Back Propagation Neural networks(BPNN) and Sigmoid Activation Function in Multi-Layer Networks

Renas Rajab Asaad, Rasan Ismael Ali
Department of Computer Science, Nawroz University, Duhok, Kurdistan Region – Iraq

ABSTRACT
Back propagation neural networks are known for computing the problems that cannot easily be computed (huge datasets analysis or training) in artificial neural networks. The main idea of this paper is to implement XOR logic gate by ANNs using back propagation neural networks for back propagation of errors, and sigmoid activation function. This neural networks to map non-linear threshold gate. The non-linear used to classify binary inputs ($x1, x2$) and passing it through hidden layer for computing $coefficient\_errors$ and $gradient\_errors$ ($Cerrors, Gerrors$), after computing errors by ($ei = Output\_desired - Output\_actual$) the weights and thetas ($\Delta Wji = (\alpha)(Xj)(gi), \Delta\theta j = (\alpha)(-1)(gi)$) are changing according to errors. Sigmoid activation function is = $sig(x) = 1/(1 + e - x)$ and Derivation of sigmoid is = $dsig(x) = sig(x)(1 - sig(x))$. The sig(x) and Dsig(x) is between 1 to 0.
Keywords: Artificial Neural networks, Sigmoid Function, Backpropagation ANNs, Neural networks.

1. Introduction

Solving mathematics cases and algorithms are easy to define and compute from computer. But the science nowadays can't easily deal with facial recognition, language processing and every huge datasets, and these aren't easily quantified into an algorithm. The key to ANN is process the data in a same way to our bio-brains. An artificial neural networks is a system is inspired on the biological neural networks, as our brain [1].
Multi-Layer Neural networks, are neural networks in which input neurons pass signals and information to other processing elements inside a hidden layer, afterwards information is passed to the output neurons. These are called back propagation neurons, which usually solve complex problems.
Building back propagation neural networks in digital logic gates where the where an input is passed to the hidden layers between and passed to the output layer where an output is generated. To make logical decisions with Boolean logics are electronic device according to different merges of signals shows on its inputs. Digital logic gates mat contains more than one inputs and passing through hidden layer to generate more than one output and the process will continue to finally generate the output with lower errors by calculating (error coefficient, error gradient, weights, theta, desired and actual output, and derivation of sigmoid activation function in hidden layer) [2].

Academic Journal of Nawroz University (AJNU) Volume 8, No 4 (2019).
Regular research paper : Published 20 Nov 2019
Corresponding author's e-mail : renas_rekany@yahoo.com

doi : 10.25007/ajnu.v8n4a464

2. A Biological Neuron

A biological neuron is most basic information processing unit in the nervous system. a biological neuron consists of the following parts: (Dendrites (input), Cell body, Axon (output)). A biological neuron takes signals from it's dendrites and processes the signal and outputs a signal from it's axon based on the input signal [5].

3. Artificial Neural networks

A neural networks is a group of connected I/O units where each connection has a weight associated with its computer programs. It helps you to build predictive models from large databases. This model builds upon the human nervous system. It helps you to conduct

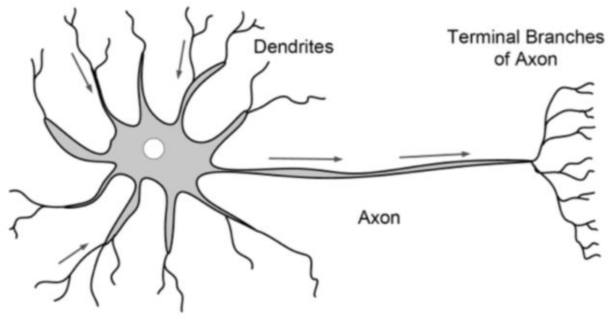

image understanding, human learning, computer speech, etc [9].

