COMP9444: Neural Networks and Deep Learning

Week 2: Tutorial

Sonit Singh
School of Computer Science and Engineering

May 31, 2024

Question 1

Any questions from the first tutorial.

Recap:

- What is Perceptron and how to construct a Perceptron using our understanding of geometry?
- What is Perceptron Learning Algorithm?
- How to construct any logical function with a 2-layer network?
- What are the limitations of Perceptron and its learning algorithm?
- How to construct a Multi-Layer Perceptron (MLP) to compute XOR function?
- What are the implications of deep learning?

Note: The solution of Week 1 tutorial (Sample solution along with video) are released. Please watch the video to know solution to all questions of Tutorial 1 in details.

Question 2

Simple Gradient Descent by Hand

Consider the simplest possible machine learning task:
Solve $f(x) = wx$ such that $f(1) = 1$, *i.e.*, $f(x) = t$, for $x = 1, t = 1$.
We can solve this by gradient descent using the loss function $E = \frac{1}{2}(wx - t)^2$, with a learning rate $\eta = 0.5$ and initial value $w = 0$.

1. Perform the first epoch of training by completing this table:

   $w = 0$

   $x = 1$

   $f(x) = ?$

   $E = ?$

   $\frac{\partial E}{\partial w} = ?$

   $w \leftarrow w - ? = ?$
2. Repeat these calculations for the second epoch.
3. Use the code provided in the Exercise in Lesson 2b on Ed to explore what happens for different values of the learning rate. Discuss your findings.

Question 2

First Epoch

$f(x) = wx$ such that $f(1) = 1$, *i.e.*, $f(x) = t$, for $x = 1, t = 1$.
Loss function $E = \frac{1}{2}(wx - t)^2$, $\eta = 0.5$ and initial value $w = 0$.

1. Performing the first epoch of training:

   $w = 0$

   $x = 1$

   $f(x) = wx = 0 * 1 = 0$

   $E = \frac{1}{2}(wx - 1)^2 = \frac{1}{2}(0 * 1 - 1)^2 = \frac{1}{2} = 0.5$

   $\frac{\partial E}{\partial w} = \frac{1}{2} * 2(wx - t) * x = (0 * 1 - 1) = -1$

   $w \leftarrow w - \eta * \frac{\partial E}{\partial w} = 0 - 0.5 * (-1) = 0.5$

Question 2

Second Epoch

$f(x) = wx$ such that $f(1) = 1$, *i.e.*, $f(x) = t$, for $x = 1, t = 1$.
Loss function $E = \frac{1}{2}(wx - t)^2$, $\eta = 0.5$ and **new value of w**, $w = 0.5$.

1. Performing the second epoch of training:

   $w = 0.5$

   $x = 1$

   $f(x) = wx = 0.5 * 1 = 0.5$

   $E = \frac{1}{2}(wx - 1)^2 = \frac{1}{2}(0.5 * 1 - 1)^2 = \frac{1}{2}(-0.5)^2 = 0.125$

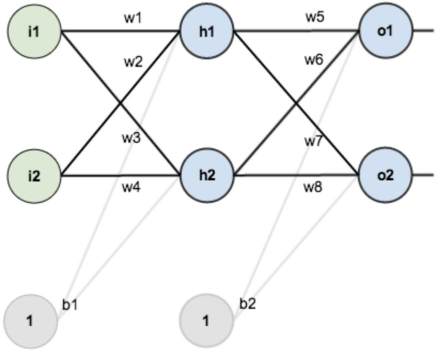
   $\frac{\partial E}{\partial w} = \frac{1}{2} * 2(wx - t) * x = (0.5 * 1 - 1) * 1 = -0.5$

   $w \leftarrow w - \eta * \frac{\partial E}{\partial w} = 0.5 - 0.5 * (-0.5) = 0.5 + 0.25 = 0.75$

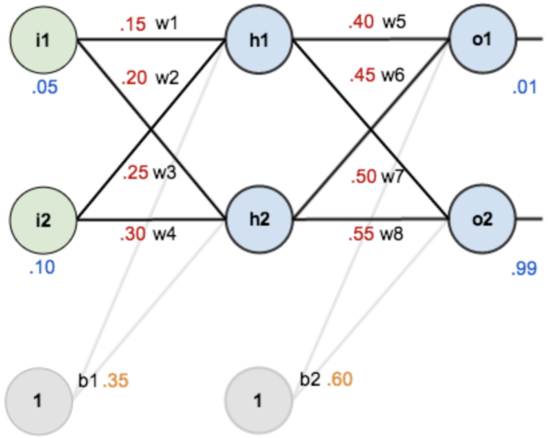
Question 3

Understanding Backpropagation through an Example

For the given 2-layer network with initial weights and data, try to calculate by hand the forward and the backward pass to understand how backpropagation works.



