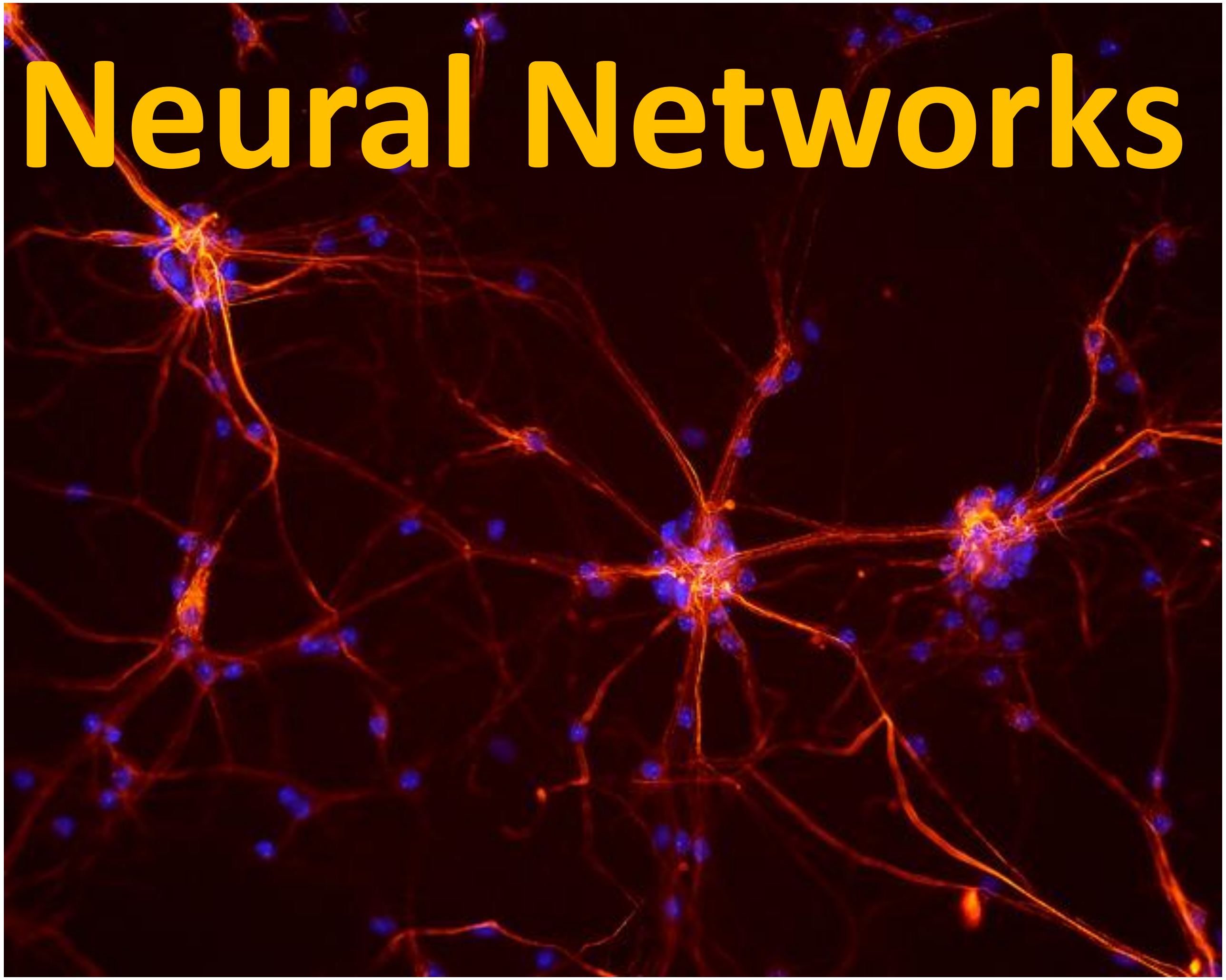
