**International University for Science & Technology (IUST)**
**Department of Computer &Informatics Engineering**
**Neural Networks unit (5)**

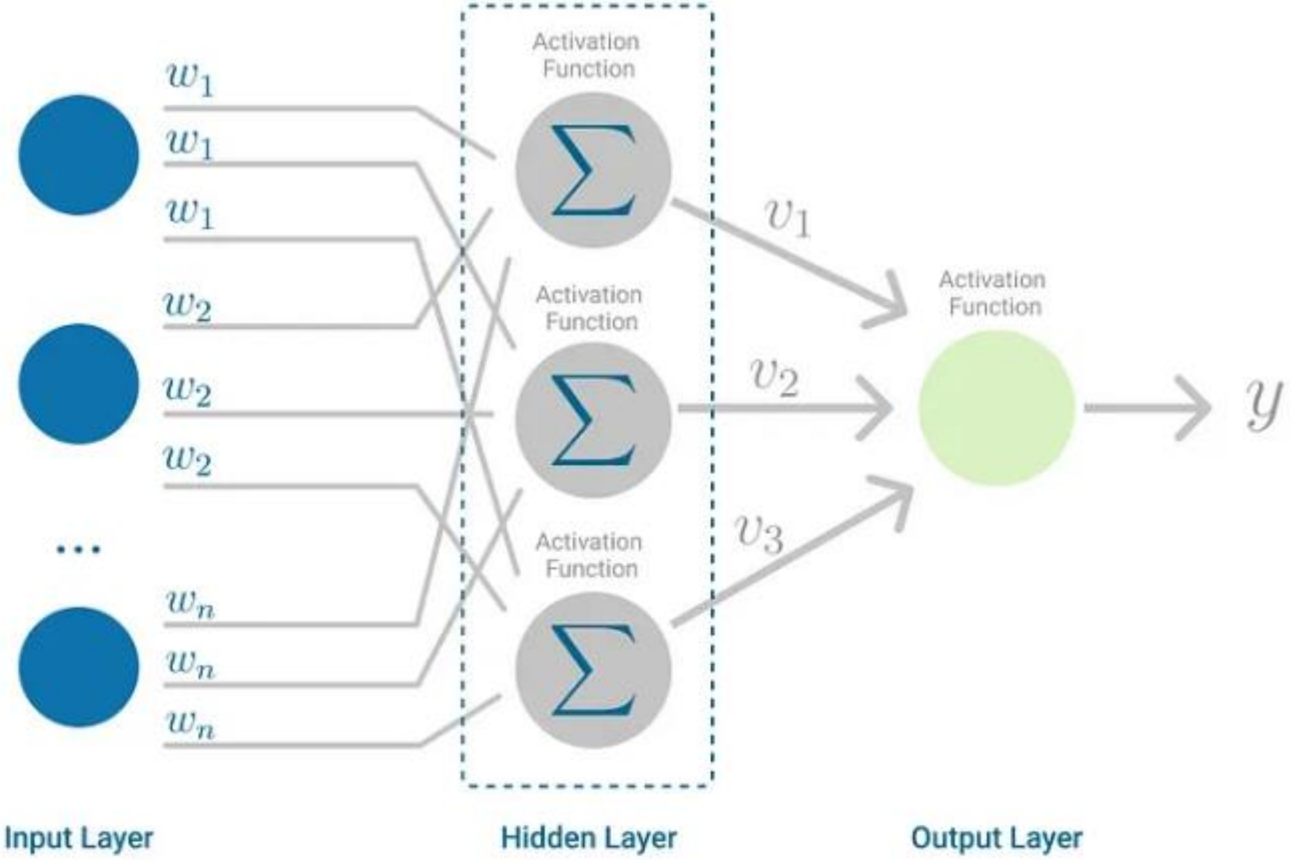
# Multilayer Perceptron (MLP) & Backpropagation

Neural Networks

Dr. Ali Mayya

# MLP Architecture

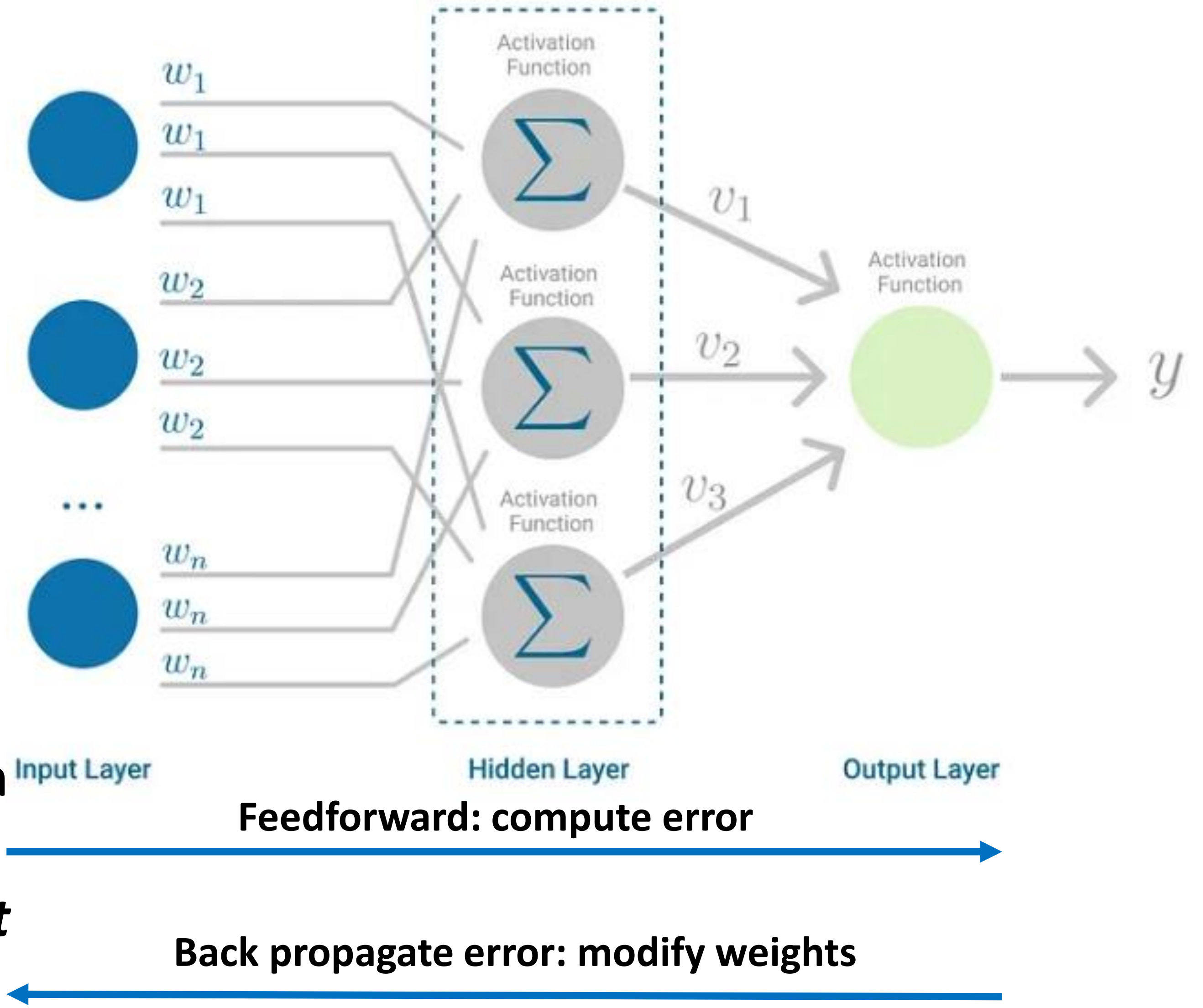
- **Multilayer Perceptron MLP** is a feedforward model, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron.
- The ***difference*** is that each linear combination is propagated to the next layer.
- Each layer is **feeding** the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.
- It's trained by the **Backpropagation** learning algorithm in order to minimize the cost (error) function.




