is helpful in the correction of error. Also the constants can be neglected since it doesn’t make any difference to the outcome. Hence the final slope of the error is,

$$\frac{\partial E}{\partial w_{j,k}} = -(t_k - O_k) \cdot \frac{\partial A_{funct}\left(\sum_j w_{j,k} \cdot O_j\right)}{\partial w_{j,k}} \tag{20}$$

Now the weights can be updated since the slope of the error is known. A constant called learning factor is also introduced which can be used to tune the rate of learning of neural networks. The mathematical expression for the update of weights is given by,

$$new\, w_{j,k} = old\, w_{j,k} - \alpha \frac{\partial E}{\partial w_{j,k}} \quad (21)$$

where α is the learning factor. Note that since the slope is negative in (20), the subtraction of the slope is done from the old weight to obtain the new weight.

D. Different activation functions

The activation function translates the input signals to output signals. The purpose of these functions is to introduce non linearity into the network. They promote the linear regression of the unbounded data points given. The most commonly used activation functions are

a) Sigmoid function - A sigmoid function is a mathematical function having an "S" shaped curve (sigmoid curve). Often, sigmoid function refers to the special case of the logistic function shown in the Fig. 5 and defined by the formula

$$S(t) = \frac{1}{1+e^{-t}} \quad (22)$$

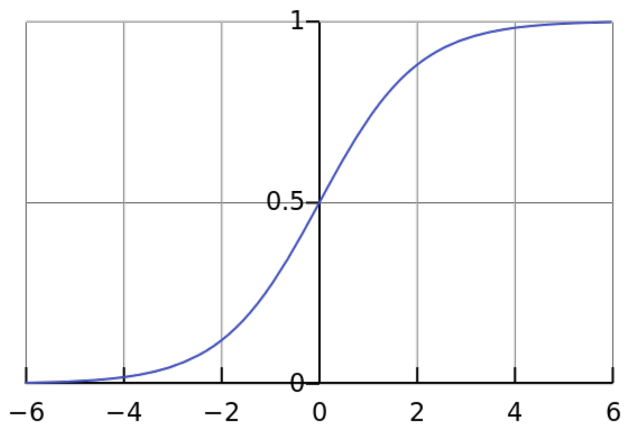

Fig. 5. The plot of sigmoid function

b) Hyperbolic tangent function - The hyperbolic tangent function is defined as the ratio between the hyperbolic sine and the cosine function as shown in the Fig. 6 and is defined by the formula

$$\tanh(t) = \frac{e^{t} - e^{-t}}{e^{t} + e^{-t}} \quad (23)$$

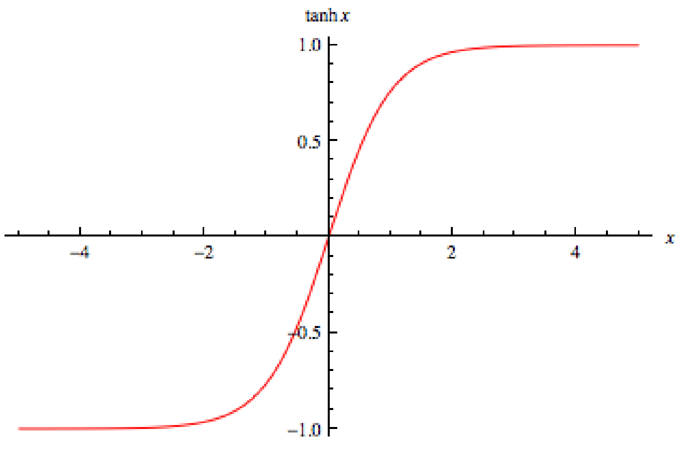

Fig. 6. The plot of hyperbolic tangent function

c) Arctangent function – The arctangent function also known as inverse tangent function is the inverse of trigonometric tangent function. The function is shown in the Fig. 7 and the formula is defined as

$$\arctan(t) = \tan^{-1}(t) \quad (24)$$

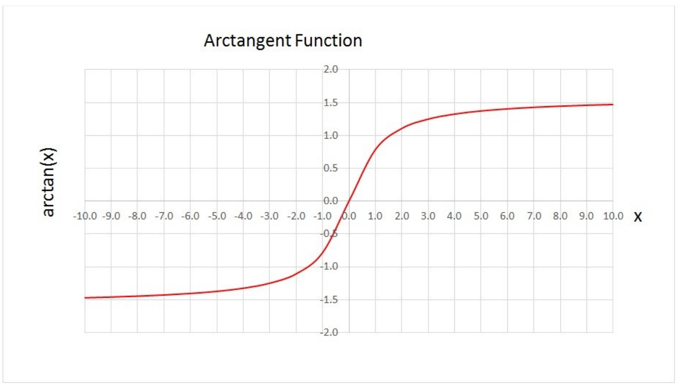

Fig. 6. The plot of arctangent or inverse tangent function

IV. MNIST Datasets and Recognition of Digits

Every neural network needs to be trained on data sets to get the better performance and achieve accuracy in the task given. To train a network to recognize the handwritten digits, the MNIST dataset. The MNIST is the database comprising of handwritten digits and is provided by the respected neural network researcher Yann LeCun’s website[4]. There are two sets of data namely training set and test set. The training set provided to the network is set of labeled examples that is the income come with the desired output. The test set is used to analyze the development of algorithm and the working of the network idea. The first value of these data set is called label, that is, the actual digit the handwriting is supposed to represent and the subsequent values are the pixel values of handwritten digits which marks the pixel coordinates in plotted graph. These subsequent values form a pixel array of 28 by 28 makes into seven hundred and eighty four values after the label.

