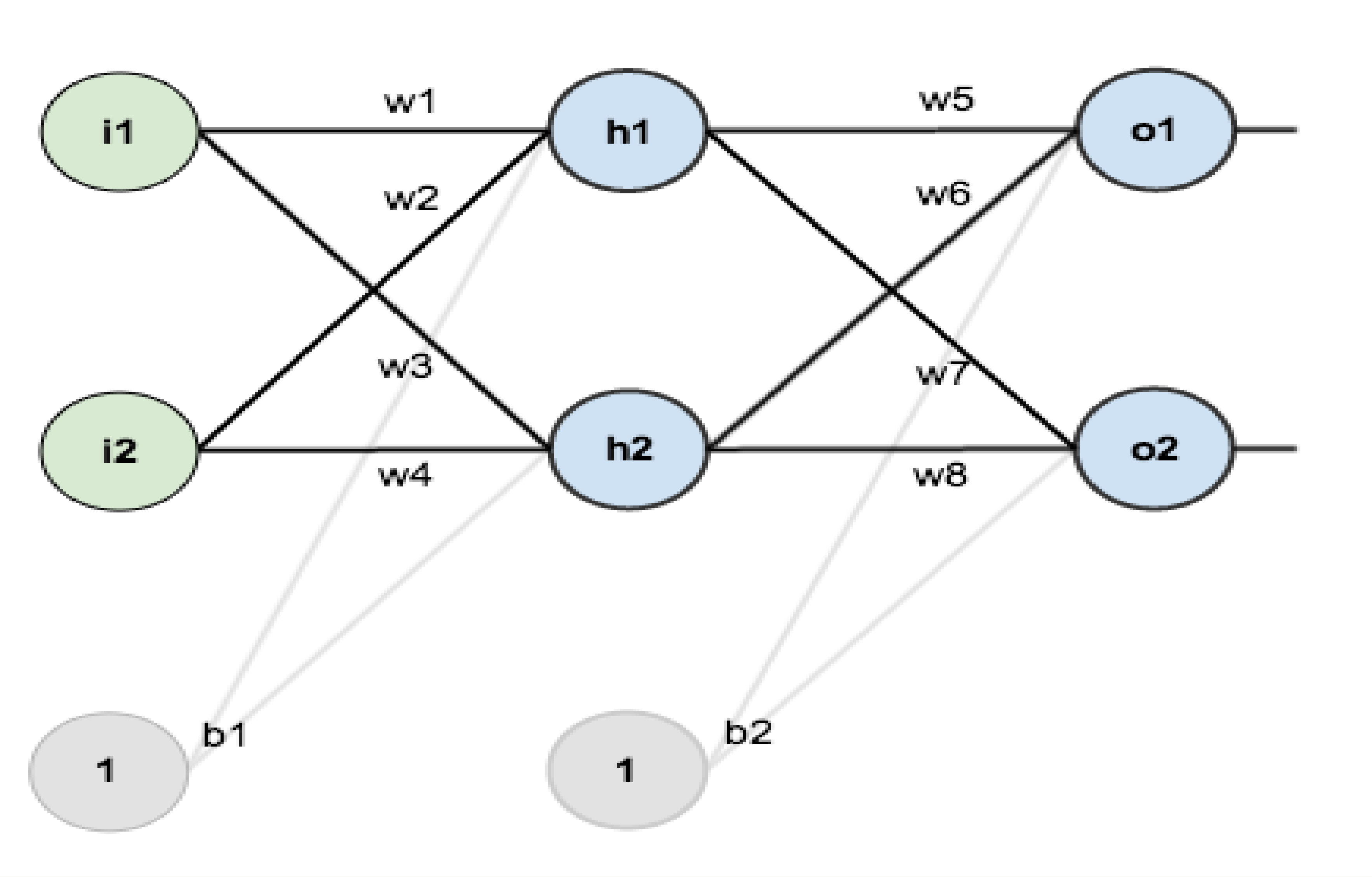