https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

# Backpropagation Learning Algorithm

- **Backpropagation** is a learning mechanism that allows the Multilayer Perceptron to iteratively adjust the weights in the network, with the goal of minimizing the cost function.
- Consists of two steps:
    - ***Feed forward*** part: compute the error between the target and the actual output.
    - ***Backpropagation***: propagate error in a backward way to modify all weights.
    - The backpropagation is mainly based on derivative of the error and propagate it backward and this is called the ***Gradient descent*** algorithm.

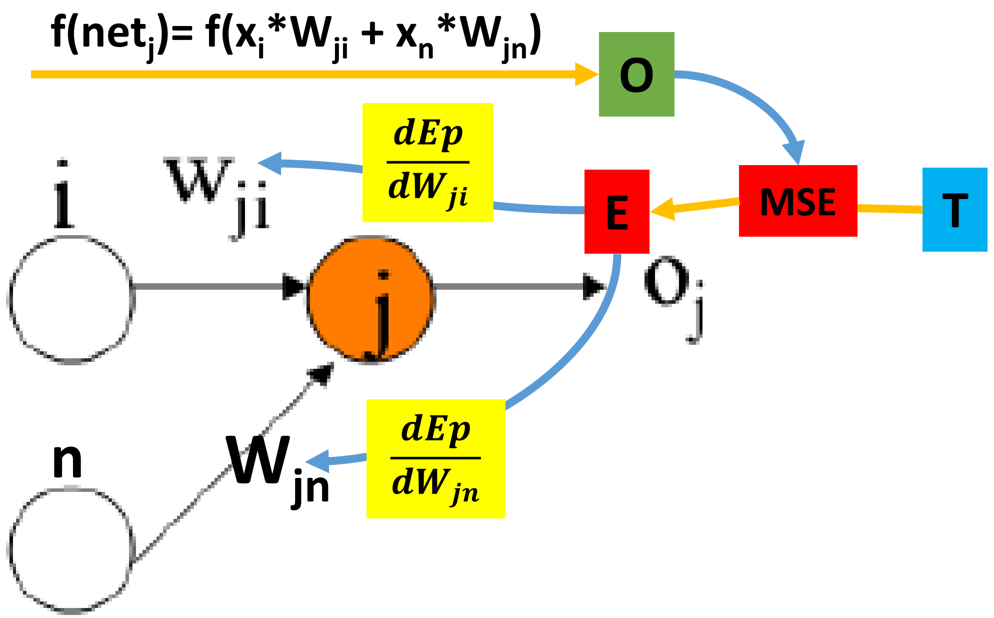



https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

# Backpropagation Learning Algorithm

- **Simply** if we have two input neurons and one output neurons:
- **In the forward part** we compute the output:
- **$O_j$ =f($net_j$)= f($x_i$*$W_{ji}$ + $x_n$*$W_{jn}$)**

  where f is a non-linear activation function like (sigmoid)
- Then, we calculate the error (MSE):
- E=$\frac{1}{2N}(y_i - \hat{y}_i)^2$ for one sample: E=$\frac{1}{2}(y_i - \hat{y}_i)^2$
- In the backward part: we compute the derivative of the error in terms of weight:




$$\frac{dE_p}{dw_{ji}} = \frac{dE}{do_j} \times \frac{do_j}{dnet_j} \times \frac{dnet_j}{dw_{ji}}$$