The training of the network to recognize the hand written digits is done by feeding it the MNIST database. The neural network for the task consists of 784 input nodes which serves as the pixel array of the dataset. There are ten output nodes in which each represent the digits from zero to nine. The number of nodes in the hidden layer varies from each network and activation function. The best method on deciding the number of hidden nodes is to try variety of numbers then doing a comparative study to choose the optimum value. When the MNIST dataset is provided to input nodes, each node transmits a value of the pixel array to the hidden nodes. After the computation and processing of the values through activation functions and weights, the output is emitted as a result. The node which has the maximum value is considered to be the resultant digit. The considered digit is then compared with the label of the dataset provided to evaluate the answer. If there is an error, then the back propagation of error is done to rectify and improve the weights for better accuracy. The enhancement in accuracy and overall performance can be done by experimenting with various learning rates as given in (21).
The graph in Fig. 7 shows the performance change with respect to different learning rates. It can be seen that the performance substantially decreases as the rate of learning approaches one.

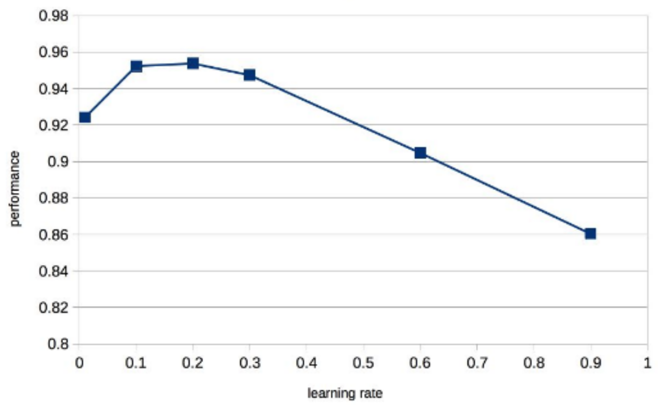

Fig. 7. The graph of the performance of neural networks with respect to the learning rate defined.

It should be noted that the number of hidden nodes can have a significant effect on the result but even it has a threshold after which no observable change in result is shown even after the increment in the number of hidden nodes.

The further improvement which can be done is by training the network with the same datasets repeatedly, This is defined as number of epochs. By running a network through 10 epochs, the network runs iteratively over the same datasets and trains itself. By collaborating this with the effects of learning rate, considerable advancement in the efficiency can be achieved. The following Fig. 8 of graph shows the increase in efficiency for the number of epochs run by the network.

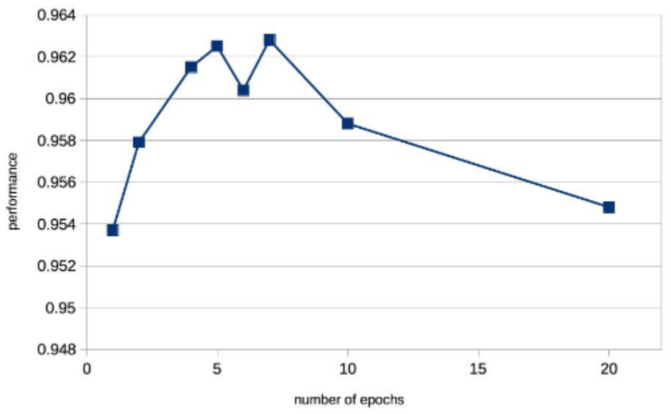

Fig. 8. The graph of performance of neural network with respect to number of epochs

V. Comparision of Efficiency in Activation Functions

The accuracy of neural network is highly dependent on the choice of activation function. Different activation functions have varied effect on the neural network since it is the major filtering device of the network. After running courses of iterations using various activation functions and keeping the constant epochs and learning rate. The comparison of outcome is given below in the table 1

TABLE I
Comparison of efficiency in various activation function

Activation function	**Efficiency in %**
Sigmoid function	97%
Hyperbolic Tangent function	92%
Arctangent function	86%
Unit function	46%

Sigmoid function is the most natural selection of most of the neural network including the handwritten digit recognition network since it has the property of smoothening the outcome of nodes and regulate the fluctuation. Hyperbolic tangent function and inverse tangent function is quite similar and has the moderate efficiency in recognizing the given digit. Unit function is the most basic function available in nature which has only two outcomes of 0 and 1. It has the least efficiency.

VI. Conclusion

This paper concludes that the sigmoid function has the most efficiency in recognizing the hand written digits and proves the capability of neural networks mathematically by deriving the error propagation methods with the help of calculus and gradient descent.

VII. References

[1] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5:115–133

[2] Paul Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, 1974

[3] Paul Werbos, Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, Volume 78, Issue 10, 1550–1560, Oct 1990, doi10.1109/5.58337

[4] Yann LeCun. Retrived from www.yann.lecun.com/exdb/mnist/