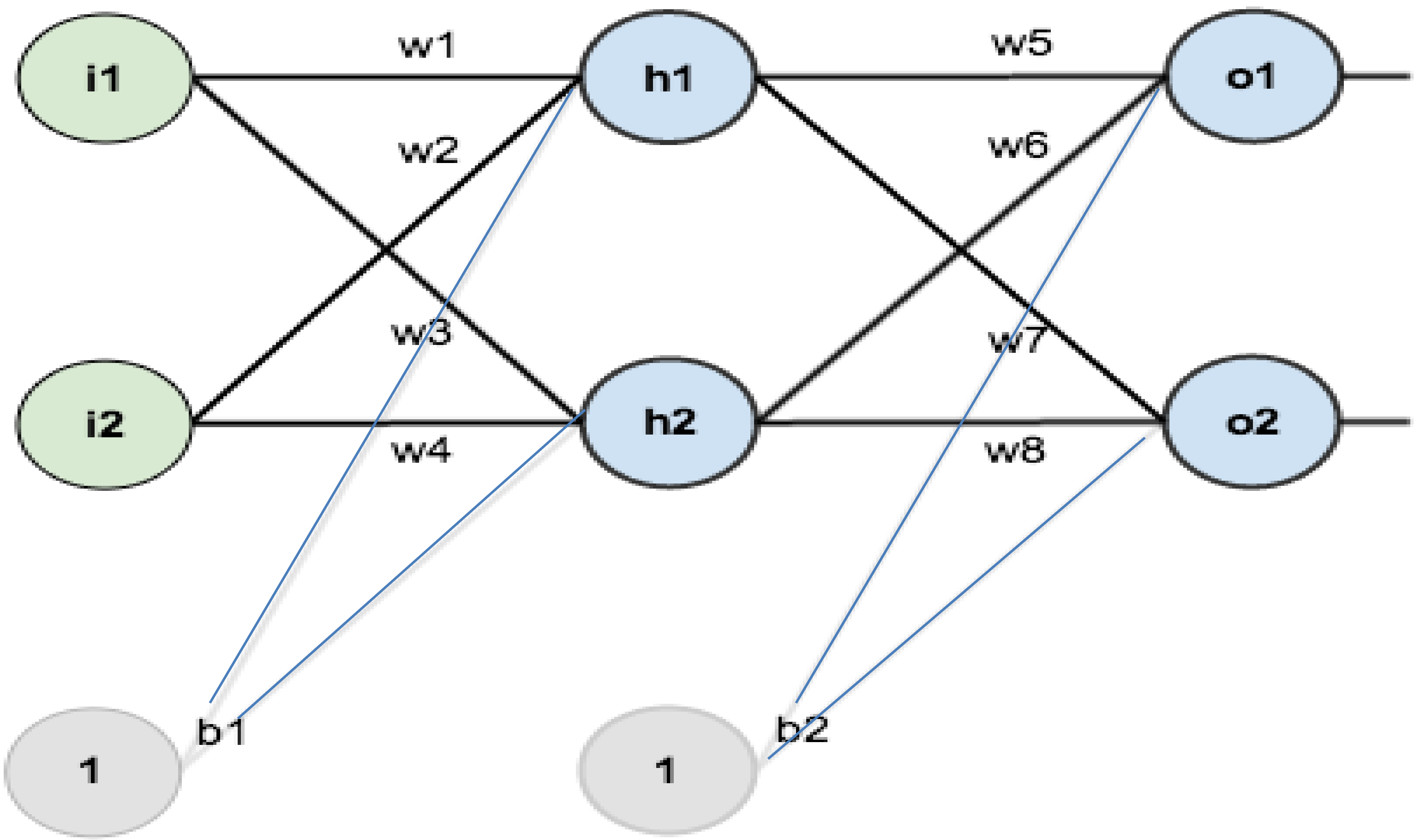
Back Propagation in Neural Networks