$$\frac{dE_p}{dw_{ji}} = -(t_j - o_j) \times f'(net_j) \times o_i$$

**$O_i$ input of node i ($net_i$)**

**F' derivative of sigmoid**
**Z=f($net_j$)=Sigmoid(net)= $\frac{1}{1+e^{-net}}$**
**F'= Z(1-Z)**

**E=0.5*($t_p$- $O_p$)$^2$**
**Derivative=**
**-2*0.5(t-o)=**
**-(t-o)**

- **Update weights:**

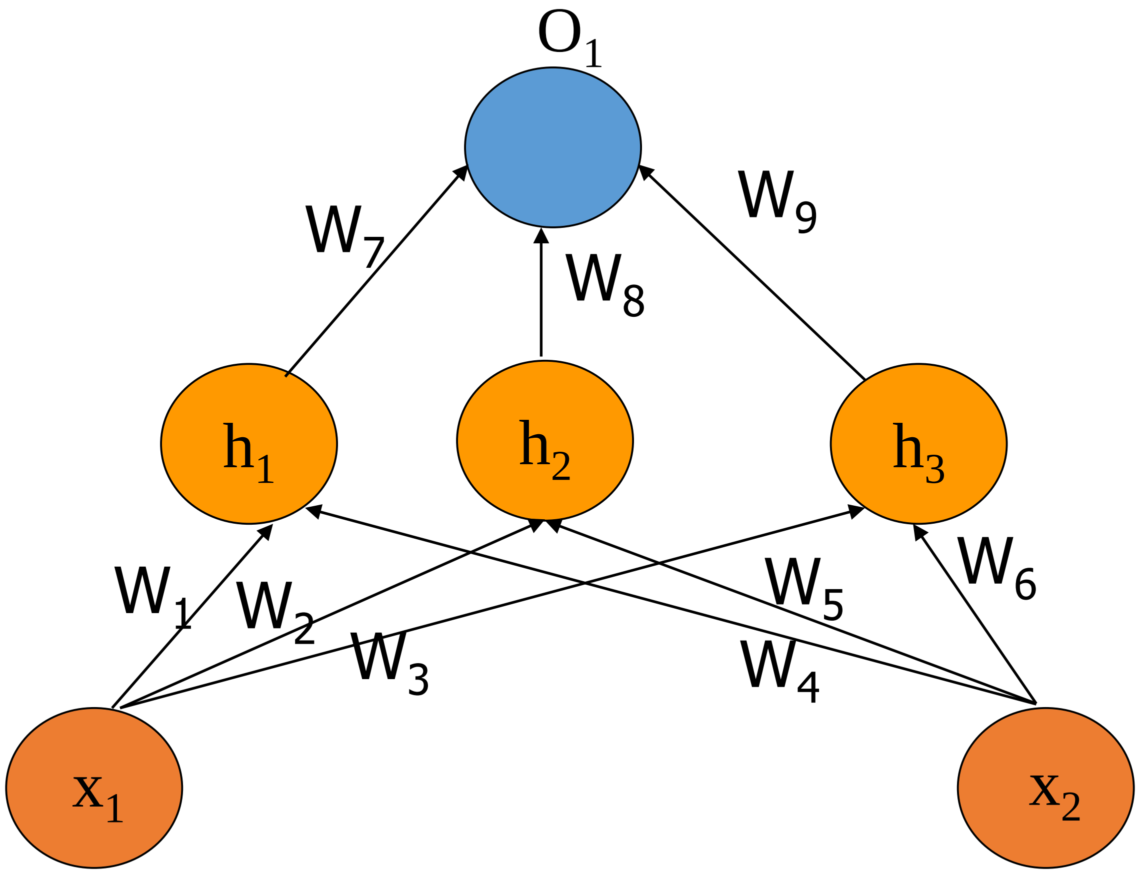
$$W_{ji}(new) = W_{ji}(old) - \alpha \frac{dEp}{dW_{ji}}$$

# Backpropagation Learning Algorithm

- **In case of multi-layers**

$$\frac{dE_p}{dw_{ji}} = \frac{dE}{do_j} \times \frac{do_j}{dnet_j} \times \frac{dnet_j}{dw_{ji}}$$

$$\frac{dE_p}{dw_{ji}} = -(t_j - o_j) \times f'(net_j) \times o_i$$




$$\frac{dE_p}{W_7} = -(t1 - O1) * f'(net) * h1$$

$$\frac{dE_p}{W_8} = -(t1 - O1) * f'(net) * h_2$$

$$\frac{dE_p}{W_9} = -(t1 - O1) * f'(net) * h_3$$

$$\frac{dE_p}{W1} = \frac{dE}{dOut_{h1}} * \frac{dOut_{h1}}{dnet_{h1}} * \frac{dnet_{h1}}{dW_1} = \frac{dE}{dO_1} * \frac{dO_1}{dnet_{O1}} * \frac{dnet_{O1}}{dOut_{h1}} * \frac{dOut_{h1}}{dnet_{h1}} * \frac{dnet_{h1}}{dW_1} =$$

