Neural Network

Introduction

Nowadays scientists are trying to find power of human brain. They try to imitate it for finding big data solution.

I feel that there is no comprehensive, easy, clear and practical article on NN. I always wanted to know how the human brain works biologically. I had many questions that remained unanswered. Neural Network details was always ambiguous for me. The most important questions that I want to answer:

1. How human brain works exactly?
2. How perceptron as an artificial neuron works - Forward Neural Network?
3. What is weight in Neural Network?
    a. what
4. What is equivalent for weight in biological neuron?
5. What is Activation Function role in Neural Network?
6. What is equivalent for Activation Function in biological neuron?
7. How backward propagation works?
8. What is the exact mathematical logic for backward propagation neural network?
9. How to implement backward propagation neural network?

1. How human brain works exactly?

For understanding that how neural network works, it is better to study about human brain operation. There are approximately 10^{11} neurons inside brain which are highly connected to each other. When you see an animal for example cat, its features such as size, color and shape are entered to your brain from your eyes gate. Then, these input information will be calculated by small cells so called neurons that are responsible for processing input data in your brain.

Firstly neurons search about your other images from cats which you have seen before. They compare between previous cat pictures in your memory and new cat. This comparison which is fundamental of supervised learning process has made your brain as a comparison tool. This is why human beings tend to compare everything. Finally response is that you saw a cat because you had seen a cat in the past.

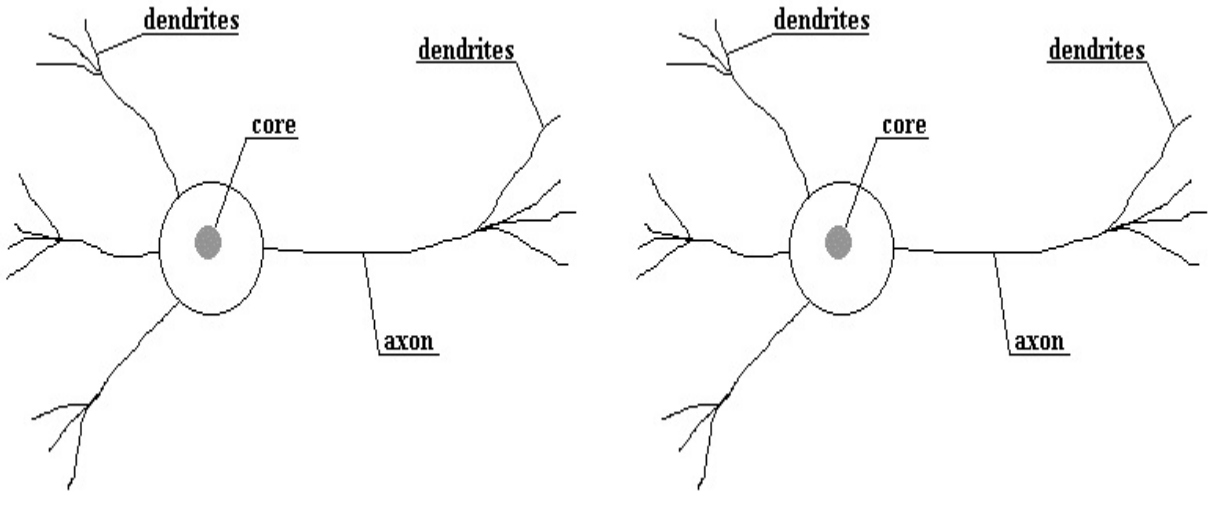
You can process 20 Mb per second just with the aid of your optical sense and it is awesome because it is regardless of your abilities to learn something or to detect the voice of someone or your auditory capability. Therefore your brain is an ubelievable huge CPU.

Assume that having much smaller scale of this huge system how much can solve nowadays problems in different areas and domains. For instance, very small portion of brain`s features can solve issues in speech and face or image recognition, sentiment analysis and opinion or emotion reading, driving a car automatically or even disease diagnosis.

There is a neuron cell in the below picture, it includes dendrites, axon and core. Please look at it from left to right. Dendrites are responsible to receive information and its core is where data is processed and the result will be transmitted from axon to the tail of the neuron. The total structure of one of this neuron in artificial intelligence is called perceptron.