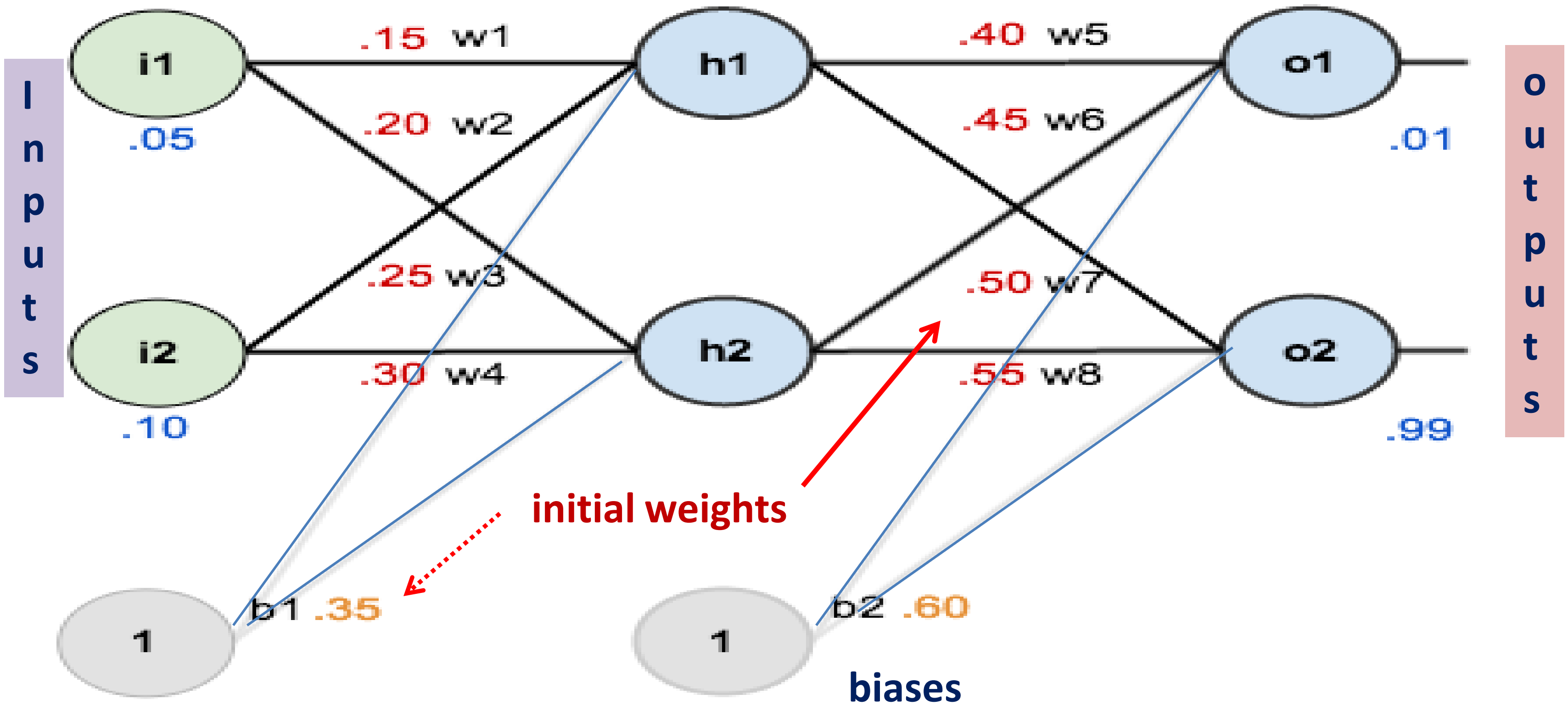


Ajeet K Jain
KMIT, Hyderabad

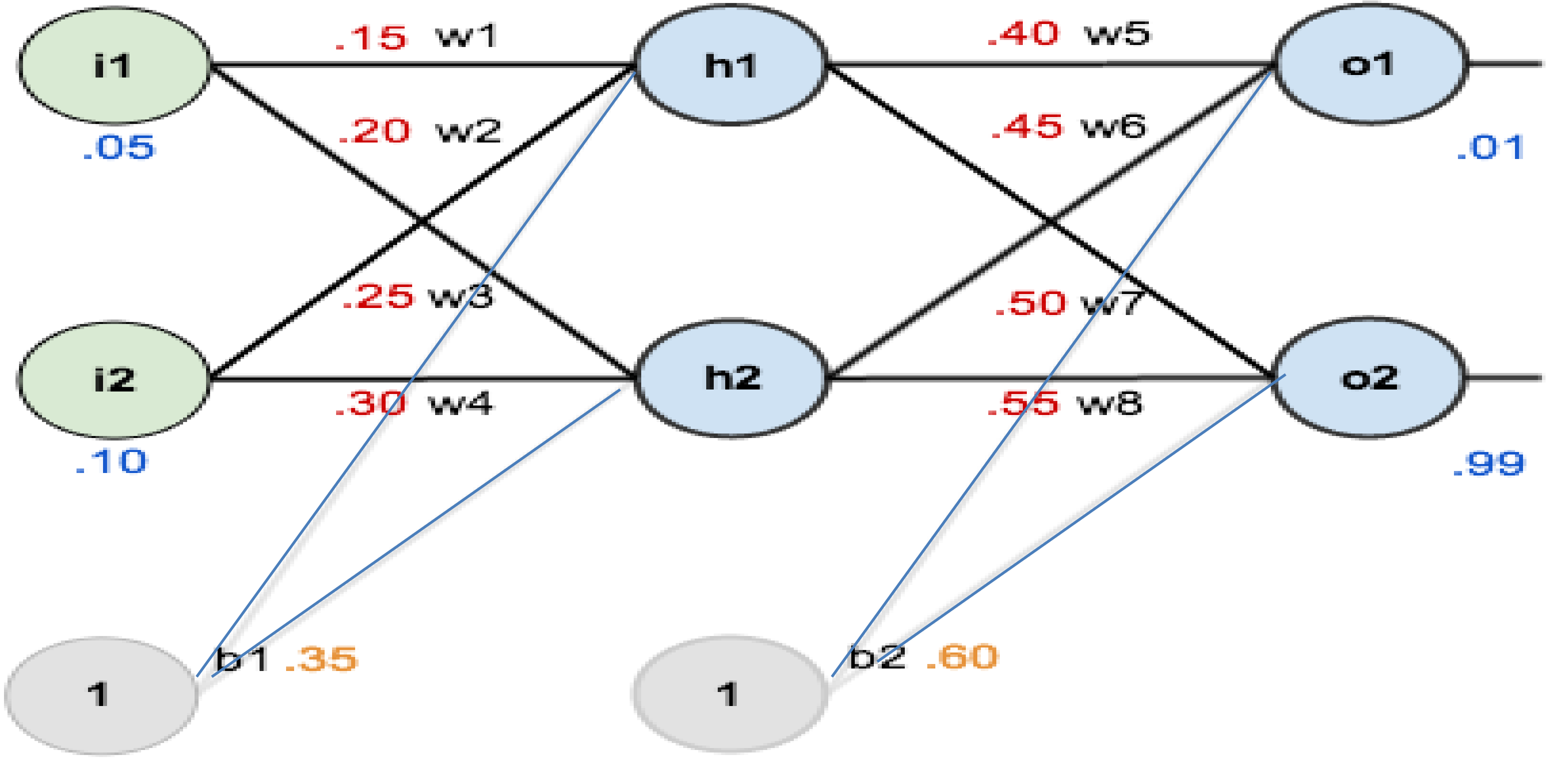
The basic structure of a NN :



In order to have some numbers to work with, here are:



The goal of back propagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.