$net_{o1}$= h1*W7 + h2*W8 + h3*W9

$net_{h1}$= $X_1$*W1 + $X_2$*W4

$$\frac{dE_p}{W1} = -(t1 - O1) * f'(net) * W_7 * Outh_1(1 - Outh_1) * X_1$$

$$\frac{dE_p}{W2} = -(t1 - O1) * f'(net) * W_8 * Outh_2(1 - Outh_2) * X_1$$

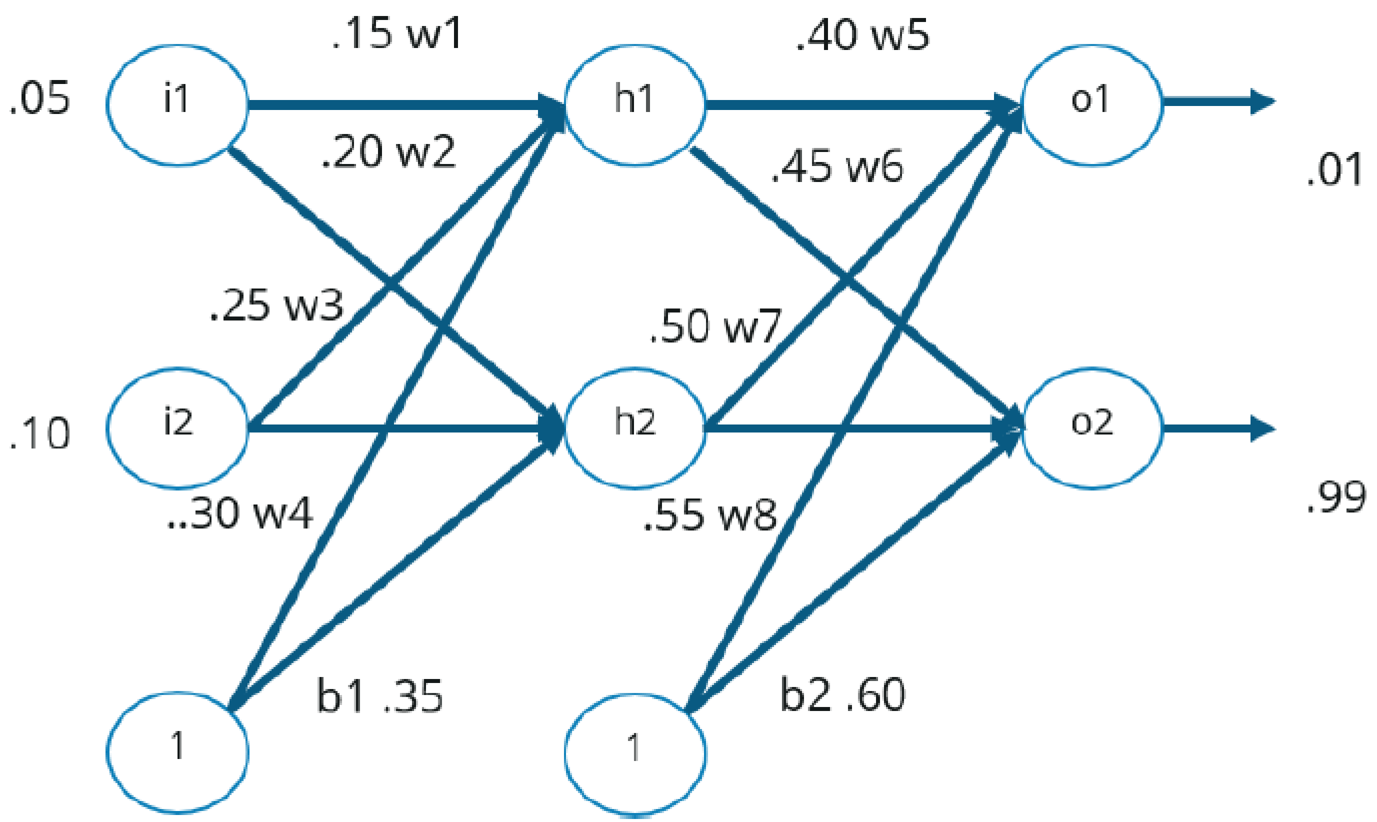
# Backpropagation Learning Algorithm

- **In case of multi-outputs**
- **Compute total error as:**

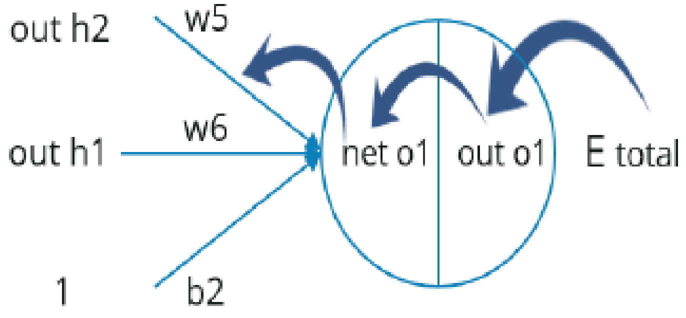
$$E_{total} = \sum \frac{1}{2}(t-o)^2$$

$$E_{o1} = \frac{1}{2}(t_1 - o_1)^2$$

$$E_{o2} = \frac{1}{2}(t_2 - o_2)^2$$




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




$$\frac{dE_{total}}{E_{o1}} = -(t_1 - o_1)$$

$$\frac{dE_{total}}{E_{o2}} = -(t_2 - o_2)$$

# Backpropagation Learning Algorithm (Huge example)

- **1. Forward propagation:**

$net_{h1}$ = w1*i1 + w2*i2+b1*1=

$net_{h1}$ = 0.15*0.05+0.2*0.1+0.35*1 = 0.3775

$net_{h2}$ = w3*i1 + w4*i2+b1*1=

$net_{h2}$ = 0.25*0.05+0.3*0.1+0.35*1 = 0.3925

$$out_{h1} = \frac{1}{1+e^{-neth1}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

$$out_{h2} = \frac{1}{1+e^{-neth2}} = \frac{1}{1+e^{-0.3925}} = 0.596884378$$

$net_{o1}$ = w5*$out_{h1}$ + w6*$out_{h2}$ +b2*1=

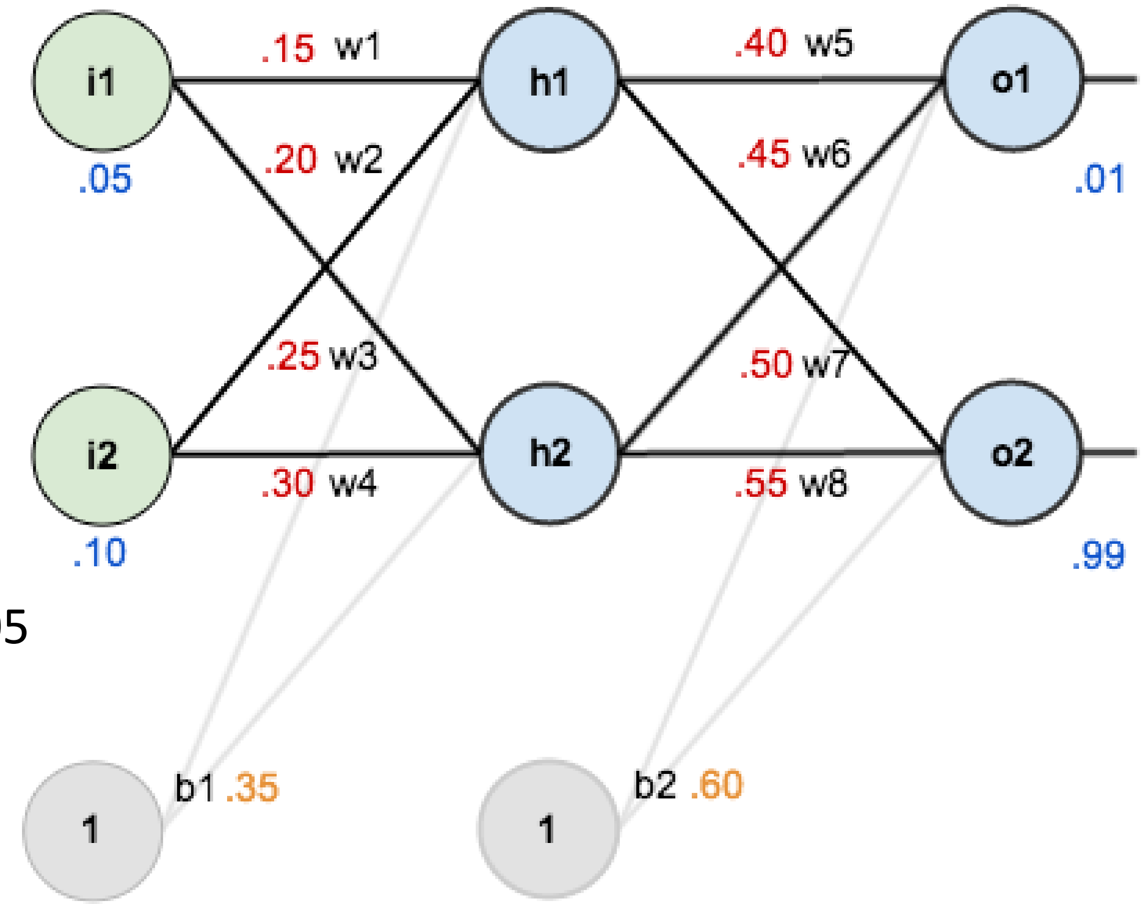
$net_{o1}$ = 0.4*0.593269992+0.45*0.596884378+0.6*1 = 1.105905