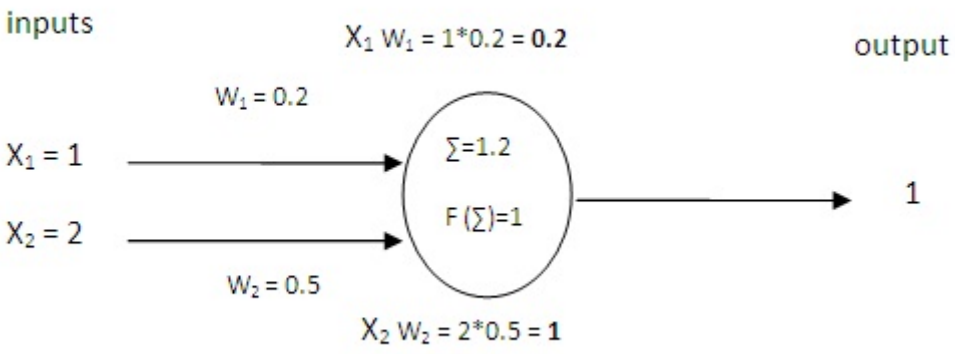
Another neuron cell on the right side will recieves the response of the left side neuron with the aid of some chemical substances which cause fire. Its ignition causes to send and transfer data to other cells. So output from left side neuron is such as input for right side neuron. This process will happen for the rest of other cells. Therefore 1010 neurons in brain collaborate to reach their own goal.

Assume how many processes per day are there to analyze in order to handling your life. Brain performs all of them without much exhausting. At the end of the day you just need 7 hours to cool these little cells. Neuroscientists found out that more learning can make your dendrites mush stronger, because igniting the connection between your neuron cells is a practice which is almost like exercising for your muscles. Therefore using brain is caused less likely to get Alzheimer disease.

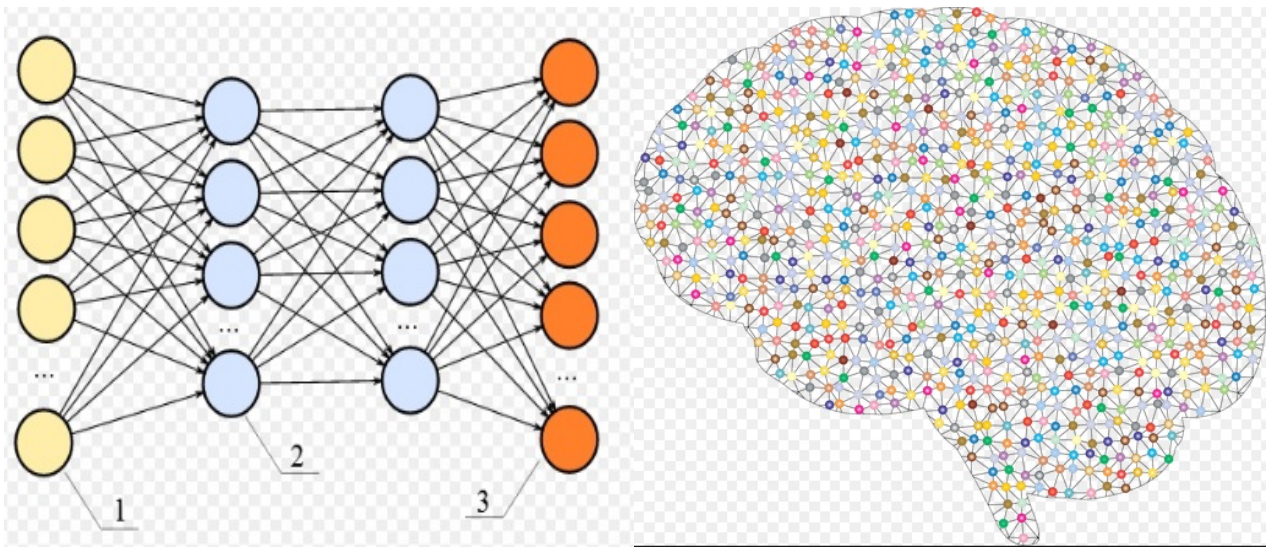


2. How perceptron as an artificial neuron works - Forward neural Network?

In the below picture you can see one simple perceptron and its operation in order to compute its output. Firstly there are two inputs as X1 X2, and then there are weights for each connection to node. In neural network all of processing unit is the node and in spite of computer systems which have complex processing unit, in NN there is simple unit for processing. we multiply two numbers (X and weight).
Then we sum up all of X*weight and apply activation function on the result value and final output is perceptron answer.