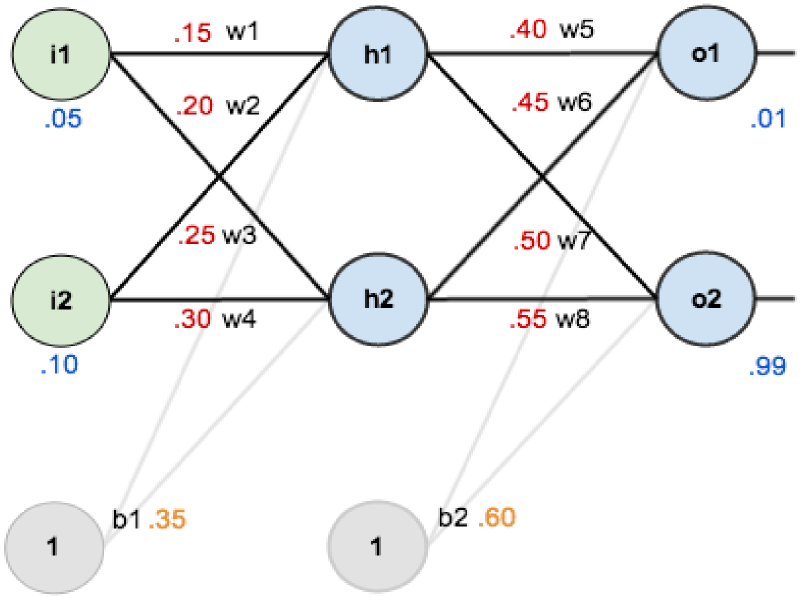
To work with a single training set:

given inputs 0.05 and 0.10, we want the neural network to

output 0.01 and 0.99.

The Forward Pass

We figure out the *total net input* to each hidden layer neuron, *squash* the total net input using an *activation function* (*logistic function*), then repeat the process with the output layer neurons.



Here's how we calculate the total net input for :

$$net_{h1} = w1 * i_1 + w2 * i_2 + b1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1$$

$$net_{h1} = 0.3775$$

$$net_{h2} = w3 * i_1 + w4 * i_2 + b1 * 1$$

$$net_{h2} = 0.25 * 0.05 + 0.3 * 0.1 + 0.35 * 1$$

$$net_{h2} = 0.3925$$

Squash it using the logistic function to get the output of ***h1*** :

$$out_{h1} = 1 / (1 + e^{-x}) = 1 / (1 + e^{-0.3775})$$

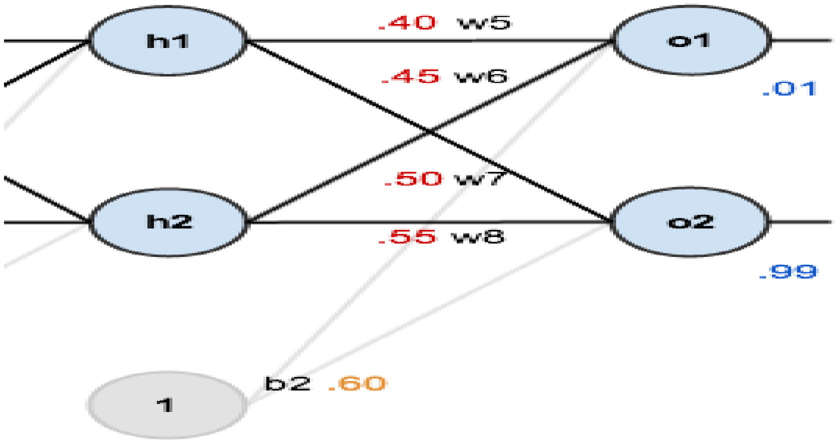
$$= 0.59326992$$

Carrying out the same process for, we get ***h2*** :

$$out_{h2} = 1 / (1 + e^{-x}) = 1 / (1 + e^{-0.\ 3925})$$

$$= 0.596884378$$

Repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.