$net_{o2}$ = w7*$out_{h1}$ + w8*$out_{h2}$ +b2*1=

$net_{o2}$ = 0.5*0.593269992+0.55*0.596884378+0.6*1 = 1.225

$$out_{o1} = \frac{1}{1+e^{-neto1}} = \frac{1}{1+e^{-1.1059}} = 0.75136507$$

$$out_{o2} = \frac{1}{1+e^{-neto2}} = \frac{1}{1+e^{-1.1059}} = 0.772928465$$




**Alpha (learning rate) = 0.5**

# Backpropagation Learning Alg

- **2. Back propagation:**

$$E_{total} = \sum \frac{1}{2}(t-o)^2$$

$E_{o1} = \frac{1}{2}(t_1 - o_1)^2$=0.5*(0.01-0.75136507)² = 0.27481183

$E_{o2} = \frac{1}{2}(t_2 - o_2)^2$= 0.5*(0.99-0.772928465)² $= 0.023560026$

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

$\frac{dE_{total}}{E_{o1}} (\frac{dE_{o1}}{dOut_{o1}}) = -(t_1 - o_1)$ =-(0.01-0.75136507)= 0.74136507

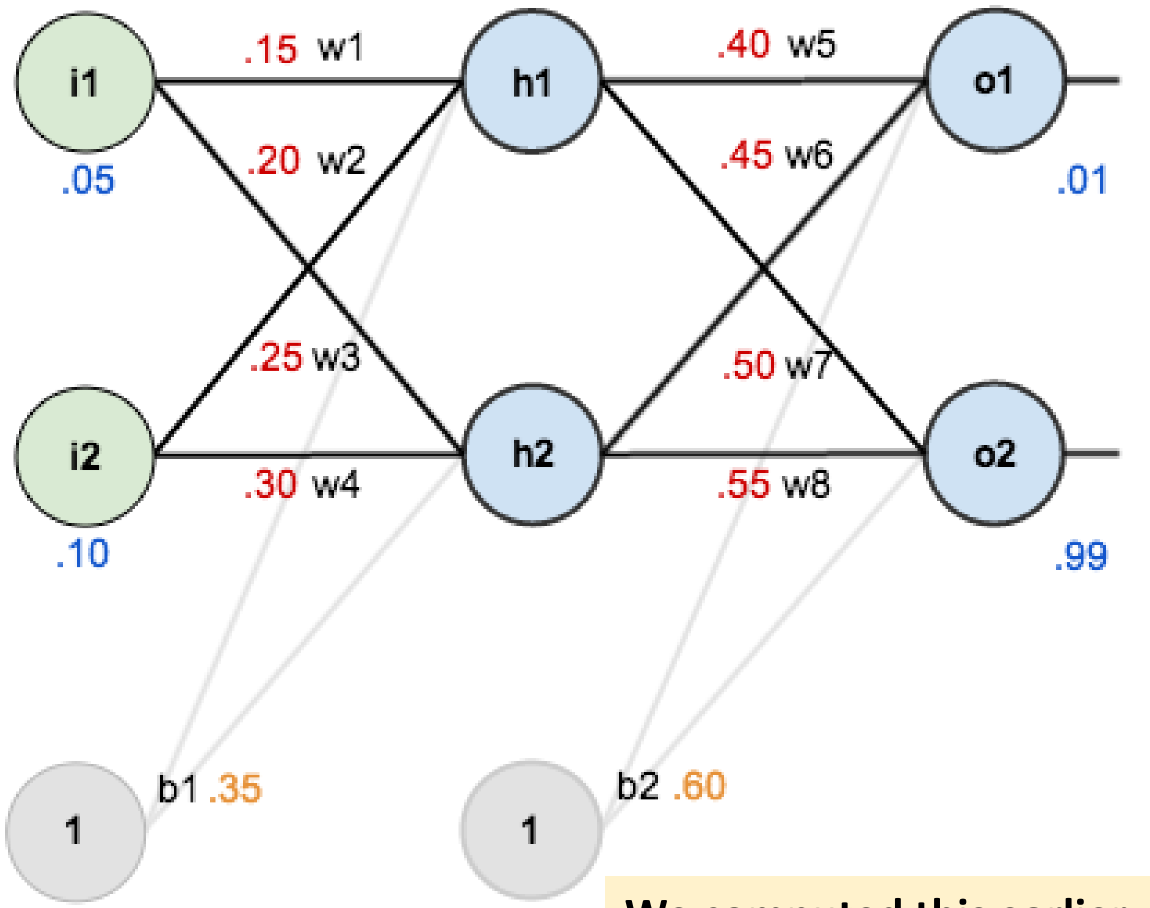
$\frac{dE_{total}}{E_{o2}} = -(t_2 - o_2)$ = -(0.99-0.772928465) =-0.217071535

$$\frac{dE_p}{dw_{ji}} = \frac{dE}{do_j} \times \frac{do_j}{dnet_j} \times \frac{dnet_j}{dw_{ji}}$$

$$\frac{dE_p}{dw_{ji}} = -(t_j - o_j) \times f'(net_j) \times o_i$$

$net_{o1}$ = $Out_{h1}$*w5 + $out_{h2}$*W6+b2
→ $dnet_{o1}$/dw5 = $Out_{h1}$

$\frac{dE_{total}}{dW_5} = -(t_1 - o_1) \ * f'(net_{o1}) *$ $dnet_{o1}$/dw5 $= -(t_1 - o_1)$ *$out_{o1}$(1-$out_{o1}$)*$Out_{h1}$
= -0.74136507 *0.75136507 (1-0.75136507)* 0.593269992 = 0.082167041




**We computed this earlier**

$out_{o1} = 0.75136507$
$out_{o2} = 0.772928465$
$net_{o1}$ = 1.105905    $net_{o2}$ = 1.225
$net_{h1}$ = 0.3775    $net_{h1}$ = 0.3925
$out_{h1} = 0.593269992$
$out_{h2} = 0.596884378$

# Backpropagation Learning Algorithm (Huge example)

- **2. Back propagation:**

$$\frac{dE_{total}}{dW_5} = -(t_1 - o_1) \; {}^*\text{out}_{o1}(1\text{-out}_{o1}){}^*\text{Out}_{h1}$$

= -0.74136507 *0.75136507 (1-0.75136507)* 0.593269992 = 0.082167041

$$\frac{dE_{total}}{dW_6} = -(t_1 - o_1) \; {}^*\text{out}_{o1}(1\text{-out}_{o1}){}^*\text{Out}_{h2}$$