The whole story in above paragraph is called as forward propagation in neural network. But indeed we use more nodes and multiple layers for NN learning. I mentioned that in our brain there are billions layers and that huge system made us as a human being today. So, for having better learning we use more layers. It can almost guarantee result improvement in learning or training at NN. There are two hidden layers as light blue in the below picture. The computation for each node will happen like a simple perceptron.



3. What is weight in Neural Network?

Weight refers to the strength of connection between nodes. Unsigned value (without +, -) of weight depends on how nodes have power to connect to each other.
It can be positive or negative. Positive means it is more likely to transmit data and having strong connection among neurons while negative is vice versa. At the initialize point we select weight randomly but for having reasonable result it is better to normalize input data as follow, X is input data:

$$\text{Standardize}=\frac{X-\mu}{\text{Standard Deviation}(X)}$$

$$\text{Standard Deviation}=\sqrt{\frac{\sum(X-\mu)^2}{(n-1)}}$$

Because our activation function in this article is sigmoid. There is a shortcut solution to select weight as randomly is to determine our weight values in a specific range such as below formula which depends on uniform distribution which I keep its logic unspoken here and will extend it in the next article.

$$(-w, w) \qquad w = 4\sqrt{\frac{6}{number_{input} + number_{output}}} \qquad n \in number\ of\ input$$

Reference for above formula is: Deep Learning Tutorials; Glorot and Bengio (2010)

I mentioned that neural network is highly interconnected and weight is the most valuable element which makes this connectivity. We select weight values randomly at the first phase. Firstly forward propagation is done from left to right.

Then make a comparison that how much output value is far from our real value. Real value is label which is "Y" in the training data set. Then backward propagation computation will be done, which performs computation in the inverse path.

Forward propagation is from left to right but back propagation is from right to left to optimize and gain new weight to enhance the next output. If the next output value has less difference from "Y" rather than previous output value, so, it shows us that we are in right direction.
So, weight is a tool to connect nodes to each other and a factor to train NN for having the less error. BP and FP and weights calibration will be measured repeatedly in order to obtain new values for weight and accurate output which decreases error.

For having the better understanding about weight role in NN, I invite you to read my article about "Machine Learning and Gradient Descent". Weight in NN is almost something near to slope "a" in $Y_{prediction}$ line, Y = aX + b. Accurate value for "a" could help us to find better prediction line in order to classify our data. here in NN weight is also a factor such as "a" and we struggle to find its value accurately to have more precision classification.

4. What is equivalent for weight in biological neuron?

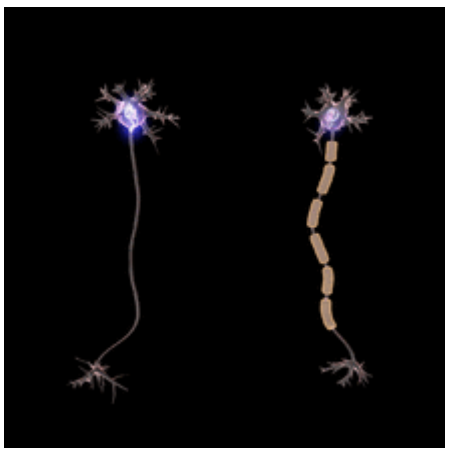
Everything in neural network structures is inspired from human brain. Therefore, unsigned values for weight means: dendrite connection between neurons + count of synapses between dendrites + pre and post synaptic terminals + gap shape among neurons + fusion intensity; last but not least is myelination.

Myelination is white and fatty substance around axon of neuron cells and it is such as sheath or protector for them. On the below picture, propagation on the right neuron with myelination is much faster than left neuron without sheath. This phenomenon occurs because of saltatory conduction.

The number of propagation and its speed can make more and stronger synapses. These factors play important role to have better learning in our brain. Activity of Functional Magnetic Resonance Imaging - FMRI - in someone who has better functionality in their own brain shows that there are more synapses with more red points.