Here's the output for O1 :

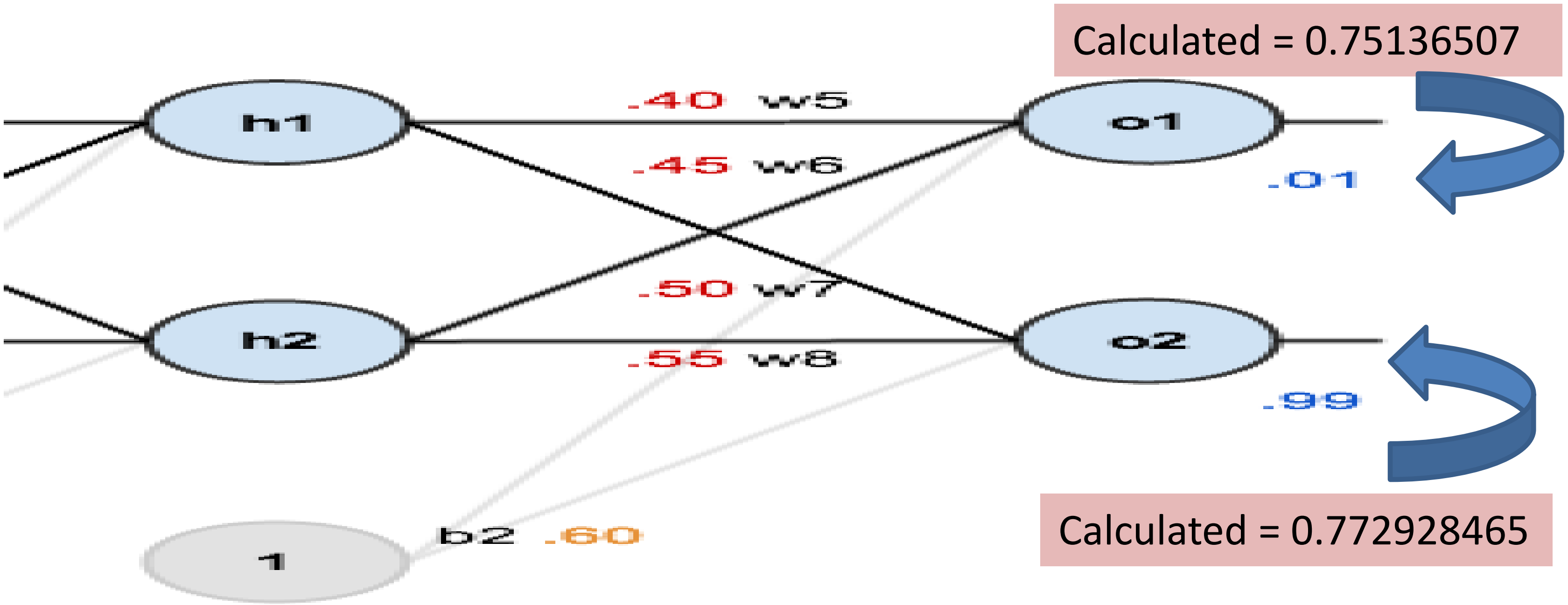
$$net_{o1} = w5 * out_{h1} + w6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1$$

$$net_{h2} = 1.105905967$$

$$out_{o1} = 1 / (1 + e^{-neto1}) = 1 / (1 + e^{-1.105905967})$$

$$= 0.75136507$$



And carrying out the same process for O_2 we get:

out_{o2} = 0.772928465

Calculating the Total Error

We can now calculate the error for each output neuron using the **squared error function** and sum them to get the total error:

$$E_{total} = \sum 1/2 \, (target - output)^2$$

For example, the target output for $\boldsymbol{O_1}$ is **0.01** but the neural network output **0.75136507**, therefore its error is:

$$E_{o1} = \sum 1/2 \, (target - output)^2$$

$$E_{o1} = 1/2 \, (0.01 - 0.75136507)^2$$

$$E_{o1} = 0.274811083$$

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for ***O_2*** (remembering that the target is 0.99) we get:

$E_{o2} = 0.023560026$

$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$E_{total} = E_{o1} + E_{o2}$

$E_{total} = 0.274811083 + 0.023560026$

$E_{total} = 0.298371109$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

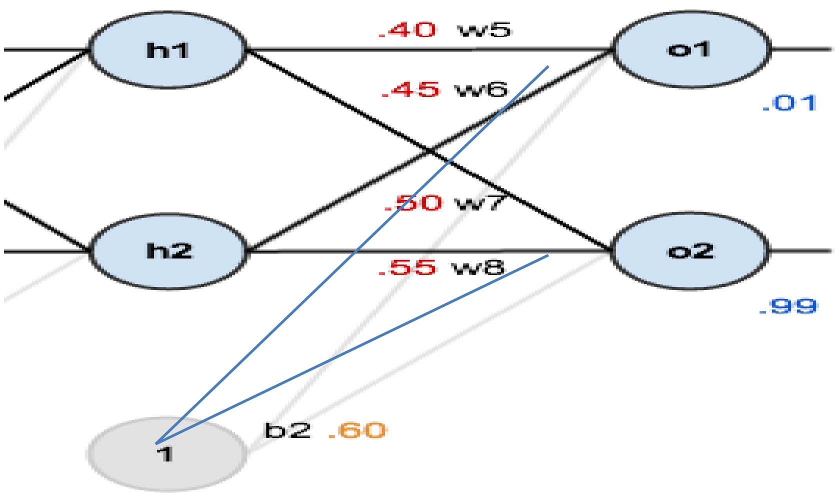
The Backwards Propagation

Our goal with back propagation is to update each of the weights in the network so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole.