Fig 1: Structure of a Typical Neuron [5]

A model of ANNs:

a. Inputs X, arrive through the preconnected path.
b. Input is modeled using real weights W. The weights are usually randomly selected.
c. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
d. Calculate the error in the outputs.
   Error= Actual Output – Desired Output
e. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

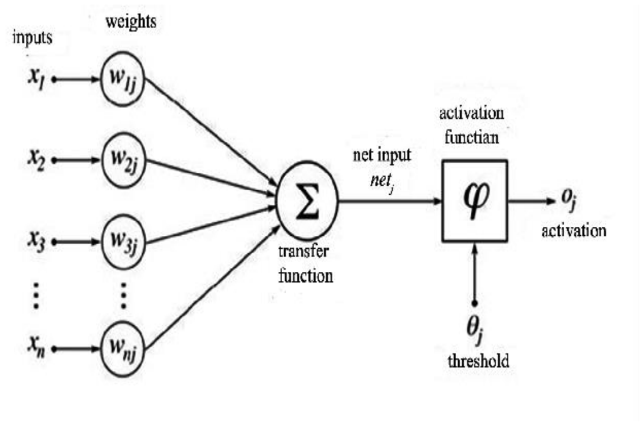

Fig 2: Artificial Neural networks Model [3]

3.1 Advantages of Backpropagation

1 Backpropagation is fast, simple and easy to program.
2 It has no parameters to tune apart from the numbers of input.
3 It is a flexible method as it does not require prior knowledge about the network.
4 It is a standard method that generally works well.
5 It does not need any special mention of the features of the function to be learned.

4. Implementation of Back Propagation Neural networks

Back propagation is shorthand for back propagation of errors, because in back propagation neural networks, the error factor must be propagated to the responsible nodes. A back propagation neural networks is a multi-layer network where an input is passed to the hidden layers between and passed to the output layer where an output is generated. Back propagation neural networks are used to map non-linear classifiers, in which outputs belong to different classes. These neural networks are trained using supervised or non-supervised learning methods, and is usually provided with the sigmoid activation function [4].

4.1 The Sigmoid Activation Function

The main problem with the step activation function is that it is non-differentiable. therefor it cannot be used to calculate error coefficients in a back propagation neural networks [6].

The sigmoid activation function is used instead.

$$sig(x) = 1/(1+e-x)$$

$$sig(\infty) = 1 \quad e^\wedge(-\infty) = 0$$

$$sig(0.5) = 0.622$$

$$sig(0) = 0.5\ e0 = 1$$

$$sig(-0.5) = 0.377$$

$$sig(-\infty) = 0\ e^\wedge\infty = \infty$$

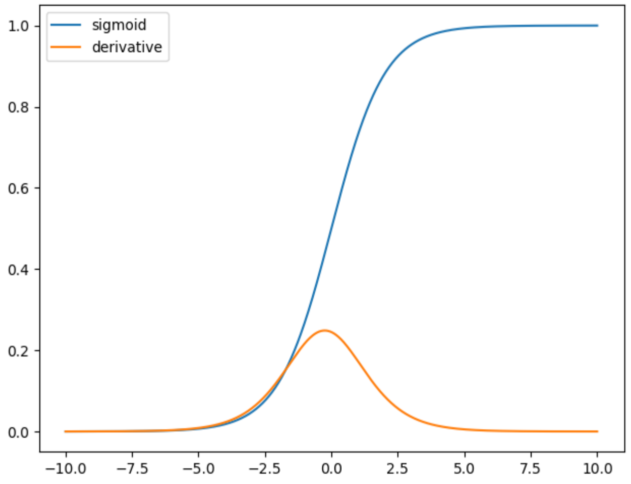

Fig 3: Sigmoid Function

4.2 Derivative of The Sigmoid Activation Function

The derivative of the Sigmoid function is used to calculate the error coefficient, which is used to propagate the error back on the neural networks [8].