Understanding Backpropagation through an Example



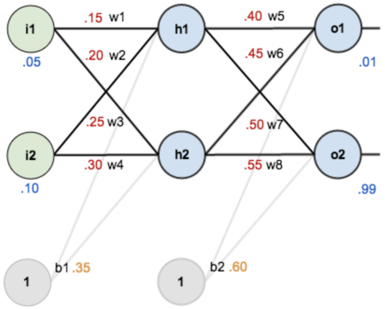
Initial weights and training data

Given:
$w_1 = 0.15$, $w_2 = 0.20$, $w_3 = 0.25$, $w_4 = 0.30$, $w_5 = 0.40$, $w_6 = 0.45$, $w_7 = 0.50$, $w_8 = 0.55$, $b_1 = 0.35$, $b_2 = 0.60$

Data $\rightarrow$ *Input* :i$_1 = 0.05$, $i_2 = 0.10$ and target: $out_{o1} = 0.01$, $out_{o2} = 0.99$

We know:
Sigmoid activation function: $g(x) = \frac{1}{1+e^{-x}}$

Forward pass



Let's calculate for the hidden layer.

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1$$
$$net_{h1} = 0.15 * 0.05 + 0.20 * 0.10 + 0.35$$
$$net_{h1} = 0.3775$$

Apply sigmoid activation fn.

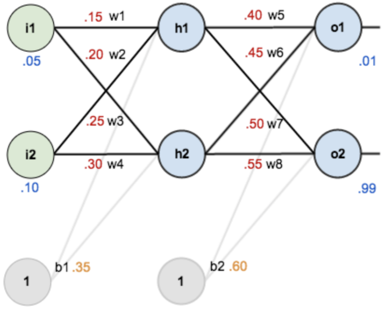
$$out_{h1} = \frac{1}{1+e^{-0.3775}} = 0.5932$$

$$net_{h2} = w_3 * i_1 + w_4 * i_2 + b_1$$
$$net_{h2} = 0.25 * 0.05 + 0.30 * 0.10 + 0.35$$
$$net_{h2} = 0.3925$$

Apply sigmoid activation fn.

$$out_{h2} = \frac{1}{1+e^{-0.3925}} = 0.5968$$

Forward pass



Let's calculate for the output layer.

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2$$
$$net_{o1} = 0.40 * 0.5932 + 0.45 * 0.5968 + 0.60$$
$$net_{o1} = 1.1058$$

Apply sigmoid activation fn.

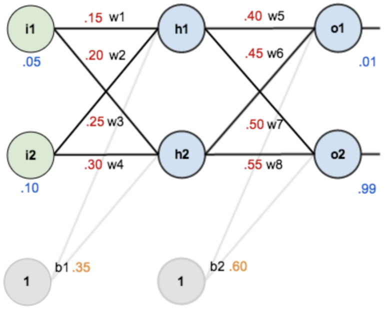
$$out_{o1} = \frac{1}{1+e^{-1.1058}} = 0.7513$$

$$net_{o2} = w_7 * out_{h1} + w_8 * out_{h2} + b_2$$
$$net_{o2} = 0.50 * 0.5932 + 0.55 * 0.5968 + 0.60$$
$$net_{o2} = 1.2248$$

Apply sigmoid activation fn.