The Desired Technique now…………….

Start from output layer and go backward towards hidden layer towards input layer – using a derivative chain rule:

Consider w5. We want to know how much a change in w5 affects the total error, aka $\frac{\partial E_{total}}{\partial w5}$.

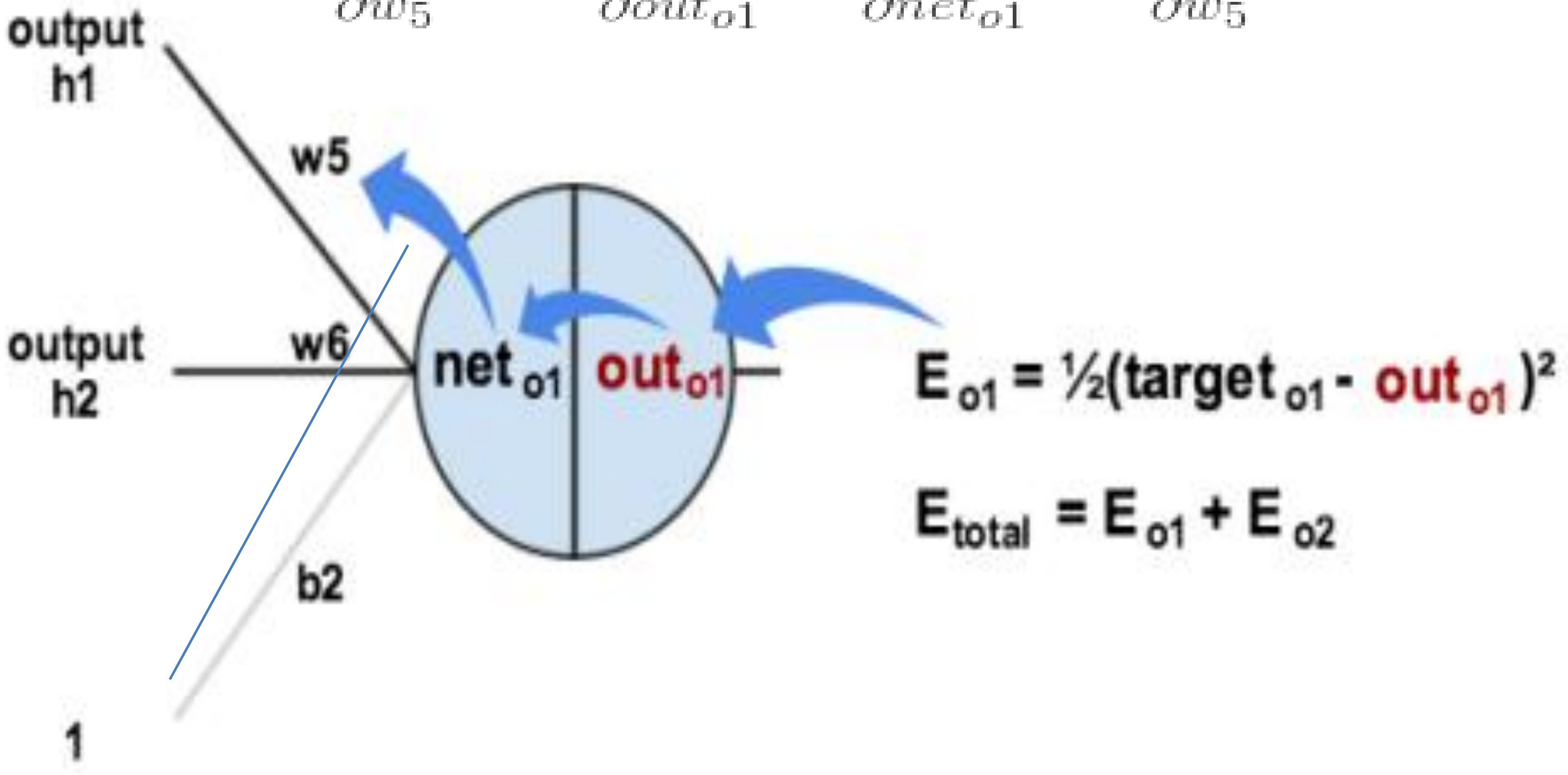


By applying the chain rule we know that:

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

Visually, here's what we're doing:

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



We need to figure out each piece in this equation.
First, how much does the total error change wrt output?

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

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

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

IMP: - (target - out) is sometimes expressed as (out – target).

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

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

When we take the partial derivative of the total error wrt out $_{01}$, the quantity ½(target$_{02}$ – out$_{02}$) ^{2} becomes zero because out$_{01}$ does not affect it – meaning we are taking the derivative of a constant - which is 0 (zero).

