Journal of Theoretical and Applied Information Technology
31st January 2013. Vol. 47 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ANALYSIS OF DIFFERENT ACTIVATION FUNCTIONS USING BACK PROPAGATION NEURAL NETWORKS

[1] P.SIBI, [2] S.ALLWYN JONES, [3] P.SIDDARTH
[1,2,3] Student, SASTRA University, Kumbakonam, India
E-mail: [1]sibi@psibi.in, [2]sallwynjones@gmail.com, [3]psiddarthkey2008@gmail.com

ABSTRACT

The Back propagation algorithm allows multilayer feed forward neural networks to learn input/output mappings from training samples. Back propagation networks adapt itself to learn the relationship between the set of example patterns, and could be able to apply the same relationship to new input patterns. The network should be able to focus on the features of an arbitrary input. The activation function is used to transform the activation level of a unit (neuron) into an output signal. There are a number of common activation functions in use with artificial neural networks (ANN). Our paper aims to perform analysis of the different activation functions and provide a benchmark of it. The purpose is to figure out the optimal activation function for a problem.
Keywords: *Artificial Neural Network (ANN), Back Propagation Network (BPN), Activation Function*

1. INTRODUCTION

A neural network is called a mapping network if it is able to compute some functional relationship between its input and output. For example, if the input to a network is the value of an angle, and the output is the cosine of the angle, the network performs the mapping θ→ cos(θ). Suppose we have a set of P vector pairs $(x_{1,}y_1),(x_{2,}y_2),...,(x_p,y_p)$ which are examples of a functional mapping $y=\varphi(x): x \in R^N, y \in R^M$. We have to train the network so that it will learn an approximation $o = y' = \varphi'(x)$. It should be noted that learning in a neural network means finding an approximate set of weights.

Function approximation from a set of input-output pairs has numerous scientific and engineering applications. Multilayer feed forward neural networks have been proposed as a tool for nonlinear function approximation [1], [2], [3]. Parametric models represented by such networks are highly nonlinear. The back propagation (BP) algorithm is a widely used learning algorithm for training multilayer networks by means of error propagation via variational calculus [4], [5]. It iteratively adjusts the network parameters to minimize the sum of squared approximation errors using a gradient descent technique. Due to the highly nonlinear modeling power of such networks, the learned function may interpolate all the training points. When noisy training data are present, the learned function can oscillate abruptly between data points. This is clearly undesirable for function approximation from noisy data.

2. BACK PROPAGATION NETWORK MECHANISM

Apply the input vector to the input units. Input vector is

$$X_p = (x_{p1}, x_{p2}, ..., x_{pN})^t$$

where X_p is the input vector.

Calculate the net input values to the hidden layer units:

$$net_{pj}^h = \sum_{(i=1)}^{N} w_{ji}^h x_{pi} + \theta_j^h$$

where net_{pj}^h is the net input to hidden layer , w_{ji}^h is the weight on the connection from i^{th} input unit θ_j^h is the bias term and "h" refers to quantities on the hidden layer.

Calculate the outputs from the hidden layer:

$$i_{pj} = f_j^h(net_{pj}^h)$$

where i_{pj} is the output from hidden layer and

f_j^h is the activation function.

Journal of Theoretical and Applied Information Technology
31st January 2013. Vol. 47 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Move to the output layer. Calculate the net-input values to each units:

$$net^o_{pk} = \sum_{(j=1)}^{L} w^o_{kj} i_{pj} + \theta^o_k$$

where net^o_{pk} is the net input to the output layer, w^o_{kj} is the weight in the connection from j^{th} hidden unit, θ^o_k is the bias term and "o" refers to quantities on the output layer.