So, weight in neural network is as same as combination of above factors as biologically.

https://en.wikipedia.org/wiki/Myelin

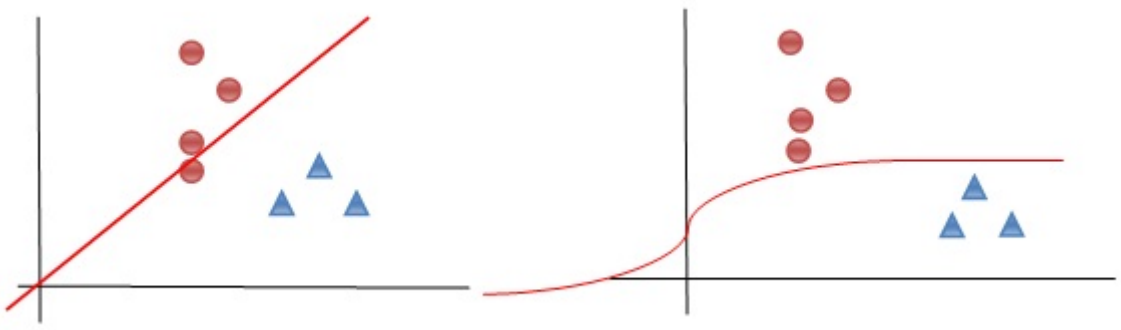


5. What is Activation Function role in Neural Network?

Activation function is (although a bit) equivalent to polarization and stabilizing. I want to bring an example as an introduction for polarization and making stabilizing in mathematic. For having easy computation we need some polarizing on values especially decimal ones. For example we have 1.298456 and we just need one number as decimal.

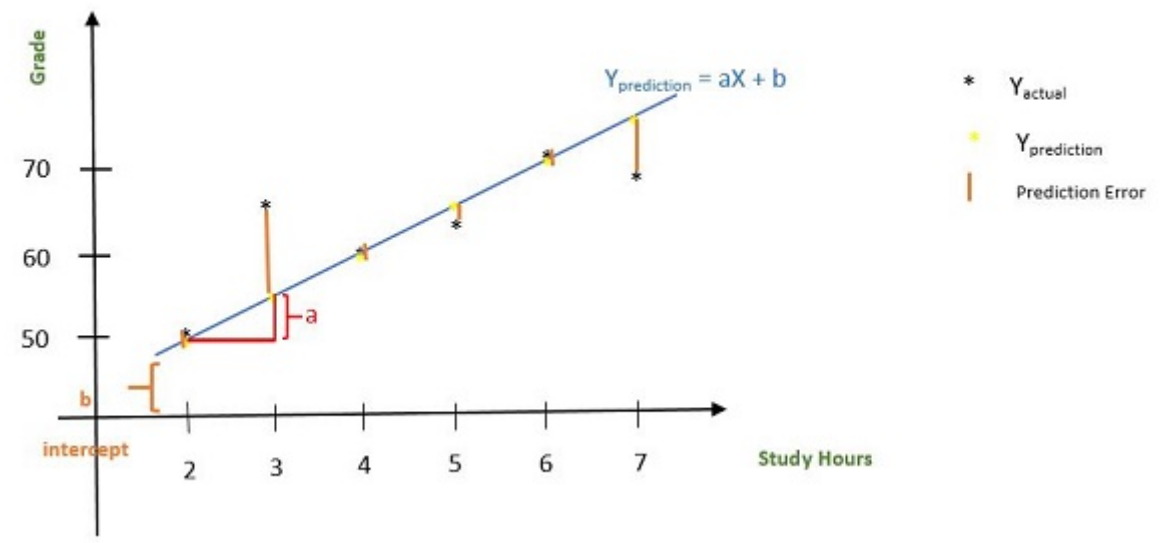
in order to round and polarizing and having easy and fast computation. We convert 1.298456 to 1.3 because the next number after 2 in decimal is 9 and it is more than 5 so we convert 2 into 3. In these cases rounding can help to have more elegant values and results.

In neural network we want better distinguish and prediction. So, non linear functions have more rounding and bending. Please look at below picture. In comparison between linear and non linear functions, it is obvious that non linear has more accurate to predict and have better boundary decision line for categorizing between two different classes.