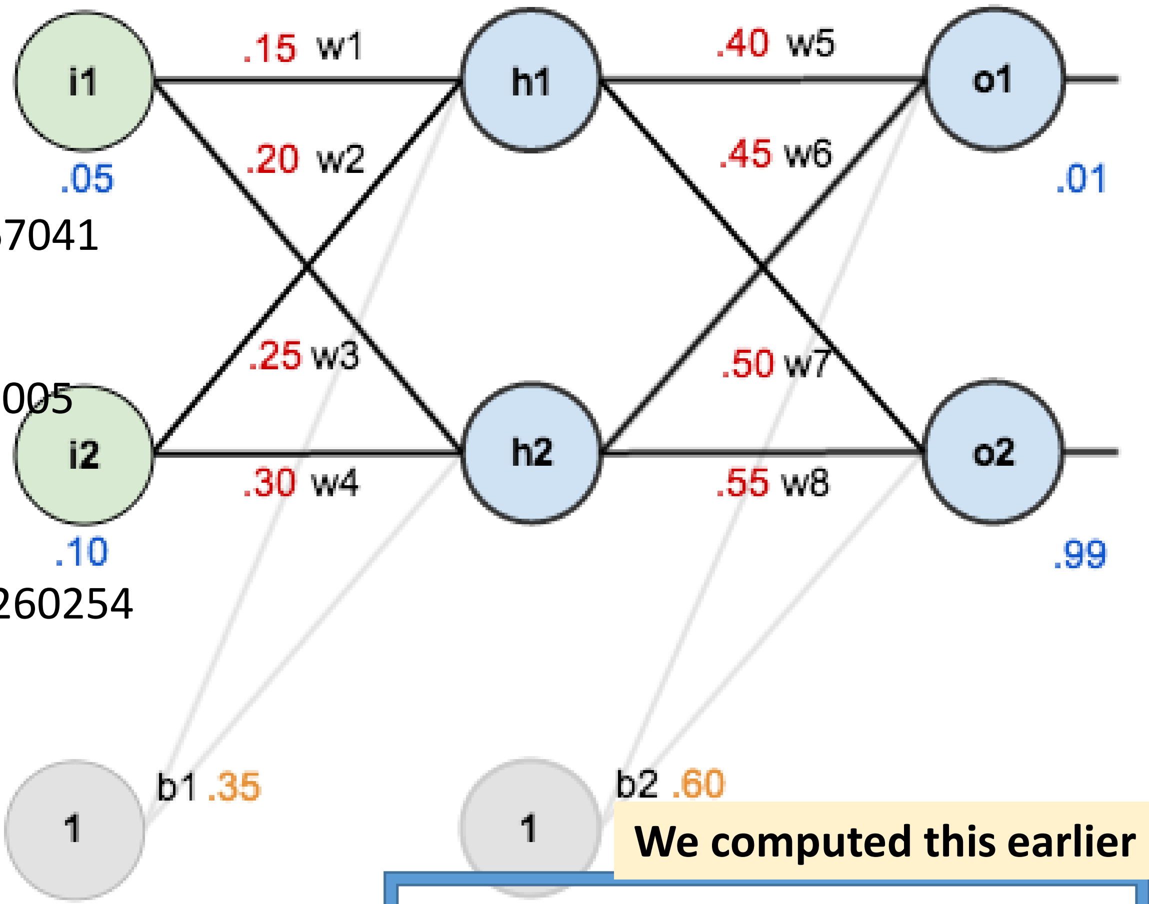
= 0.74136507 *0.75136507 (1-0.75136507)* 0.596884378 =0.082662005

$$\frac{dE_{total}}{dW_7} = -(t_2 - o_2) \; {}^*\text{out}_{o2}(1\text{-out}_{o2}){}^*\text{Out}_{h1}$$

=-0.217071535 *0.772928465(1-0.772928465)* 0.593269992 =-0.02260254

$$\frac{dE_{total}}{dW_8} = -(t_2 - o_2) \; {}^*\text{out}_{o2}(1\text{-out}_{o2}){}^*\text{Out}_{h2}$$

=-0.217071535 *0.772928465(1-0.772928465)* 0.596884378 =-0.0227402422




$W_5^{new}$= $W_5$- alpha * $\frac{\delta E_i}{\delta w5}$= 0.4-0.5*0.082167041=0.358916479

$W_6^{new}$= $W_6$- alpha * $\frac{\delta E_i}{\delta w6}$= 0.45-0.5*0.082662005= 0.408668975

$W_7^{new}$= $W_7$- alpha * $\frac{\delta E_i}{\delta w7}$= 0.5-0.5* $-0.02260254$= 0.51130127

$W_8^{new}$= $W_8$- alpha * $\frac{\delta E_i}{\delta w8}$= 0.55-0.5* $-0.0227402422$= 0.561370121