Calculate the outputs:

$O_{pk} = f^o_k\left(net^o_{pk}\right)$ where O_{pk} is the output got from the output layer

Calculate the error terms for the output units

$\delta^o_{pk} = \left(y_{pk} - O_{pk}\right) f^{o\prime}_j\left(net^o_{pk}\right)$ where δ^o_{pk} is the error at each output unit,

$\delta^o_{pk} = y_{pk} - o_{pk}$ where y_{pk} is the desired error and o_{pk} is the actual error

Calculate the error terms for hidden units:

$\delta^h_{pj} = f^{h\prime}_j\left(net^h_{pj}\right) \sum_k \delta^o_{pk} w^o_{kj}$ where δ^h_{pj} is the error at each hidden unit

Notice that the error terms on the hidden units are calculated before the connection weights to the output-layer units have been updated.

Update weights on the output layer:

$$w^o_{kj}(t+1) = w^o_{kj}(t) + \eta \delta^o_{pk} i_{pj}$$

Update weights on the hidden layer:

$$w^h_{ji}(t+1) = w^h_{ji}(t) + \eta \delta^h_{pj} x_i$$

where η is the learning rate parameter. The order of the weight updates on an individual layer is not important. Be sure to calculate the error term

$$E_p = 1/2 \sum_{(k=1)}^{M} \delta^2_{pk}$$

since this quantity is the measure of how well the network is learning. When the error is acceptably small for each of the training-vector pairs, training can be discontinued [6]. The network for back propagation is illustrated in Figure 1.

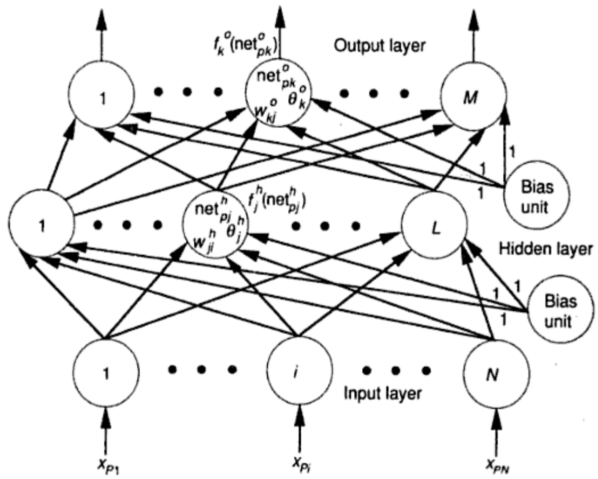

Figure 1

3. ACTIVATION FUNCTION TYPES

Every neuron model consists of a processing element with synaptic input connections and a single output. The signal flow of neuron inputs, x_i, is considered to be unidirectional [7]. The neuron output signal is given by the relationship o=f(∑), which is illustrated in Figure 2

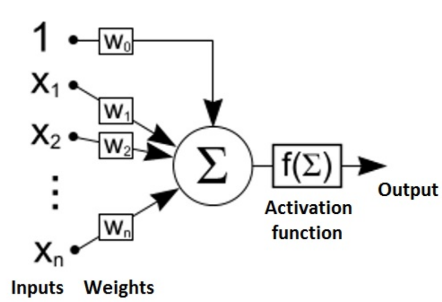

Figure 2

The functions are described with parameters where

- x is the input to the activation function,
- y is the output,
- s is the steepness and
- d is the derivation.

3.1 Linear Activation Function

The linear activation function will only produce positive numbers over the entire real number range.

$span: -\infty < y < \infty, y = x * s, \quad d = 1 * s,$

Cannot be used in fixed point.

Journal of Theoretical and Applied Information Technology
31st January 2013. Vol. 47 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.2 Sigmoid Activation Function

The sigmoid function will only produce positive numbers between 0 and 1. The sigmoid activation function is most useful for training data that is also between 0 and 1. It is one of the most used activation functions.

$$span: 0 < y < 1, \quad y = 1/(1+\exp(-2*s*x)),$$
$$d = 2*s*y*(1-y)$$

3.3 Sigmoid Stepwise Activation Function

The stepwise sigmoid activation function is a piecewise linear approximation of the usual sigmoid function with output between zero and one. It is faster than sigmoid but a bit less precise.