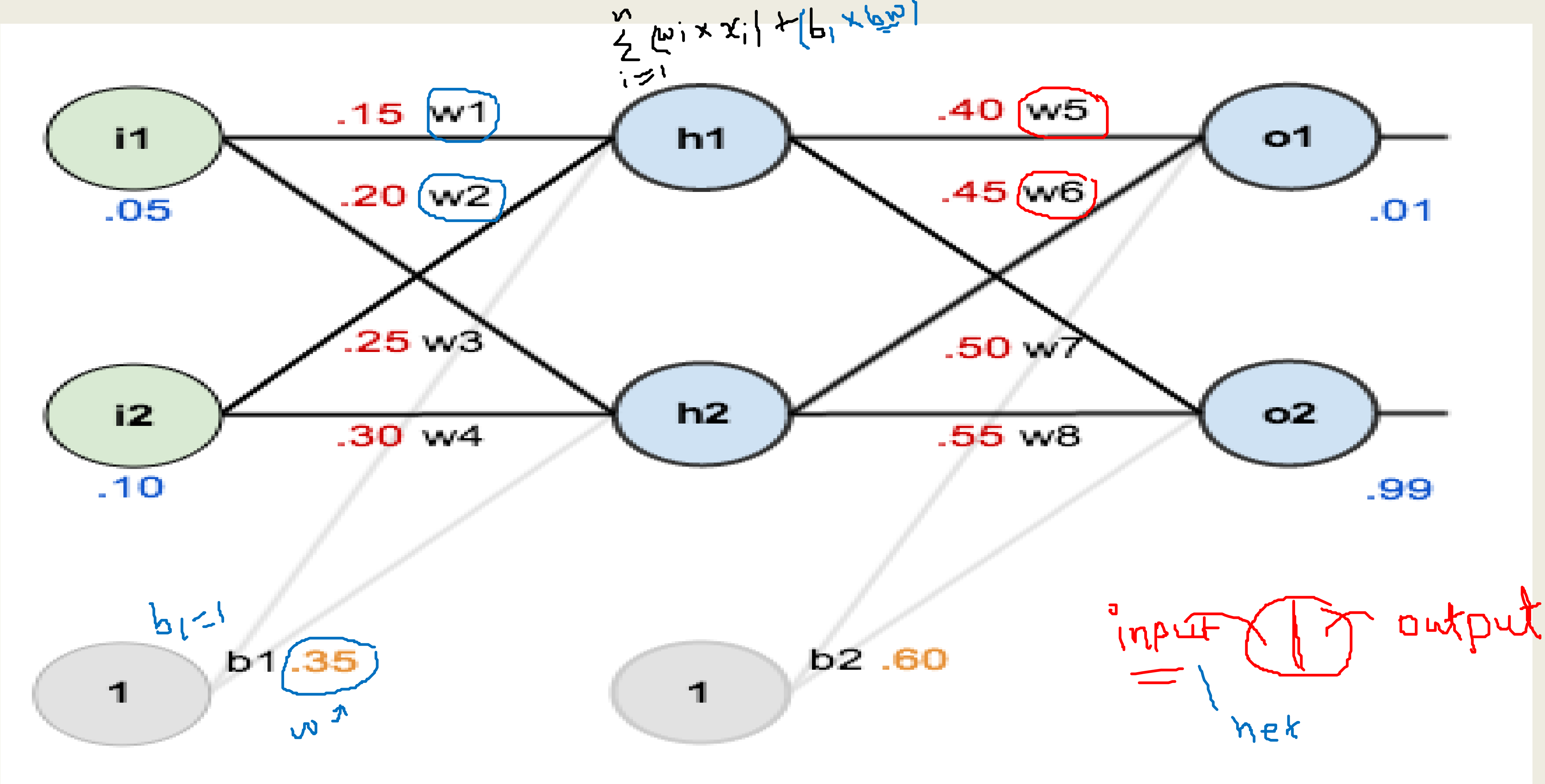
Numerical on Back-Propagation