$$out_{o2} = \frac{1}{1+e^{-1.2248}} = 0.7729$$

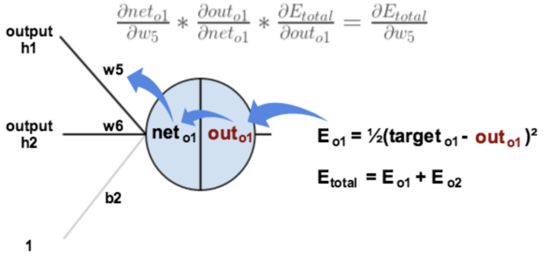
Forward pass



Calculating total error at the output layer:

$$E_{total} = \sum \tfrac{1}{2}(target - output)^2$$
$$E_{total} = E_{o1} + E_{o2}$$
$$E_{total} = \tfrac{1}{2}(target_{o1} - out_{o1})^2 + \tfrac{1}{2}(target_{o2} - out_{o2})^2$$
$$E_{total} = \tfrac{1}{2}(0.01 - 0.7513)^2 + \tfrac{1}{2}(0.99 - 0.7729)^2$$
$$E_{total} = 0.2747 + 0.0235$$
$$E_{total} = 0.2983$$

Backward pass



$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
w5
output h2
w6
net $_{o1}$
out $_{o1}$
b2
1
$E_{o1} = ½(target_{o1} - out_{o1})^2$
$E_{total} = E_{o1} + E_{o2}$

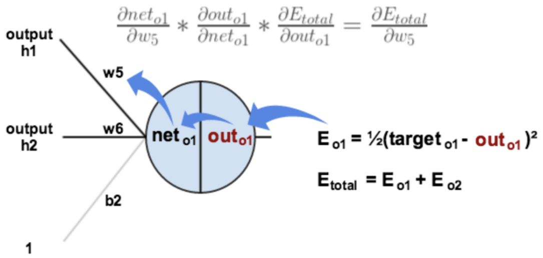
We will be using *Backpropagation* to update each of the weights in the network so that actual output is close to the target output, thereby minimising the error for each output neuron and in turn, minimising total error for the entire network.

Let's start with w_5. Basically, we want to know how much a change in w_5 affects the total error, *i.e.*, $\frac{\partial E_{total}}{\partial w_5}$

We know *Chain Rule*:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

Backward pass



$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
w5
output h2
w6
net o1
out o1
b2
1
$E_{o1} = ½(target_{o1} - out_{o1})^2$
$E_{total} = E_{o1} + E_{o2}$

We need to calculate this:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

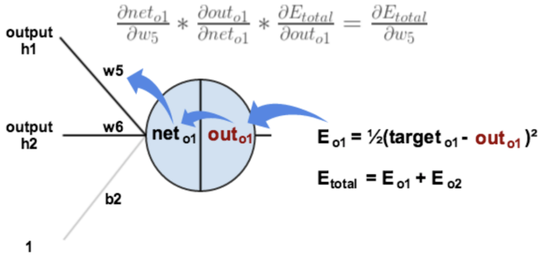
$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1}) * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1})$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(0.01 - 0.7513)$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 0.7413$$

Backward pass



$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
w5
output h2
w6
b2
1
net_{o1}
out_{o1}
$E_{o1} = ½(target_{o1} - out_{o1})^2$
$E_{total} = E_{o1} + E_{o2}$

For second term, we know:

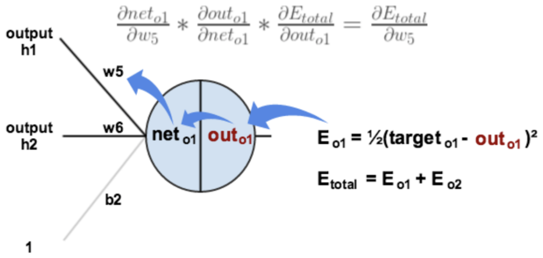
If $z(s) = \frac{1}{1+e^{-s}}$, then $z^{'}(s) = z(1-z)$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1})$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = 0.7513(1 - 0.7513)$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = 0.1868$$

Backward pass

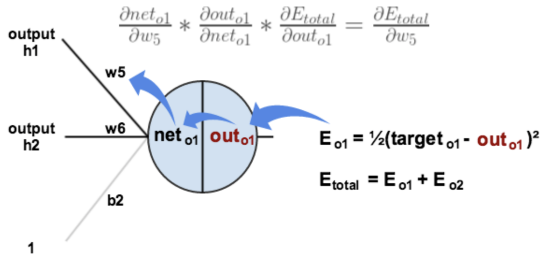


$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
output h2
w5
w6
b2
1
net o1
out o1
E o1 = ½(target o1 - out o1)²
E total = E o1 + E o2