$$\boldsymbol{dsig(x) = sig(x)(1 - sig(x))\ ....(1)}$$

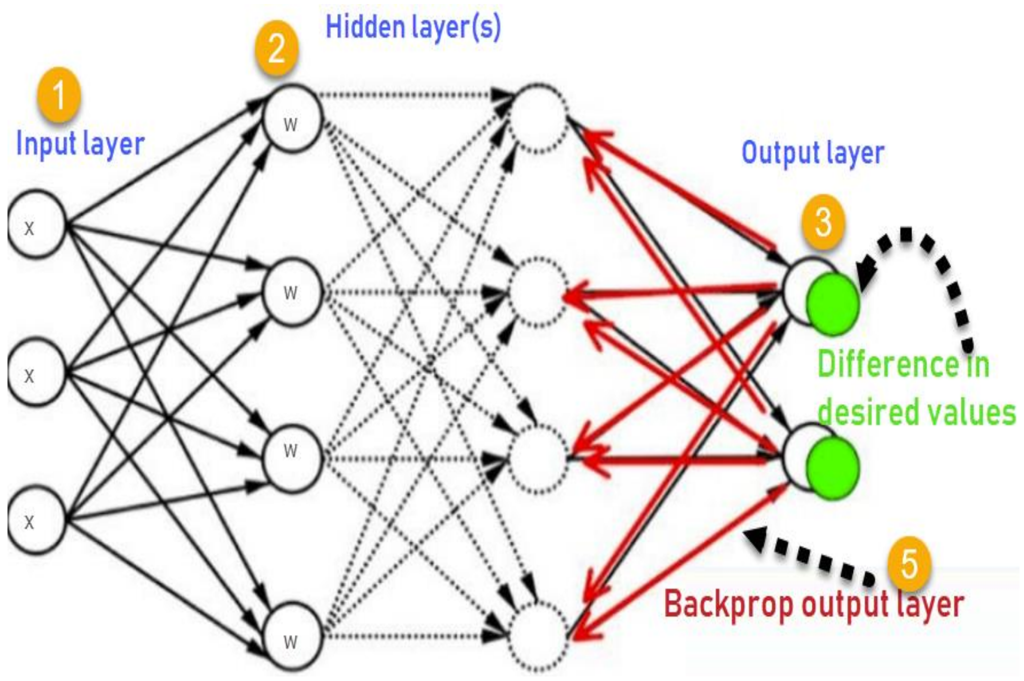

Fig 4: Multi-layer Perceptron

4.3 Back propagation Learning

Back propagation learning is based on calculating the error coefficient using the below equation in the output layer [7]:

$$ei = output\_desired - output\_actual \text{.....(2)}$$

then we calculate the error gradient (g) at the output layer.

$$gi = Dsig(Yi)(ei) \text{ .....(3)}$$

Afterwards, we update the network weights and thetas at the output layer, based on the below equation:

$$\Delta Wji = (\alpha)(Yj)(gi) \text{ .....(4)}$$
$$\Delta \theta i = (\alpha)(-1)(gi) \text{ .....(5)}$$

at the hidden layers, the error gradient is calculated using the below equation:

$$gj = Dsig(Yj)(Wji)(gi) \text{ .....(6)}$$

Afterwards, we update the network weights and thetas, based on the below equation:

$$\Delta Wji = (\alpha)(Xj)(gi) \text{ .....(7)}$$
$$\Delta \theta j = (\alpha)(-1)(gi) \text{ .....(8)}$$

the deltas acquired are added as below:

$$Wj = Wj + \Delta W \text{ .....(9)}$$
$$\theta j = \theta j + \Delta \theta \text{ .....(10)}$$

4.4 XOR gate using back propagation neural networks

The input matrix is The output matrix is

$p = \{0\,0, 01, 10, 11\}$ $t = \{0, 1, 1, 0\}$

Bias1(Θ_1), Bias2(Θ_3), and Bias3(Θ_3) is equal to -1.