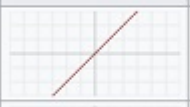
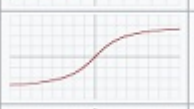
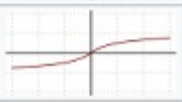
I used "Some Squared Error SSE" in the gradient descent. Activation function for neural network should be non linear function such as exponential or tangent and also it must be differentiable, because in the backward propagation, we need to find global minimum point. Indeed backward propagation performs gradient descent. Please read article about gradient descent with this link.

Sum of Squared Errors (SSE) = ½ Sum $(Y_{actual} - Y_{predicted})^2$



SSE measures error value between Yactual and Ypredicted. Therefor for having best prediction line instead of blue line in above picture we differentiate SSE and compute new slope for this line.

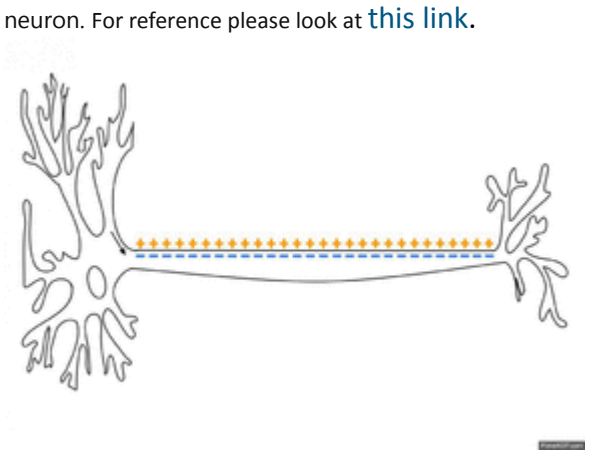
You can select one of below functions as your activation function. For reference please look at this link.

Name	Plot	Equation	Derivative (with respect to *x*)
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Arc Tan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Softsign [7][8]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$

6. What is equivalent for Activation Function in biological neuron?

Activation function in neural network is called as transfer function. Transfer function in neural network makes output for nodes according to their own inputs. Activation function is called as action potential in biological which is related to how signals travel in axon.

The chemical substances make electrical ignition and cause to stimulate neurons and then its axon in order to transmit signal in just one direction of neuron. It helps to produce result for current neuron. For reference please look at this link.



7. How backward propagation works?

In order to sum up all of above concepts, I want to divide backward propagation in various steps as follow:

* There is training data set at the initial step which has one or more than one columns for X as input and one label as Y which should be read and considered. So, numbers of input and output layers are defined.

*We need to select number of hidden layers; count of hidden layers shows depth of learning. More hidden layers can imitate human brain better and enhance its accuracy. But the most important issue is that more hidden layers need more calculation especially in backward propagation and it consumes more memory.

*After hidden layers definition, we need weigh values which are selected randomly with Gaussian and there is formula that I have explained at third section of this article 3. What is weight in Neural Network?

Forward Propagation to reach output value:

Each layer has node and I assume two values for them, first is without applying sigmoid function so called “Input_sigma” or “hidden_ sigma” and the next is with sigmoid function as “hidden_node”, “output_node”. Then we start computation from left to right for forward propagation.

(1) Input_sigma = input_node * weight_1

(2) hidden_node = Sigmoid (input_sigma)

(3) hidden_sigma = hidden_node * weight_2

(4) Sigmoid (hidden_sigma) = output_node

(5) margin_error = expected – output_node

Backward Propagation to reach better weight:

In backward propagation because we need optimum value so we differentiate from sigmoid function and go inversely from right to left, in order to finding new values for weights.