For third term, we know:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2$$
$$\frac{\partial net_{o1}}{\partial w_5} = out_{h1} + 0 + 0$$
$$\frac{\partial net_{o1}}{\partial w_5} = 0.5932$$

Backward pass



$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
w5
output h2
w6
net_{o1}
out_{o1}
b2
1
$E_{o1} = ½(target_{o1} - out_{o1})^2$
$E_{total} = E_{o1} + E_{o2}$

Combining all three terms

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.7413 * 0.1868 * 0.5932$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.0821$$

Backward pass

If you look closely to the three terms:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1})$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1})$$

$$\frac{\partial net_{o1}}{\partial w_5} = out_{h1}$$

Therefore, we got:

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

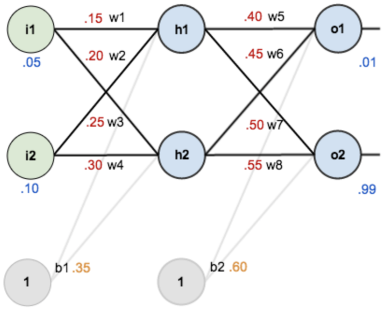
We can define δ_{o1}, *i.e.*, *node delta* as follows:

$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial net_{o1}}$$

Therefore, $\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$

Hence $\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} * out_{h1}$

Backward pass

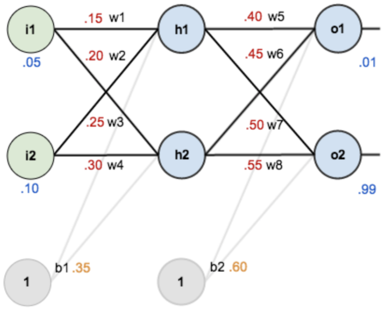


We calculated $\frac{\partial E_{total}}{\partial w_5} = 0.0821$,
So, let's update weights w_5:

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5}$$
$$w_5^+ = 0.40 - 0.5 * 0.0821$$
$$w_5^+ = 0.3589$$

Hence, we updated the weight w_5.

Backward pass



You can repeat the same process to get updated weights w_6, w_7, and w_8.

$w_6^+ = 0.4086$

$w_7^+ = 0.5113$

$w_8^+ = 0.5613$

Backward pass (Hidden Layer)

We will apply backward pass to calculate new values of weights, w_1, w_2, w_3, and w_4. Let's first focus on w_1:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

However, we know that output of each hidden layer neuron contributes to the output of multiple output neurons (hence error as well).

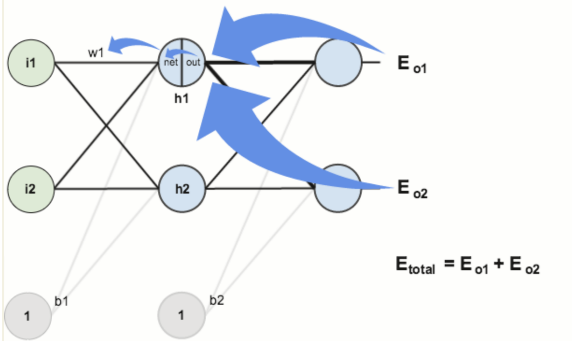
We know out_{h1} affects both out_{o1} and out_{o2}, therefore:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

Backward pass (Hidden Layer)

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



Backward pass (Hidden Layer)

First term:

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.7413 * 0.1868 = 0.1389$$

We know: $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2$

Therefore: $\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$

Putting these values: $\frac{\partial E_{o1}}{\partial out_{h1}} = 0.1389 * 0.40 = 0.0553$

Following the same process for $\frac{\partial E_{o2}}{\partial out_{h1}}$, we get:

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.0190$$

Therefore: $\frac{\partial E_{total}}{\partial out_{h1}} = 0.0553 - 0.0190 = 0.0363$

Backward pass (Hidden Layer)