Derivative [edit]

The standard logistic function has an easily calculated derivative. The derivative is known as the logistic distribution:

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

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \frac{e^x \cdot (1+e^x) - e^x \cdot e^x}{(1+e^x)^2} = \frac{e^x}{(1+e^x)^2} = \underbrace{f(x)\big(1 - f(x)\big) = f(x)f(-x)}.$$

The derivative of the logistic function is an even function, that is,

$$f'(-x) = f'(x).$$

Next, how much does the output of O_1 change with respect to its total net input?
The partial **derivative of the logistic function** is the output multiplied by 1 minus the output:

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

Out $_{01}$ = 1/(1 + e $^{-net01}$)

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

Finally, how much does the total net input of $o1$ change with respect to w_5?

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

Putting it all together:

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

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, **η** (**eta**), which we 'll set to 0.5):

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

Confusion :

Some sources use

α (alpha) to represent learning rate,

others use η(eta), and

others even use ε (epsilon).

We can repeat this process to get the new weights w_6, w_7, and w_8:

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

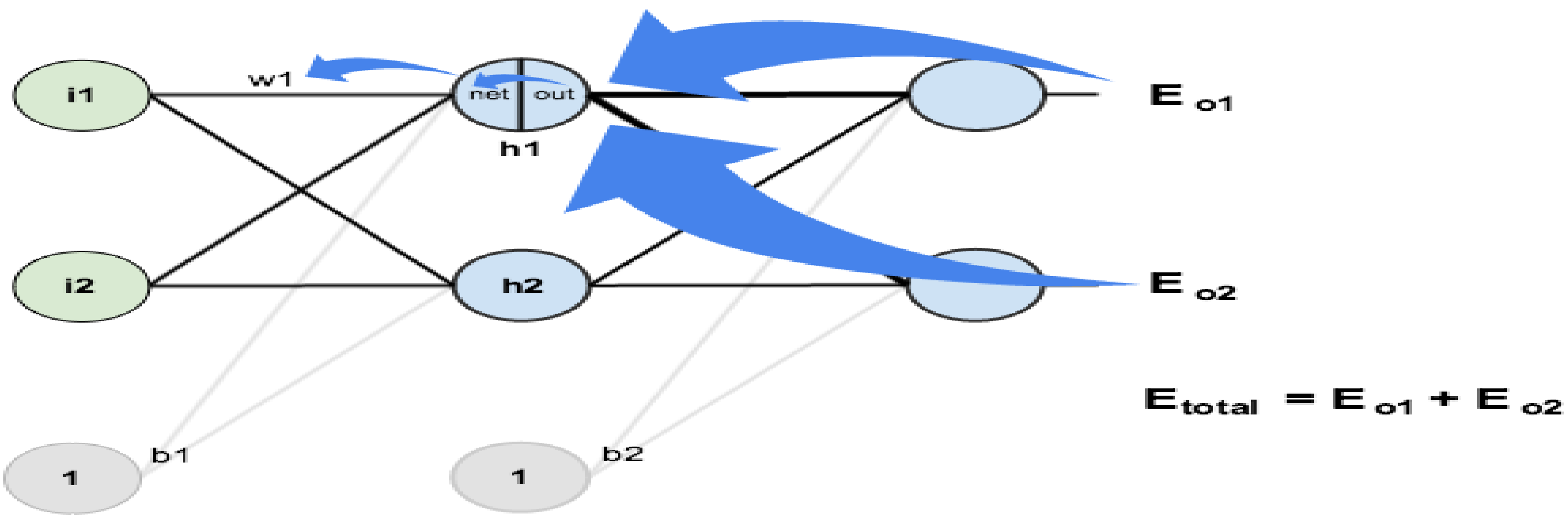
$$w_8^+ = 0.561370121$$

We perform the actual updates in the neural network *after* we have the new weights leading into the hidden layer neurons (i.e., we use the original weights, not the updated weights, when we continue the back propagation algorithm).