3.4 Sigmoid Symmetric Activation Function

The symmetrical sigmoid activation function is the usual tanh sigmoid function with output between minus one and one. It is one of the most used activation functions.

$$span: -1 < y < 1$$
$$y = \tanh(s*x) = 2/(1+\exp(-2*s*x)) - 1$$
$d = s*(1-(y*y))$ where tanh is tangent hyperbolic function.

3.5 Sigmoid Symmetric Stepwise Activation Function

The symmetrical sigmoid activation function is a piecewise linear approximation of the usual tanh sigmoid function with output between minus one and one. It is faster than symmetric sigmoid but a bit less precise.

3.6 Gaussian Activation Function

Gaussian activation function can be used when finer control is needed over the activation range. The output range is 0 to 1: 0 when x=∞ and 1 when x=0.

$$span: 0 < y < 1, \qquad y = \exp(-x*s*x*s),$$
$$d = -2*x*s*y*s$$

3.7 Gaussian Symmetric Activation Function

Gaussian symmetric activation function can be used when finer control is needed over the activation range. The output range is -1 to 1: -1 when x= -∞, 1 when x=0, 0 when x=∞.

$$span: -1 < y < 1,$$
$$y = \exp(-x*s*x*s)*2-1,$$
$$d = -2*x*s*(y+1)*s$$

3.8 Elliot Activation Function

The Elliott Activation Function is higher-speed approximation of the Hyperbolic Tangent Activation Function. The output range is 0 to 1.

$$span: 0 < y < 1 ,$$
$$y = ((x*s)/2)/(1+|x*s|)+0.5,$$
$$d = s*1/(2*(1+|x*s|)*(1+|x*s|))$$

3.9 Elliot Symmetric Activation Function

The Elliot symmetric activation function is higher speed approximation of Sigmoid activation functions. The output range is -1 to 1.

$$span: -1 < y < 1, \; y = (x*s)/(1+|x*s|),$$
$$d = s*1/((1+|x*s|)*(1+|x*s|))$$

3.10 Linear Piecewise Activation Function

This activation function is also called saturating linear function and can have either a binary or bipolar range for the saturation limits of the output. the output range is 0 to 1.

$$span: 0 < y < 1, y = x*s, d = 1*s$$

3.11 Linear Piece Symmetric Activation Function

This activation function is also called saturating linear function and can have either a binary or bipolar range for the saturation limits of the output. the output range is -1 to 1.

$$span: -1 < y < 1, y = x*s \;, d = 1*s$$

4. EXPERIMENTAL RESULTS

A dataset was chosen for evaluation of the activation network. A simulator was specially developed for testing the activation function using an open source library fann (Fast Artifical Neural Network). The simulator was written in Python and language bindings for fann was used which itself was created using SWIG (Simplified Wrapper Interface Generator). The dataset chosen for analysis is mushroom data. The mushroom classification problem is to determine whether a mushroom is edible or poisonous based on its observable features . The 22 input features were converted into 125 binary attributes. The input features of the dataset is represented in Table 1.

Journal of Theoretical and Applied Information Technology
31st January 2013. Vol. 47 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1

Features	Values
Cap-shape	bell/conical/convex/flat/ knobbed/sunken
Cap-surface	fibrous/grooves/scaly/smooth
Cap-color	brown/buff/cinnamon/gray/green/ pink/purple/red/white/yellow
Bruises	true/false
Odor	almond/anise/creosote/fishy/foul/ musty/none/pungent/spicy
Gill-attachment	attached/descending/free/notched
Gill-spacing	close/crowded/distant
Gill-size	broad/narrow
Gill-color	black/brown/buff/chocolate/gray/ green/orange/pink/purple/red/white/yellow
Stalk-shape	enlarging/tapering
Stalk-root	bulbous/club/cup/equal/ rhizomorphs/rooted/missing
Stalk-surface-above-ring	fibrous/scaly/silky/smooth
Stalk-surface-below-ring	fibrous/scaly/silky/smooth
Stalk-color-above-ring	brown/buff/cinnamon/gray/ orange/pink/red/white/yellow
Stalkcolor-below-ring	brown/buff/cinnamon/gray/ orange/pink/red/white/yellow
Veil-type	partial/universal
Veil-color	brown/orange/white/yellow
Ring-number	none/one/two
Ring-type	cobwebby/evanescent/flaring /large/none/pendant/sheathing/zone
Spore-print-color	black/brown/buff/chocolate/green /orange/purple/white/yellow
Population	abundant/clustered/numerous/ scattered/several/solitary
Habitat	grasses/leaves/meadows/paths/ urban/waste/woods