Update Weights of hidden layer

**We computed this earlier**

$$out_{o1} = 0.75136507$$

$$out_{o2} = 0.772928465$$

$net_{o1}$ = 1.105905 $net_{o2}$ = 1.225

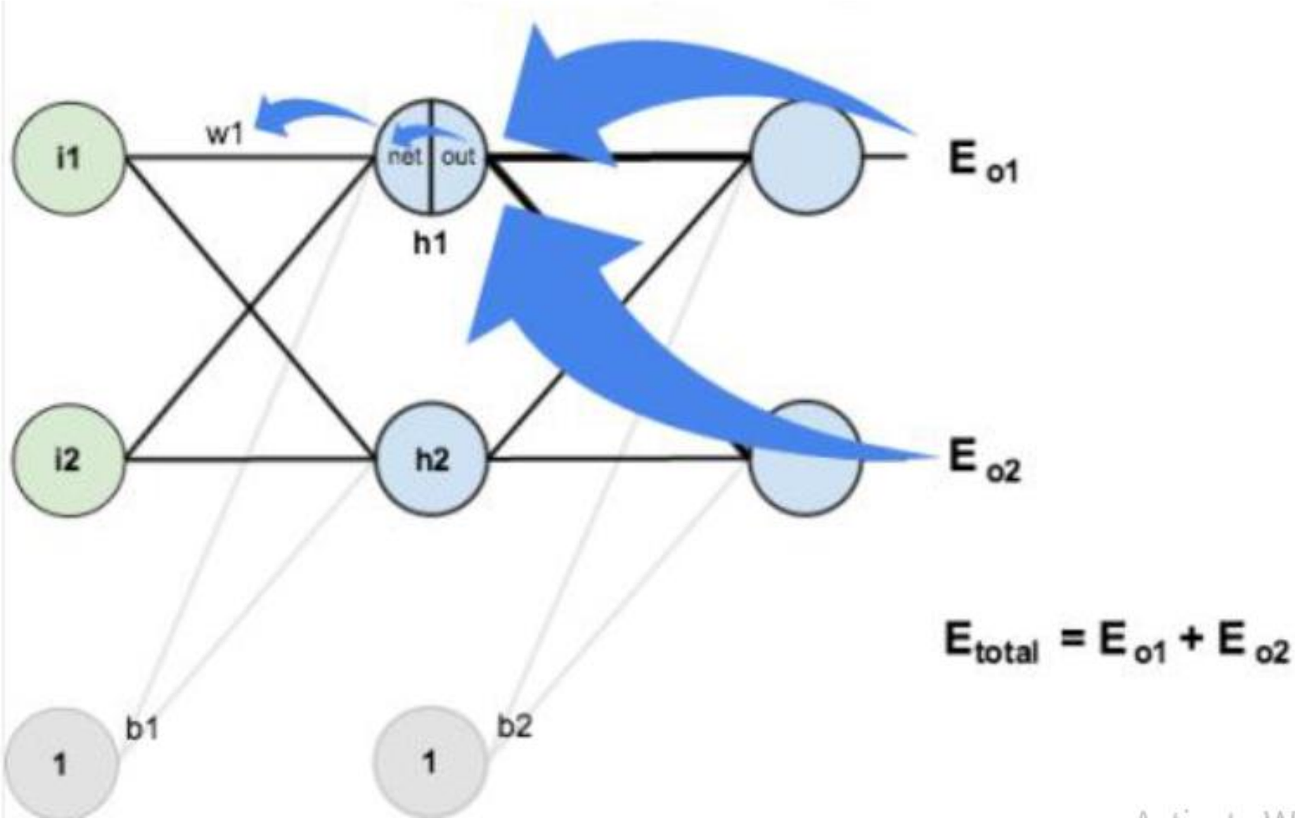
$net_{h1}$ = 0.3775 $net_{h1}$ = 0.3925

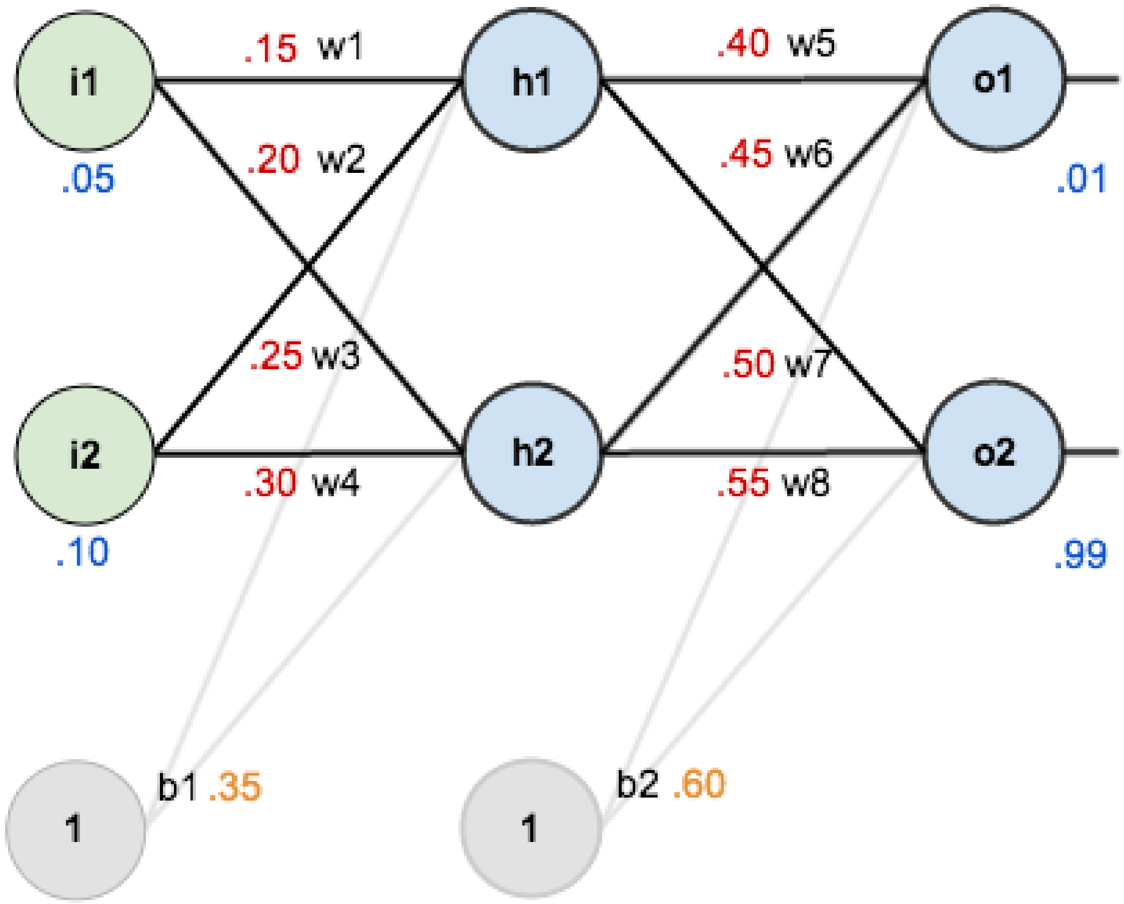
$$out_{h1} = 0.593269992$$

$$out_{h2} = 0.596884378$$

# Backpropagation Learning Algorithm (Huge example)

- **2. Back propagation: now propagate to input-hidden layer weights.**
- **We will modify weight W1 and the rest will maintain the same way.**



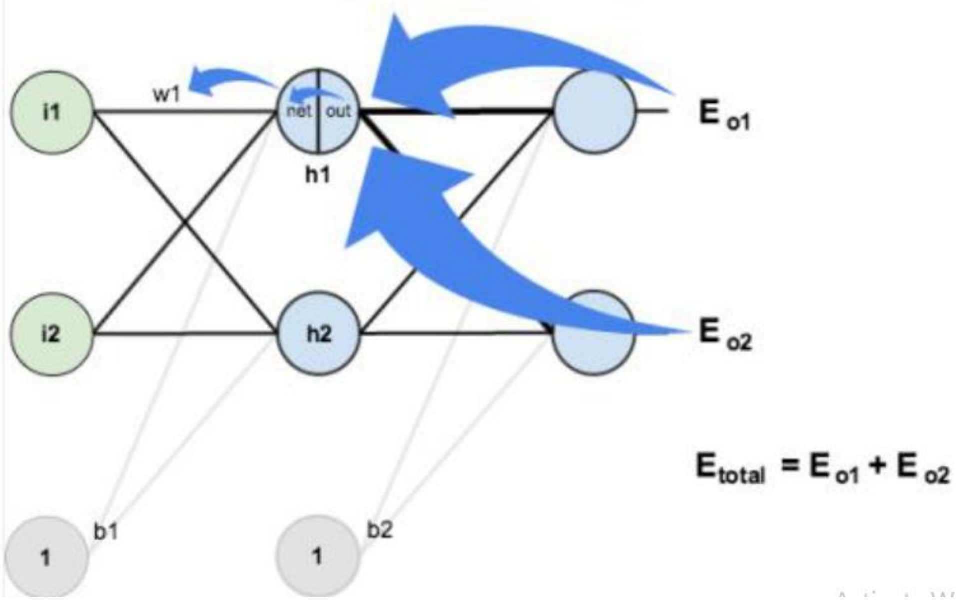
# Backpropagation Learning Algorithm (Huge example)

