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 (ΔWji = (a)(k)(j)(gj), Δθj = (a)(−1)(gj)) are changing according to errors. Sigmoid activation function is = sig(x) = 1/(1 + e − x) and Derivation of sigmoid is = d.sig(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].

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.

\[ \text{Error} = \text{Actual Output} - \text{Desired Output} \]

e. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

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. Therefore it cannot be used to calculate error coefficients in a back propagation neural networks [6]. The sigmoid activation function is used instead.

\[
\text{sig} \(x\) = \frac{1}{1 + e^{-x}}
\]

\[
\text{sig}(\infty) = 1 \quad e^{\infty} = 0
\]

\[
\text{sig}(0) = 0.622
\]

\[
\text{sig}(0.5) = 0.377
\]

\[
\text{sig}(-0.5) = 0 \quad e^{-\infty} = 0
\]

\[
\text{sig}(-\infty) = 0 \quad e^{\infty} = \infty
\]

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].

\[
d\text{sig}(x) = \text{sig}(x)(1 - \text{sig}(x)) \quad \text{....}(1)
\]
4.3 Back propagation Learning

Back propagation learning is based on calculating the error coefficient using the below equation in the output layer [7,8,9]:

\[ e_i = output_{desired} - output_{actual} \] ....(2)

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

\[ g_i = Dsig(Y_i)(e_i) \] .....(3)

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

\[ \Delta W_{ji} = (\alpha)(X_j)(g_i) \] .....(4)
\[ \Delta \theta_i = (\alpha)(-1)(g_i) \] .....(5)

the deltas acquired are added as below:

\[ W_j = W_j + \Delta W \] .....(9)
\[ \theta_j = \theta_j + \Delta \theta \] .....(10)

4.4 XOR gate using back propagation neural networks

The input matrix is

\[ p = \{ 0 0, 01, 10, 11 \} \]

The output matrix is

\[ t = \{ 0, 1, 1, 0 \} \]

Bias1(\( \Theta_1 \)), Bias2(\( \Theta_2 \)), and Bias3(\( \Theta_3 \)) is equal to -1.
Table 1: List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviations</th>
<th>Variables Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ei</td>
<td>Error Coefficient</td>
</tr>
<tr>
<td>gi</td>
<td>Error Gradient</td>
</tr>
<tr>
<td>ΔWji</td>
<td>The Changes in Weights</td>
</tr>
<tr>
<td>Δθj</td>
<td>The Changes in Theta</td>
</tr>
<tr>
<td>p</td>
<td>Perceptron</td>
</tr>
<tr>
<td>t</td>
<td>Target</td>
</tr>
<tr>
<td>Outputi_desired</td>
<td>Desired Output</td>
</tr>
<tr>
<td>Outputi_actual</td>
<td>Actual Output</td>
</tr>
<tr>
<td>sig</td>
<td>Sigmoid Activation Function</td>
</tr>
<tr>
<td>Dsig</td>
<td>Derivative of Sigmoid Activation Function</td>
</tr>
<tr>
<td>neti</td>
<td>The Networks</td>
</tr>
<tr>
<td>yi</td>
<td>The Network’s Output</td>
</tr>
<tr>
<td>α</td>
<td>Alpha</td>
</tr>
<tr>
<td>Xi</td>
<td>Network’s Input</td>
</tr>
<tr>
<td>Yi</td>
<td>Network’s Input</td>
</tr>
<tr>
<td>ei</td>
<td>Network’s Error</td>
</tr>
</tbody>
</table>

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:

\[
net3 = (x1 \cdot w11 + x2 \cdot w21) - \theta_1 \\
= (1 \cdot 0.5 + 1 \cdot 0.4) - 0.8 \\
= 0.1
\]

\[
y3 = \text{sig}(net3) \\
= 0.5249
\]
\[ \text{net4} = (x1 \cdot w12 + x2 \cdot w22) - \theta2 \]
\[ = (1 \cdot 0.9 + 1 \cdot 1) - (-0.1) = 2 \]
\[ y4 = \text{sig}(\text{net4}) \]
\[ = 0.8808 \]
\[ \text{net5} = (y3 \cdot w31 + y4 \cdot w32) - \theta3 \]
\[ = (0.5249 \cdot -1.2 + 0.8808 \cdot 1.1) \cdot 0.3 = (-0.62988 \cdot 0.96888) \]
\[ - 0.3 = 0.039 \]
\[ y5 = \text{sig}(\text{net5}) \]
\[ = 0.5097 \]
\[ e5 = yd - yact \]
\[ = 0 \cdot 0.5097 = -0.5097 \]
\[ g5 = d\text{sig}(y5) \cdot e5 \]
\[ = \text{sig}(y5) \cdot (1 - \text{sig}(y5)) \cdot e5 \]
\[ = 0.5097 \cdot (1 - 0.5097) \cdot -0.5097 = -0.1274 \]
\[ g3 = d\text{sig}(y3) \cdot w31 \cdot g5 \]
\[ = \text{sig}(y3) \cdot (1 - \text{sig}(y3)) \cdot w35 \cdot g5 \]
\[ = 0.5249 \cdot (1 - 0.5249) \cdot -1.2 \cdot -0.1274 = 0.0381 \]
\[ g4 = d\text{sig}(y4) \cdot w32 \cdot g5 \]
\[ = \text{sig}(y4) \cdot (1 - \text{sig}(y4)) \cdot w45 \cdot g5 \]
\[ = 0.8808 \cdot (1 - 0.8808) \cdot 1.1 \cdot -0.1274 = -0.0147 \]
\[ \Delta w11 = \alpha x1 g3 \]
\[ = 0.1 \cdot 1 \cdot 0.0381 = 0.00381 \]
\[ \Delta w12 = \alpha x1 g4 \]
\[ = 0.1 \cdot 1 \cdot -0.0147 = -0.00147 \]
\[ \Delta w21 = \alpha x2 g3 \]
\[ = 0.1 \cdot 1 \cdot 0.0381 = 0.00381 \]
\[ \Delta w22 = \alpha x2 g4 \]
\[ = 0.1 \cdot 1 \cdot -0.0147 = -0.00147 \]

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 \((x1, x2)\) 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

doi : 10.25007/ajnu.v8n4a464
are used supervised or non-supervised learning methods by calculating the sigmoid activation function were realized.

References