Hidden Layer

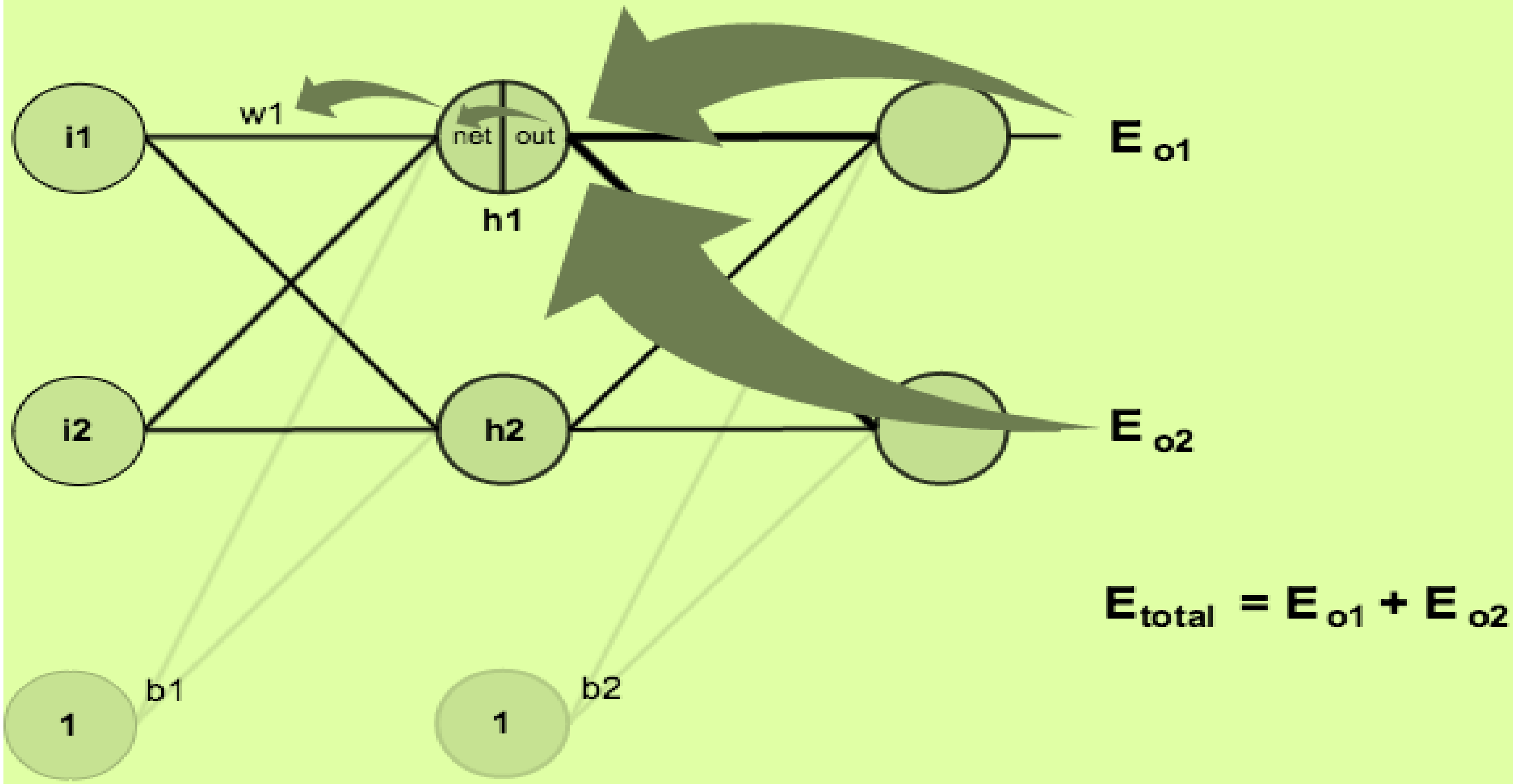
Next , we continue the backwards pass by calculating new values for w1, w2, w3 and w4. (*these are the weights which are right side of I/P and left side of hidden layer*).

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w1}$$



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

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



We're going to use a similar process as we did for the output layer, but slightly different to account for the fact that the output of each hidden layer neuron contributes to the output (and therefore error) of multiple output neurons. We know that out_{h1} affects both out_{o1} and out_{o2} therefore the $\frac{\partial E_{total}}{\partial out_{h1}}$ needs to take into consideration its effect on the both output neurons:

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

Starting with $\frac{\partial E_{o1}}{\partial out_{h1}}$:

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

We are going to use a similar process as we did for the output layer, but slightly different to account for the fact that the output of each hidden layer neuron contributes to the output (and therefore error) of multiple output neurons. We know that $\mathbf{out_{h1}}$ affects both out_{O1} and out_{O2} therefore the $\partial E_{total} / \partial out_{h1}$ needs to take into consideration its effect on the both output neurons:

We can calculate $\frac{\partial E_{o1}}{\partial net_{o1}}$ using values we calculated earlier:

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

And $\frac{\partial net_{o1}}{\partial out_{h1}}$ is equal to w_5:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

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

Plugging them in:

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

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

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

Therefore:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

Now that we have $\frac{\partial E_{total}}{\partial out_{h1}}$, we need to figure out $\frac{\partial out_{h1}}{\partial net_{h1}}$ and then $\frac{\partial net_{h1}}{\partial w}$ for each weight:

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

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

We calculate the partial derivative of the total net input to h_1 with respect to w_1 the same as we did for the output neuron:

$$net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

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

Putting it all together:

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

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

You might also see this written as:

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

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

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

We can now update w_1:

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

Repeating this for w_2, w_3, and w_4

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

Finally, we’ve updated all of our weights! When we fed forward the 0.05 and 0.1 inputs originally, the error on the network was 0.298371109. After this first round of back propagation, the total error is now down to 0.291027924.

It might not seem like much, but after repeating this process 10,000 times, for example, the error plummets to 0.0000351085. At this point, when we feed forward 0.05 and 0.1, the two outputs neurons generate 0.015912196 (vs 0.01 target) and 0.984065734 (vs 0.99 target).