Dr. Virendra Singh Kushwah
Assistant Professor Grade-II
School of Computing Science and Engineering
Virendra.Kushwah@vitbhopal.ac.in
7415869616

How Backpropagation Works?



In order to have some numbers to work with, here are the initial weights, the biases, and training inputs/outputs:



Step – 1: Forward Propagation

- To begin, lets see what the neural network currently predicts given the weights and biases above and inputs of 0.05 and 0.10. To do this we'll feed those inputs forward though the network.

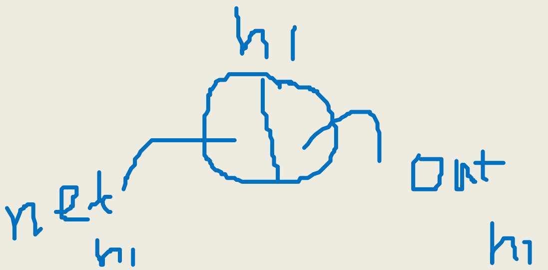
- ***We figure out the total net input to each hidden layer neuron, squash the total net input using an activation function (here we use the logistic function), then repeat the process with the output layer neurons.***

sigmoid

net h1 out

Net Input For h1:

net h1 = w1*i1 + w2*i2 + b1*1 → net h1 = 0.15*0.05 + 0.2*0.1 + 0.35*1 = 0.3775



activation function $\frac{1}{1+e^{-x}}$

Net Input For h1:

net h1 = w1*i1 + w2*i2 + b1*1 → net h1 = 0.15*0.05 + 0.2*0.1 + 0.35*1 = 0.3775

Output Of h1:

out h1 = $1/1+e^{-\text{net h1}}$ → $1/1+e^{.3775}$ = 0.593269992

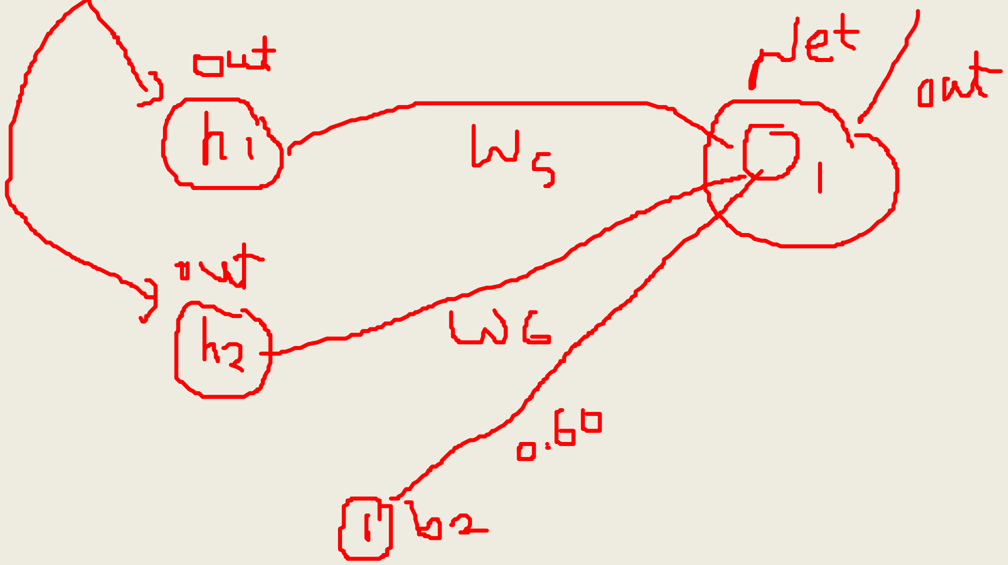
Output Of h2:

out h2 = 0.596884378

- We will repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Output For o1:

net o1 = w5*out h1 + w6*out h2 + b2*1 → 0.4*0.593269992 + 0.45*0.596884378 + 0.6*1 = 1.105905967



Output For o1:

net o1 = w5*out h1 + w6*out h2 + b2*1 → 0.4*0.593269992 + 0.45*0.596884378 + 0.6*1 = 1.105905967

Out o1 = $1/1+e^{-net\ o1}$ → $1/1+e^{-1.105905967}$ = 0.75136507

Output For o2:

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:

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

$$E_{o_1} = \frac{1}{2}(0.01 - 0.7513650 7)^2 = \frac{1}{2}(target_{o_1} - out_{o_1})^2$$

$$E_{o_2} = ?$$

Now, let's see what is the value of the error:

VIT BHOPAL
www.vitbhopal.ac.in

Error For o1:

Predicted or expected

$E\ o1 = \Sigma 1/2(target\ - output)^2$ → ½ (0.01 – 0.75136507)² = 0.274811083

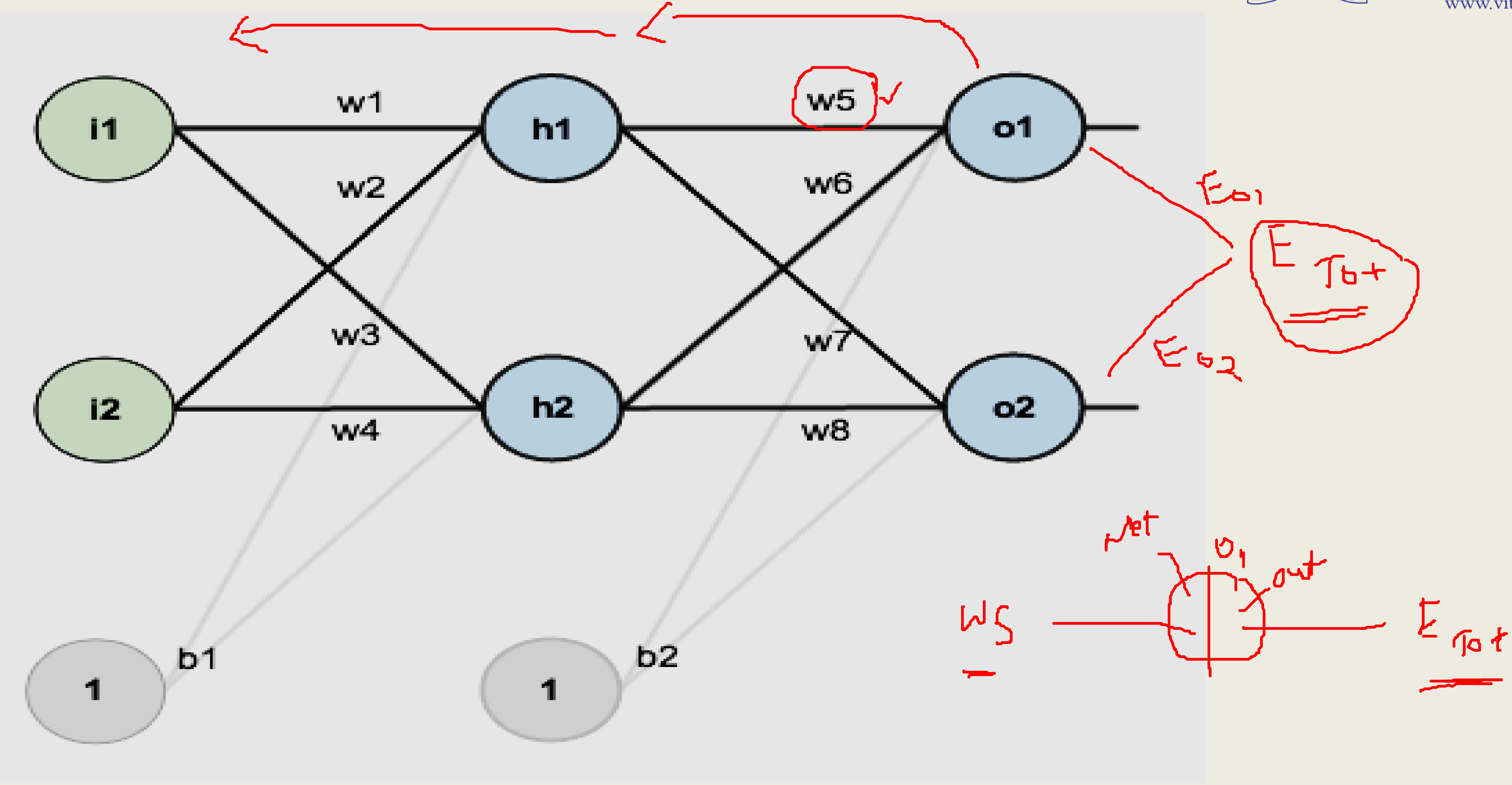
Actual

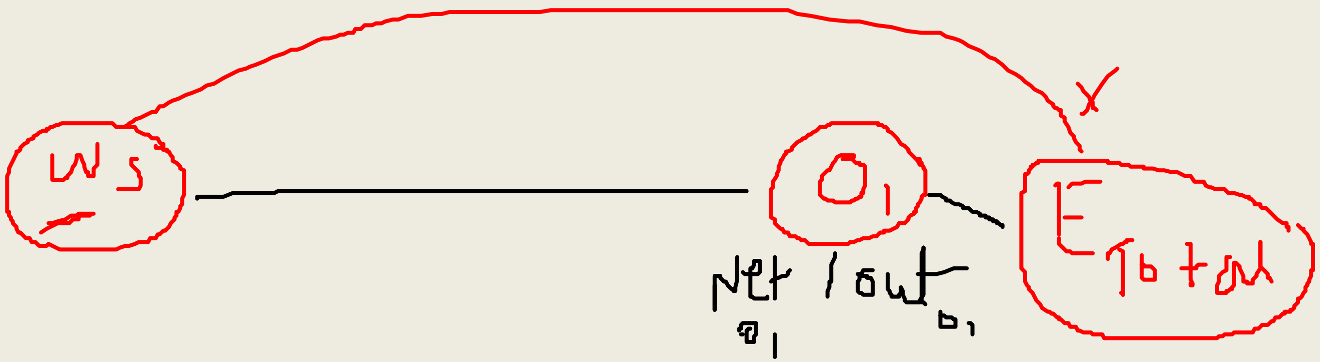
Error For o2:

E o2 = 0.023560026

Total Error:

Etotal = E o1 + E o2 → 0.274811083 + 0.023560026 = 0.298371109


i1
i2
h1
h2
o1
o2
w1
w2
w3
w4
w5
w6
w7
w8
1
b1
1
b2
Eo1
E Tot
Eo2
net
o1
out
w5
E Tot



Chain Rule

$$\frac{\partial E_{Tot}}{\partial w5} = ?$$

$$\frac{\partial E_{TOT}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial Net_{o1}} * \frac{\partial net_{o1}}{\partial w5}$$

Step – 2: Backward Propagation

- Our goal with backpropagation 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.

Output Layer

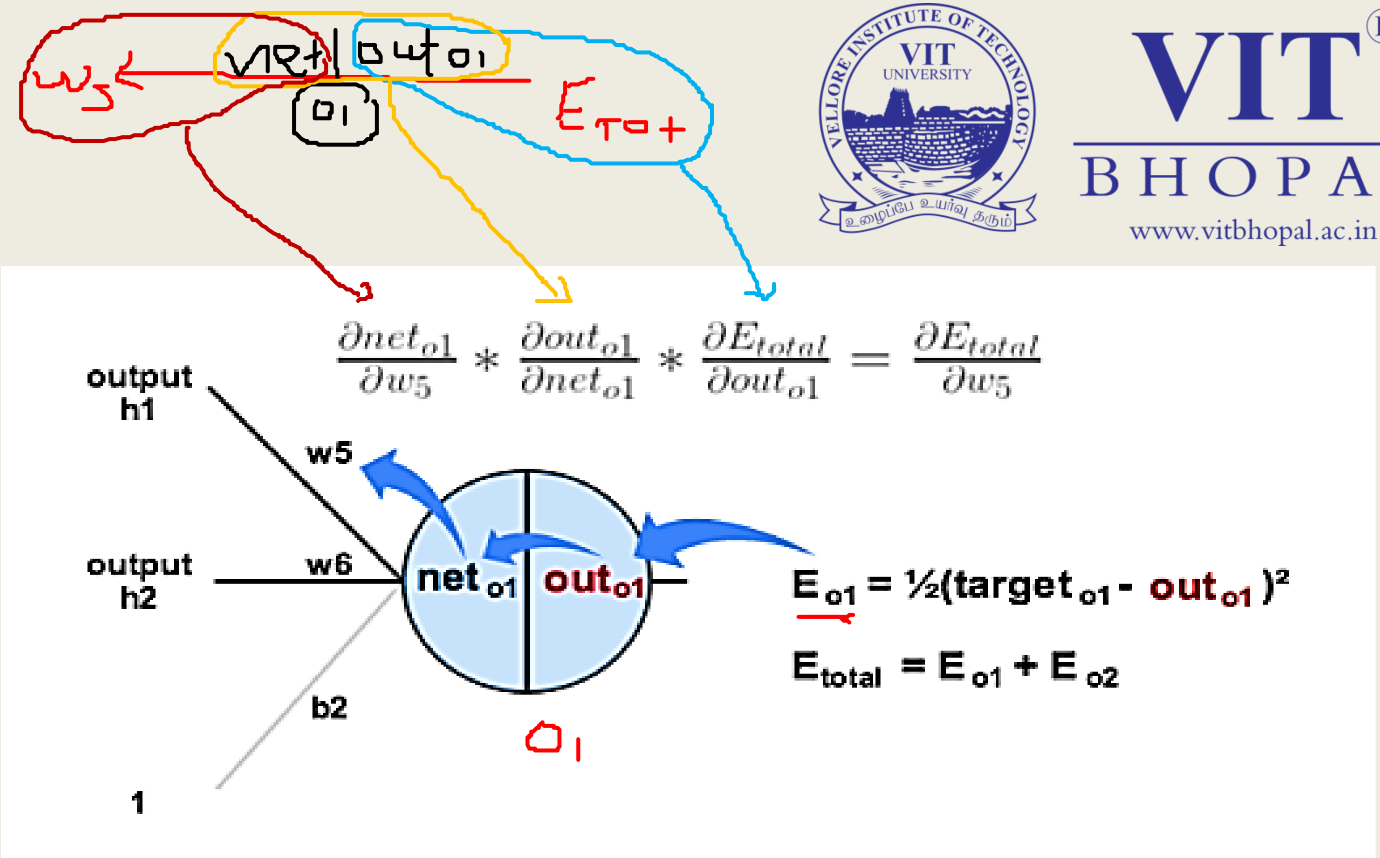
Consider w_5. We want to know how much a change in w_5 affects the total error, aka $\frac{\partial E_{total}}{\partial w_5}$.