- **2. Back propagation: now propagate to input-hidden layer weights.**

$$\frac{dE_{total}}{W_1} = \frac{dE_{total}}{dOut_{h1}} * \frac{dOut_{h1}}{dnet_{h1}} * \frac{dnet_{h1}}{dW_1}$$

1 1 2 3

But! $\frac{dE_{total}}{dOut_{h1}} = \frac{dE_{o1}}{dOut_{h1}} + \frac{dE_{o2}}{dOut_{h2}}$




$$\frac{dE_{o1}}{dOut_{h1}} = \frac{dEo_1}{dOut_{o1}} * \frac{dOut_{o1}}{dnet_{o1}} * \frac{dnet_{o1}}{dOut_{h1}}$$

$net_{o1} = Out_{h1}*w5 + out_{h2}*W6+b2$
→ $dnet_{o1}/dOut_{h1} = W_5$

$$\frac{dE_{o1}}{dOut_{h1}} = 0.74136507 * 0.75136507 * (1 - 0.75136507) * 0.4$$

$$\frac{dE_{total}}{dOut_{h1}} = 0.055399424$$

$$\frac{dE_{o2}}{dOut_{h1}} = -0.217071535 * 0.772928465 * (1 - 0.772928465) * 0.5$$

$net_{o2} = Out_{h1}*w7 + out_{h2}*W8+b2$
→ $dnet_{o1}/dOut_{h1} = W_7$

$$\frac{dE_{o2}}{dOut_{h1}} = -0.019049119$$

**We computed this earlier**

$$\frac{dE_{total}}{E_{o1}} \left(\frac{dE_{o1}}{dOut_{o1}}\right) = 0.74136507$$

$$\frac{dE_{total}}{E_{o2}} \left(\frac{dE_{o2}}{dOut_{o2}}\right) = -0.217071535$$

$out_{o1} = 0.75136507$

$out_{o2} = 0.772928465$ $out_{h1} = 0.593269992$

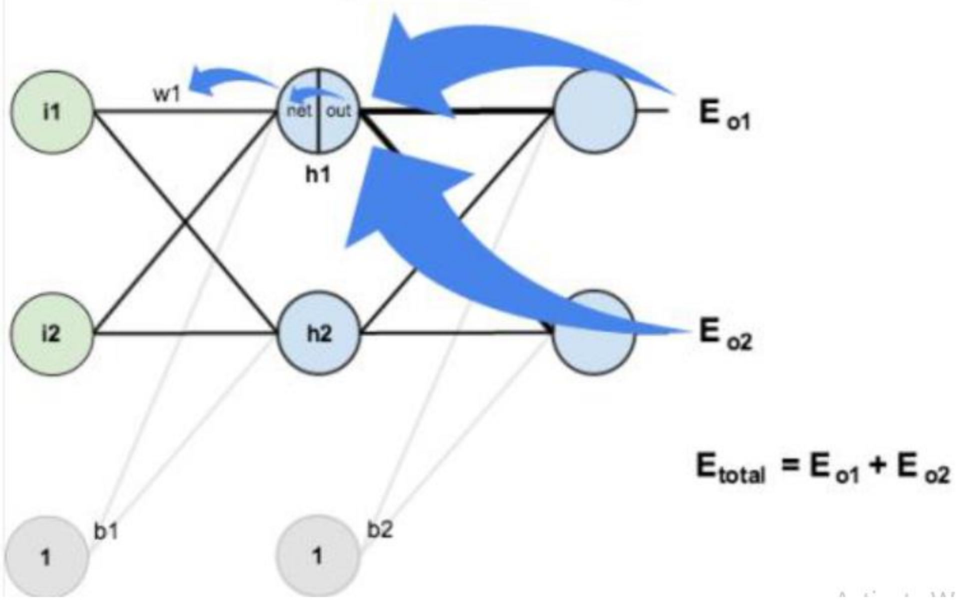
2 $\frac{dOut_{h1}}{dnet_{h1}} = Out_{h1}*(1-Out_{h1}) = 0.593269992 * (1 - 0.593269992) = 0.2413007$

3 $\frac{dnet_{h1}}{dW_1} = \frac{d(i1 * W1 + i2 * W2 + b1)}{dW_1} = i1 = 0.05$

$$\frac{dE_{total}}{W_1} = (0.055399424 - 0.019049119) * 0.2413007 * 0.05 = 0.000438568$$

# Backpropagation Learning Algorithm (Huge example)

- **2. Back propagation: now propagate to input-hidden layer weights.**




$$\frac{dE_{total}}{W_1} = 0.000438568$$

$$W1(new) = W1(old) - alpha * \frac{dE_{total}}{W_1} = 0.15\text{-}0.5*(0.000438568)$$

$$W1(new) = 0.15\text{-}0.5*(0.000438568) = 0.149780716$$

*Similarly*:

$$\frac{dE_{total}}{W_2} = (0.055399424 - 0.019049119) * 0.2413007 * 0.1 = 0.000877135404$$

$$W2(new) = 0.2\text{-}0.5*(0.000877135404) = 0.19995614323$$

$$W3(new) = 0.24975114$$

$$W4(new) = 0.29950229$$

**We computed this earlier**

$$\frac{dE_{total}}{E_{o1}}\left(\frac{dE_{o1}}{dOut_{o1}}\right) = 0.74136507$$

$$\frac{dE_{total}}{E_{o2}}\left(\frac{dE_{o2}}{dOut_{o2}}\right) = -0.217071535$$

$$out_{o1} = 0.75136507$$

$$out_{o2} = 0.772928465 \qquad out_{h1} = 0.593269992$$

*End of the first training epoch*