5. PERFORMANCE EVALUATION

Training activity was carried out in mushroom dataset with an expected error of 0.0999. The algorithm used for training was RPROP (Resilient Propagation). The increase factor and decrease factor for the algorithm was chosen the optimal value of 1.2 and 0.5 respectively. The delta min value was taken as 0 and the delta max value as 50. The number of hidden layers for the network was 3 with 4, 5 and 5 neurons in each layer respectively. The result obtained by the simulation is illustrated in Table 2.

Table 2

Evaluation of Mushroom dataset			
Activation Function	**Total Number of Epochs**	**Error at Last Epoch**	**Bit Fail at Last Epoch**
LINEAR	47	0.0063356720	21
SIGMOID	30	0.0003930641	4
SIGMOID STEPWISE	41	0.0007385524	6
SIGMOID STEPWISE SYMMETRIC	26	0.0095451726	50
GAUSSIAN	50	0.0079952301	24
GAUSSIAN SYMMETRIC	21	0.0063603432	8
ELLIOT	22	0.0096499957	6
ELLIOT SYMMETRIC	42	0.0090665855	125
LINEAR PIECE	71	0.0095399031	90
LINEAR PIECE SYMMETRIC	28	0.0084868055	110
SIN SYMMETRIC	33	0.0087634288	64
COS SYMMETRIC	49	0.0061022025	48

6. CONCLUSION:

Activation function is one of the essential parameter in a Neural Network. The performance evaluation of different activation functions shows up that there is a not a huge difference between them. When a network gets trained up successfully, every activation function has approximately the same effect on it. The paper clearly shows up to which extent an activation function is important. Selection of an activation function for a network or it's specific nodes is an important task. But as the results show, if a network gets trained up successfully with a particular activation function, then there is a high probability that other activation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

functions will also lead to proper training of the neural network.

We emphasize that although selection of an activation function for a neural network or it's node is an important task, other factors like training algorithm, network sizing and learning parameters are more vital for proper training of the network as the results by the simulator shows us that there is only a trivial differences between training when configured with different activation functions.

REFERENCES:

[1] K. Homik, M. Stinchcombe, and H. White, “Multilayer feed forward networks are universal approximators”, *Neural Networks*, vol. 2, pp.359-366, 1989.
[2] C. Ji, R. R. Snapp, and D. Psaltis, “Generalizing smoothness constraints from discrete samples”, *Neural Computation*, vol. 2, pp. 188-197, 1990.
[3] T. Poggio, and F. Girosi, “Networks for approximation and learning.” *Proc. IEEE*. vol. 78, no. 9, pp. 1481-1497. 1990.
[4] Y. Le Cun, “A theoretical framework for back propagation”, in Proc.*1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. June 17-26, 1988. San Mateo, CA: Morgan Kaufmann, pp. 21-28.
[5] D. E. Rummelhart, G. E. Hinton. and R. J . Williams. “Parallel Distributed Processing: Explorations in the Microstructure of Cognition”, vol. I. MIT Press, ch. 8.
[6] James A. Freeman and David M. Skapura. “Neural Networks Algorithms, Applications and Programming Techniques”, pp 115-116, 1991.
[7] Jacek M. Zurada, "Introduction to Artificial Neural Systems", pp 32-36, 2006.