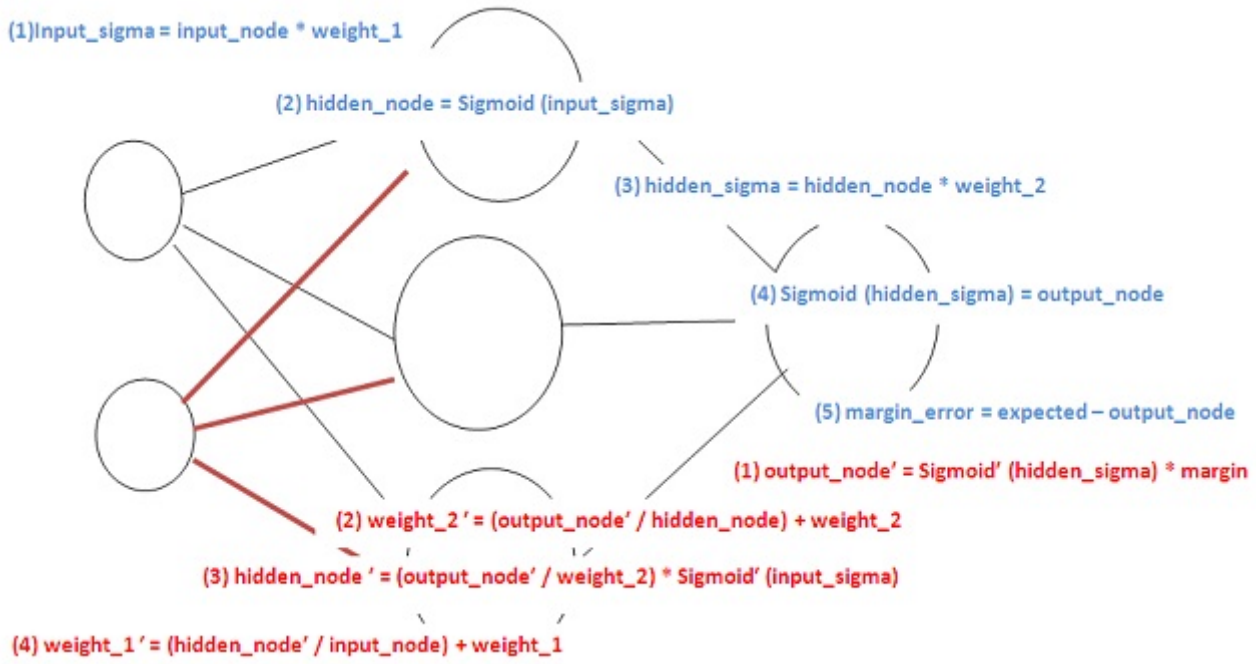
(6) output_node' = Sigmoid' (hidden_sigma) * margin

(7) weight_2 ' = (output_node' / hidden_node) + weight_2

(8) hidden_node ' = (output_node' / weight_2) * Sigmoid' (input_sigma)

(9) weight_1 ' = (hidden_node' / input_node) + weight_1

(10) Again we repeat steps 1 to 5 with new weights and comparison value from current margin errors and previous margin errors if current error is less than previous one, so it shows us that we are in right direction.
(11) We iterate step 1 to 10 until margin error is near to our "Y".



8. What is the exact mathematical logic for backward propagation neural network?

Backward propagation performs as same as gradient descent and we need to have differentiation of activation function. I described its computation as follow:

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \rightarrow Sigmoid(x)' = \frac{-\left((-1) * e^{-x}\right)}{(1+e^{-x})^2}$$

$$= \frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})} * \frac{1}{(1+e^{-x})} = \left(1 - f(x)\right) * f(x)$$

$$\frac{e^{-x}}{(1+e^{-x})} = \frac{1+e^{-x}-1}{1+e^{-x}} = 1 - \frac{1}{1+e^{-x}} = 1 - f(x)$$

$$\frac{1}{1+e^{-x}} = Sigmoid(x) = f(x)$$

$$Sigmoid(x)' = f(x)\left(1 - f(x)\right)$$

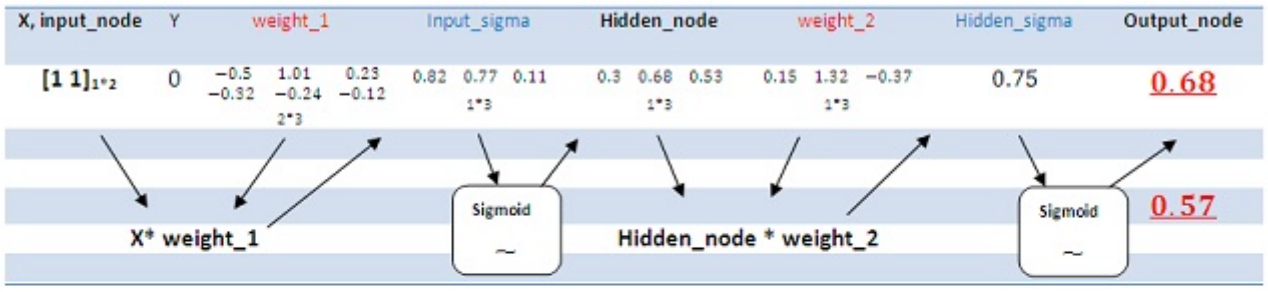
XOR is one simplest sample to test our first neural network. There is XOR table with two inputs and one output:

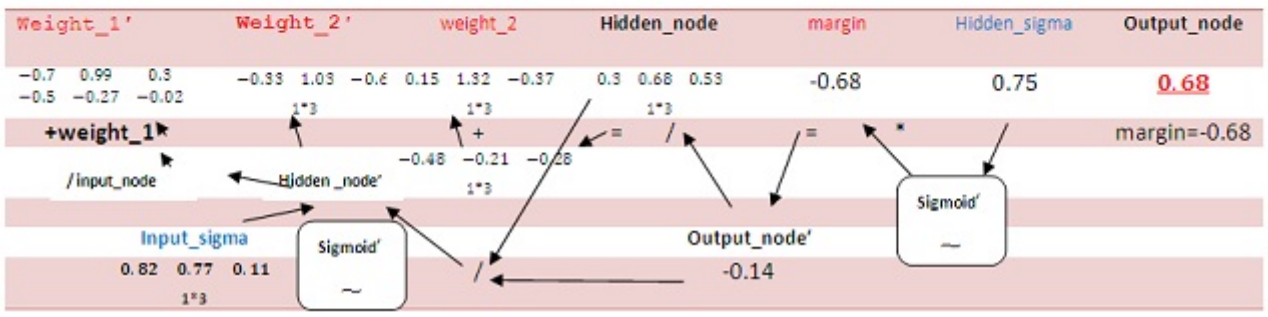
XOR		
inputs		**outputs**
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

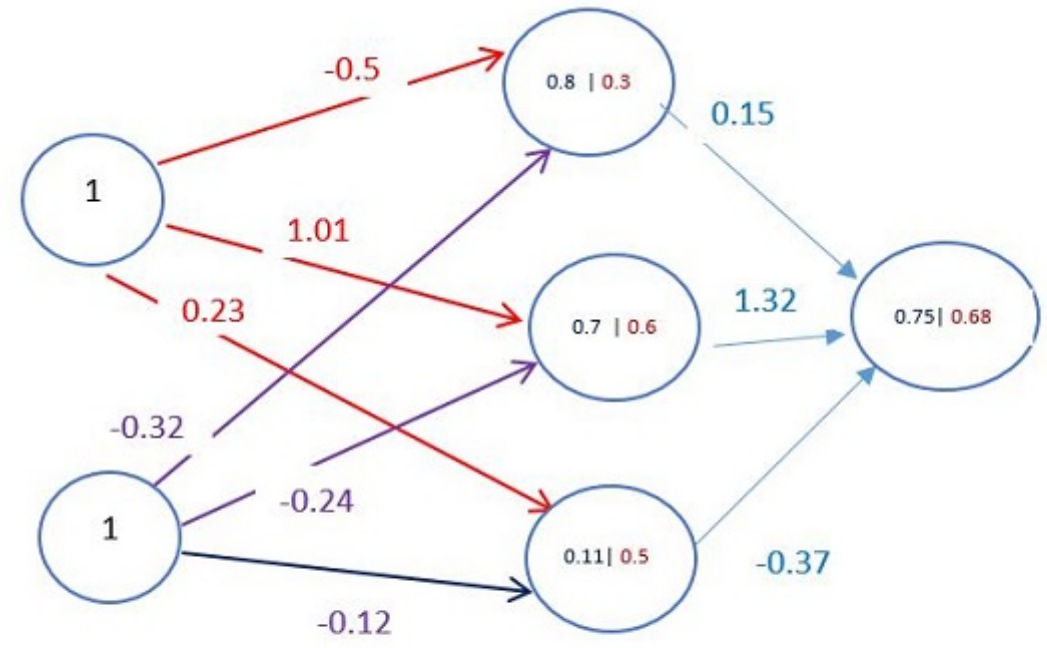
I want to implement forth row in XOR which is (1, 1) = 0.