Second term: $\frac{\partial out_{h1}}{\partial net_{h1}} = ?$

We know: $out_{h1} = \frac{1}{1+e^{-net_{h1}}}$

Hence, $\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.5932(1 - 0.5932) = 0.2413$

Third term: $\frac{\partial net_{h1}}{\partial w_1} = ?$

We know: $net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1$

Hence, $\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$

Putting it all together:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.0363 * 0.2413 * 0.05 = 0.0004$$

Backward pass (Hidden Layer)

We cal also represent in δ terms as follows:

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \frac{\partial E_{total}}{\partial out_o} * \frac{\partial out_o}{\partial net_o} * \frac{\partial net_o}{\partial out_{h1}}\right) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \delta_o * w_{ho}\right) * out_{h1}(1 - out_{h1}) * i_1$$

$$\frac{\partial E_{total}}{\partial w_1} = \delta_{h1} i_1$$

Backward pass (Hidden Layer)

We can now update w_1 based on the calculations:

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.0004 = 0.1497$$

Repeating the same process, we will get:

$$w_2^+ = 0.1995$$

$$w_3^+ = 0.2497$$

$$w_4^+ = 0.2995$$

Backward pass (Hidden Layer)

We now got all weights updated in the network:
Original Weights: $w_1 = 0.15, w_2 = 0.20, w_3 = 0.25, w_4 = 0.30, w_5 = 0.40, w_6 = 0.45, w_7 = 0.50, w_8 = 0.55, b_1 = 0.35, b_2 = 0.60$ and data: (0.05, 0.10),

Total Error (E_{total}): 0.2983 (See slide 11)

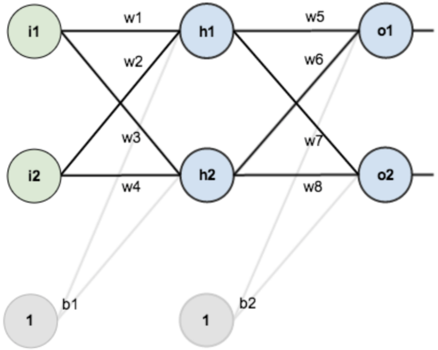
After first round of backpropagation, we got:

Updated Weights: $w_1 = 0.1497, w_2^+ = 0.1995, w_3^+ = 0.2497, w_4^+ = 0.2995, w_5^+ = 0.3589, w_6^+ = 0.4086, w_7^+ = 0.5113, w_8^+ = 0.5613$ (See slide 18, 19, 25)

Total Error (E_{total}): 0.2910

We can see that error reduces from 0.2983 to 0.2910.

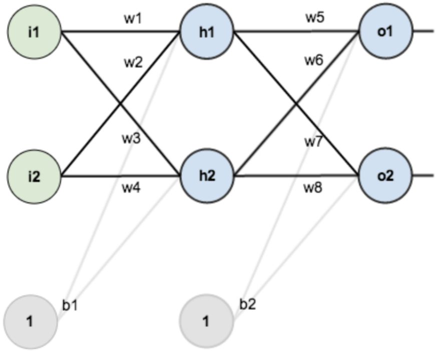
What's next



Let's check network output after first round of backpropagation:
Input: $(i_1 = 0.05, i_2 = 0.10)$ and updated weights:
We get output: $o_1 = 0.7420, o_2 = 0.7752$

You can see here that o_1 has decreased (previous o_1 was 0.7513) whereas o_2 has increased (previous o_1 was 0.7729), showing output is moving towards target (or desired) output.

What's next



Let's check network output after 1000 rounds of backpropagation:
Input: $(i_1 = 0.05, i_2 = 0.10)$ and updated weights:
We get output: $o_1 = 0.0159, o_2 = 0.9840$

You can see here output is close to target (or desired) output.

Acknowledgements

- Matt Mazur’s Blog on “A Step by Step Backpropagation Example”
- Backpropagation Step by Step
- Datamapu’s blog on Backpropagation Step by Step
- Back-Propagation is very simple. Who made it Complicated?
- Machine Learning Mastery Blog on “How to Code a Neural Network with Backpropagation In Python (from scratch)”
- Alexander Schiendorfer’s Blog on “A worked example of backpropagation”
- Backprop Explainer: Interactive Explanation of Backpropagation in Neural Network Training