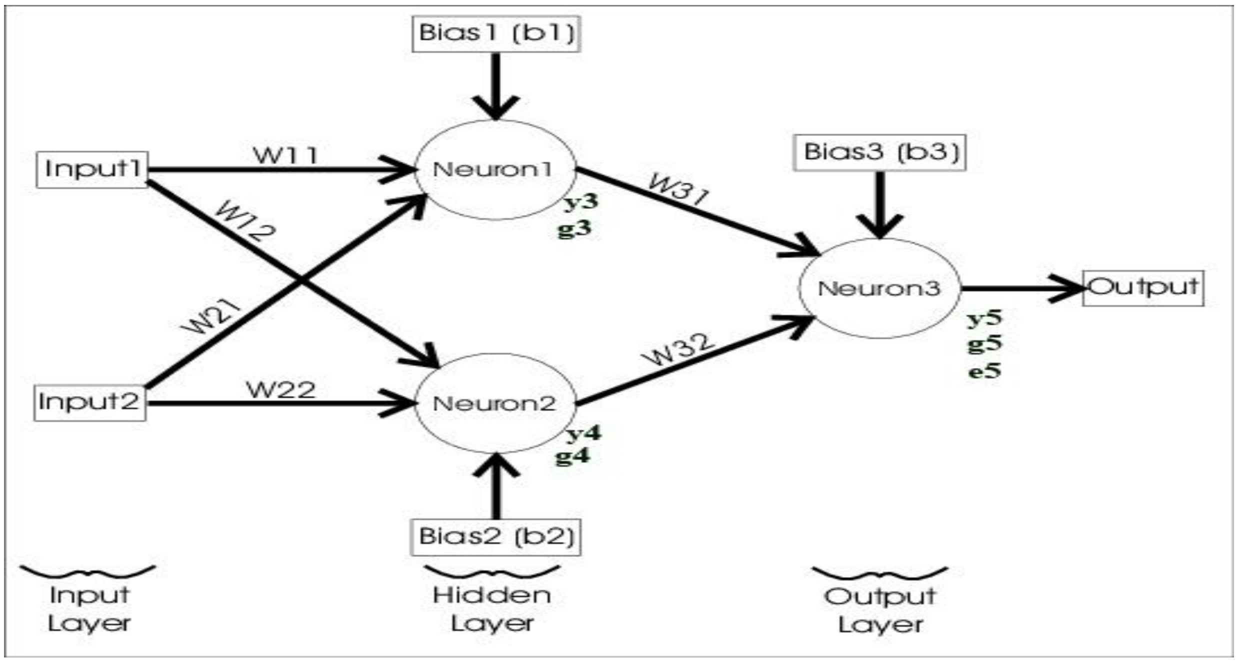

Fig 5: XOR Gate with Weights and Errors

Table 1: List of Abbreviations

Abbreviations	Variables Name
ei	Error Coefficient
gi	Error Gradient
ΔWji	The Changes in Weights
$\Delta\theta j$	The Changes in Theta
p	Perceptron
t	Target
$Outputi\_desired$	Desired Output
$Outputi\_actual$	Actual Output
sig	Sigmoid Activation Function
$Dsig$	Derivative of Sigmoid Activation Function
net_i	The Networks
y_i	The Network's Output
α	Alpha
Xi	Network's Input
Yi	Network's Input
ei	Network's Error

5. Results and Implementations

The result shows that XOR logic gate trained in hidden layer and the results compute two iterations in first epoch.

$$Assume\ W11 = 0.5, W12 = 0.9, W21 = 0.4, W22 = 1, W31 = -1.2, W32 = 1.1, \theta 1 = 0.8, \theta 2 = -0.1, \theta 3 = 0.3, \alpha = 0.1.$$

So:

$$net3 = (x1 * w11 + x2 * w21) - \theta 1$$

$$= (1 * 0.5 + 1 * 0.4) - 0.8 \quad =0.1$$

$$y3 = sig(net3)$$

$$= 0.5249$$

$net4 = (x1 * w12 + x2 * w22) - \theta 2$

= (1 * 0.9 + 1 * 1) - (-0.1) =2

$y4 = sig(net4)$

= 0.8808

$net5 = (y3 * w31 + y4 * w32) - \theta 3$

= (0.5249 * -1.2 + 0.8808 * 1.1) - 0.3 = (-0.62988 * 0.96888) - 0.3 = 0.039

$y5 = sig(net5)$

= 0.5097

$e5 = yd - yact$

= 0 - 0.5097 = -0.5097

$g5 = dsig(y5) * e5$

$= sig(y5) * (1 - sig(y5)) * e5$

= 0.5097 * (1 - 0.5097) * -0.5097 = -0.1274

$g3 = dsig(y3) * w31 * g5$

$= sig(y3) * (1 - sig(y3)) * w35 * g5$

= 0.5249 * (1 - 0.5249) * -1.2 * -0.1274 = 0.0381

$g4 = dsig(y4) * w32 * g5$

$= sig(y4) * (1 - sig(y4)) * w45 * g5$

= 0.8808 * (1 - 0.8808) * 1.1 * -0.1274 = -0.0147

$\Delta w11 = \alpha\, x1\, g3$

= 0.1 * 1 * 0.0381 = 0.00381

$\Delta w12 = \alpha\, x1\, g4$

= 0.1 * 1 * -0.0147 = -0.00147

$\Delta w21 = \alpha\, x2\, g3$

= 0.1 * 1 * 0.0381 = 0.00381

$\Delta w22 = \alpha\, x2\, g4$

= 0.1 * 1 * -0.0147 = -0.00147

$\Delta w31 = \alpha\, y3\, g5$

= 0.1 * 0.5249 * -0.1274 = -0.0066

$\Delta w32 = \alpha\, y4\, g5$

= 0.1 * 0.8808 * -0.1274 = -0.0112

$\Delta \theta 1 = \alpha\, (-1)\, g3$

= 0.1 * -1 * 0.0381 = -0.00381

$\Delta \theta 2 = \alpha\, (-1)\, g4$

= 0.1 * -1 * -0.0147 = 0.00147

$\Delta \theta 3 = \alpha\, (-1)\, g5$

= 0.1 * -1 * -0.1274 = 0.01274

$w11 = w11 + \Delta w11$

= 0.5 + 0.00381 = 0.50381

$w12 = w12 + \Delta w12$

= 0.9 + -0.00147 = 0.89853

$w21 = w21 + \Delta w21$

= 0.4 + 0.00381 = 0.40381

$w22 = w22 + \Delta w22$

= 1 + -0.00147 = 0.99853

$w31 = w31 + \Delta w31$

= -1.2 + -0.0066 = -1.2066

$w32 = w32 + \Delta w32$

= 1.1 + -0.0112 = 0.0888

$\theta 1 = \theta 1 + \Delta \theta 1$

= 0.8 + -0.00381 = 0.79619

$\theta 2 = \theta 2 + \Delta \theta 2$

= -0.1 + 0.00147 = -0.09853

$\theta 3 = \theta 3 + \Delta \theta 3$

= 0.3 + 0.01274 = 0.31274

6. Conclusion

This paper implemented XOR logic gate in a multi-layer artificial neural networks by passing in activation function and used to map non-linear classifiers (x_1, x_2) as an inputs then through the hidden layer by computing errors. This study explain and implement a back propagation neural networks for back propagation of errors (coefficient and gradient error) to generalize the output. These trainings neural networks are used supervised or non-supervised learning methods by calculating the sigmoid activation function were realized.

References

[1] Jacobson, L. (2013). Introduction to Artificial Neural networks.
[2] Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., & dos Reis Alves, S. F. (2017). Artificial neural networks. Cham: Springer International Publishing.
[3] Alan P. O. et al (). Adaptive Boolean Logic Using Ferroelectrics Capacitors as Basic Units of Artificial Neurons. www.electronic-tutorials.ws. [Accessed Date: Nov 2019].
[4] Sibi, P., S. Allwyn Jones, and P. Siddarth. (2013). "Analysis of different activation functions using back propagation neural networks." Journal of Theoretical and Applied Information Technology 47.3: 1264-1268.
[5] Li, H., Zhang, Z., & Liu, Z. (2017). Application of artificial neural networks for catalysis: a review. Catalysts, 7(10), 306.
[6] haq Shaik, E., & Rangaswamy, N. (2017). Multi-mode interference-based photonic crystal logic gates with simple structure and improved contrast ratio. Photonic Network Communications, 34(1), 140-148.
[7] Wanto, A., Windarto, A. P., Hartama, D., & Parlina, I. (2017). Use of Binary Sigmoid Function And Linear Identity In Artificial Neural networks For Forecasting Population Density. International Journal Of Information System & Technology, 1(1), 43-54.
[8] Chung, H., Lee, S. J., & Park, J. G. (2016, July). Deep neural networks using trainable activation functions. In 2016 International Joint Conference on Neural networks (IJCNN) (pp. 348-352). IEEE.
[9] Bhattacherjee, A., Roy, S., Paul, S., Roy, P., Kausar, N., & Dey, N. (2020). Classification approach for breast cancer detection using back propagation neural networks: a study. In Deep Learning and Neural networks: Concepts, Methodologies, Tools, and Applications (pp. 1410-1421). IGI Global.