Y is 0 and output is 0.68. So margin error is -0.68. The next output is 0.57 which is less than 0.68.

Forward Propagation







9. How to implement backward propagation neural network?

Execution File:

According to all of above explanation, I want to implement it on Matlab. Firstly, I created "execution.m" file for calling prediction function.

Hide Copy Code

```
%% Machine Learning -  Neural Networks  - Simple Example

%% Initialization
clear ; close all; clc

input_node = [1 1]; %1*2

% Generate Weight By Gausian Distribution
Weight_1 = [ -0.5 1.01  0.23 ; -0.32 -0.24 -0.12 ]; %2*3
Weight_2 = [ 0.15 1.32 -0.37 ]; %1*3
pred = mypredict(Weight_1, Weight_2, input_node);
fprintf('\Final Output Backward Propagation: %f\n', perd);
```

Prediction File:

Then I wrote "myprerdict.m" which the most code will happen there:

Hide Shrink Copy Code

```
function p = mypredict(Weight_1, Weight_2, input_node)

%Forward Propagation
input_sigma = input_node*Weight_1;
hidden_node = sigmoid(input_sigma); % 1*3
hidden_sigma = hidden_node*Weight_2';
output_node = sigmoid(hidden_sigma);

for jj=1:1000
    %sigmoid' = f(x)(1-f(x))
    %output_node_prime = s'(inner_sigma)*margin
    if jj>1
        Weight_2 = Weight_2_prime;
        Weight_1 = Weight_1_prime;
    end
        margin = 0 - output_node;
        sigmoid_prime_hidden_sigma = sigmoid(hidden_sigma);
        output_node_prime = (sigmoid_prime_hidden_sigma *(1-sigmoid_prime_hidden_sigma))*margin;

        delta_weight = (output_node_prime)./hidden_node; % 1*3
```

```
        Weight_2_prime = Weight_2 + delta_weight;

        sigmoid__prime_input_sigma = sigmoid_derivative(input_sigma);
        mydivide = output_node_prime./Weight_2;
        hidden_node_prime = zeros(1,3);
        hidden_node_prime(1,1) = mydivide(1,1) * sigmoid__prime_input_sigma(1,1);
        hidden_node_prime(1,2) = mydivide(1,2) * sigmoid__prime_input_sigma(1,2);
        hidden_node_prime(1,3) = mydivide(1,3) * sigmoid__prime_input_sigma(1,3);

        delta_weight_2 = hidden_node_prime'*input_node;

        Weight_1_prime = Weight_1 + delta_weight_2';

        input_sigma = input_node*Weight_1_prime;
        hidden_node = sigmoid(input_sigma); % 1*3
        hidden_sigma = hidden_node*Weight_2_prime';
        output_node = sigmoid(hidden_sigma);
end

p = output_node;

end
```

Sigmoid Function:

Hide Copy Code

```
function y = sigmoid(x)

y = 1.0 ./ (1.0 + exp(-x));

end
```

sigmoid_derivative:

Hide Copy Code

```
function y = sigmoid_derivative(x)

 %sigmoid' = f(x)*(1-f(x))

 sigmoid_helper_2 = zeros(1,3);
    for i=1:3
        a= x(1,i);
        sigmoid_helper_2(1,i)= sigmoid(a)*(1-sigmoid(a));
    end
 y = sigmoid_helper_2;

end
```

Points of Interest

I found Neural Network very exciting, I think we can call it as mother of atificial inttelligence.

The most advantages and disadvantages of NN is that:

1. Normalizing data set and having best feature selectionscan lead us to have better accurate output in huge training data set.
2. Performance and accuracy depends on weight but if you select it in correct range. then you can nenhhance performance and accuracy.
3. Backward propagation consumes more memory than other ways.

Finally I strongly recommend you to register machine learning in coursera.org: https://www.coursera.org/learn/machine-learning

& use my github as guidence for assignments:

https://github.com/Hassankashi?tab=repositories

Feel free to leave any feedback on this article; it is a pleasure to see your opinions and **vote** about this code. If you have any questions, please do not hesitate to ask me here.