> $\frac{\partial E_{total}}{\partial w_5}$ is read as "the partial derivative of E_{total} with respect to w_5". You can also say "the gradient with respect to w_5".

By applying the chain rule we know that:

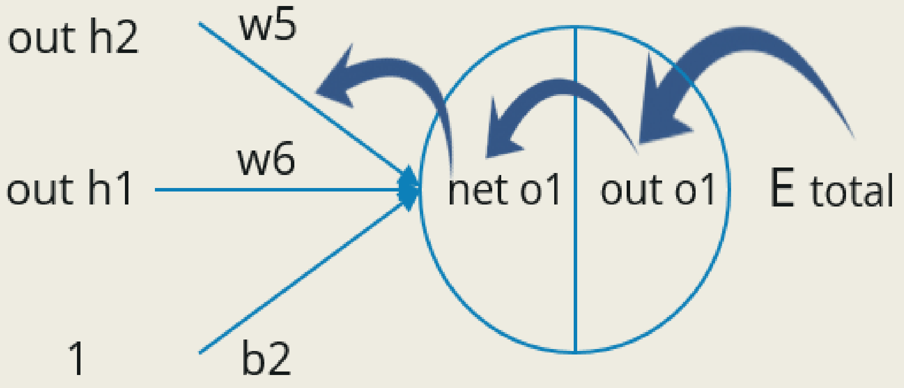
$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

- Now, we will propagate backwards. This way we will try to reduce the error by changing the values of weights and biases.



- Consider W5, we will calculate the rate of change of error w.r.t change in weight W5

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$



- Since we are propagating backwards, first thing we need to do is, calculate the change in total errors w.r.t the output O1 and O2.

Constant

$$E_{total} = 1/2(\text{target o1} - \text{out o1})^2 + 1/2(\text{target o2} - \text{out o2})^2$$

$$\frac{\delta Etotal}{\delta out\ o1} = -(\text{target o1} - \text{out o1}) = -(0.01 - 0.75136507) = 0.74136507$$

+ 0

$-(target - out)$ is sometimes expressed as $out - target$

When we take the partial derivative of the total error with respect to out_{o1}, the quantity $\frac{1}{2}(target_{o2} - out_{o2})^2$ becomes zero because out_{o1} does not affect it which means we're taking the derivative of a constant which is zero.

- Now, we will propagate further backwards and calculate the change in output O1 w.r.t to its total net input.
- The partial derivative of the logistic function is the output multiplied by 1 minus the output:

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

$$\frac{e^{+net_{o1}}}{1+e^{net_{o1}}}$$

out o1 = 1/1+e^{-neto1}

$$\frac{\delta out\ o1}{\delta net\ o1} = \text{out o1 (1 - out o1)} = 0.75136507\ (1 - 0.75136507) = 0.186815602$$

$y = x$

$\frac{dy}{dx} = 1$

$y = 25$

$\frac{dy}{dx} = 0$

- Let's see now how much does the total net input of O1 changes w.r.t W5?

Constant

$$\text{net o1} = \text{w5} * \text{out h1} + \text{w6} * \text{out h2} + \text{b2} * 1$$

$$\frac{\delta net\ o1}{\delta w5} = 1 * \text{out h1}\ w5^{(1-1)} + 0 + 0 = 0.593269992$$

Step – 3: Putting all the values together and calculating the updated weight value

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$ → 0.082167041

Let's calculate the updated value of W5:

old w5

$$w5^{+} = w5 - n\frac{\delta Etotal}{\delta w5}$$ → $w5^{+} = 0.4 - 0.5 * 0.082167041$

eta

learning rate

$w5^{+}$

Updated w5 → 0.35891648

$w5 = 0.40$

$w5^{+} = 0.358$

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

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

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

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 (ie, we use the original weights, not the updated weights, when we continue the backpropagation algorithm).