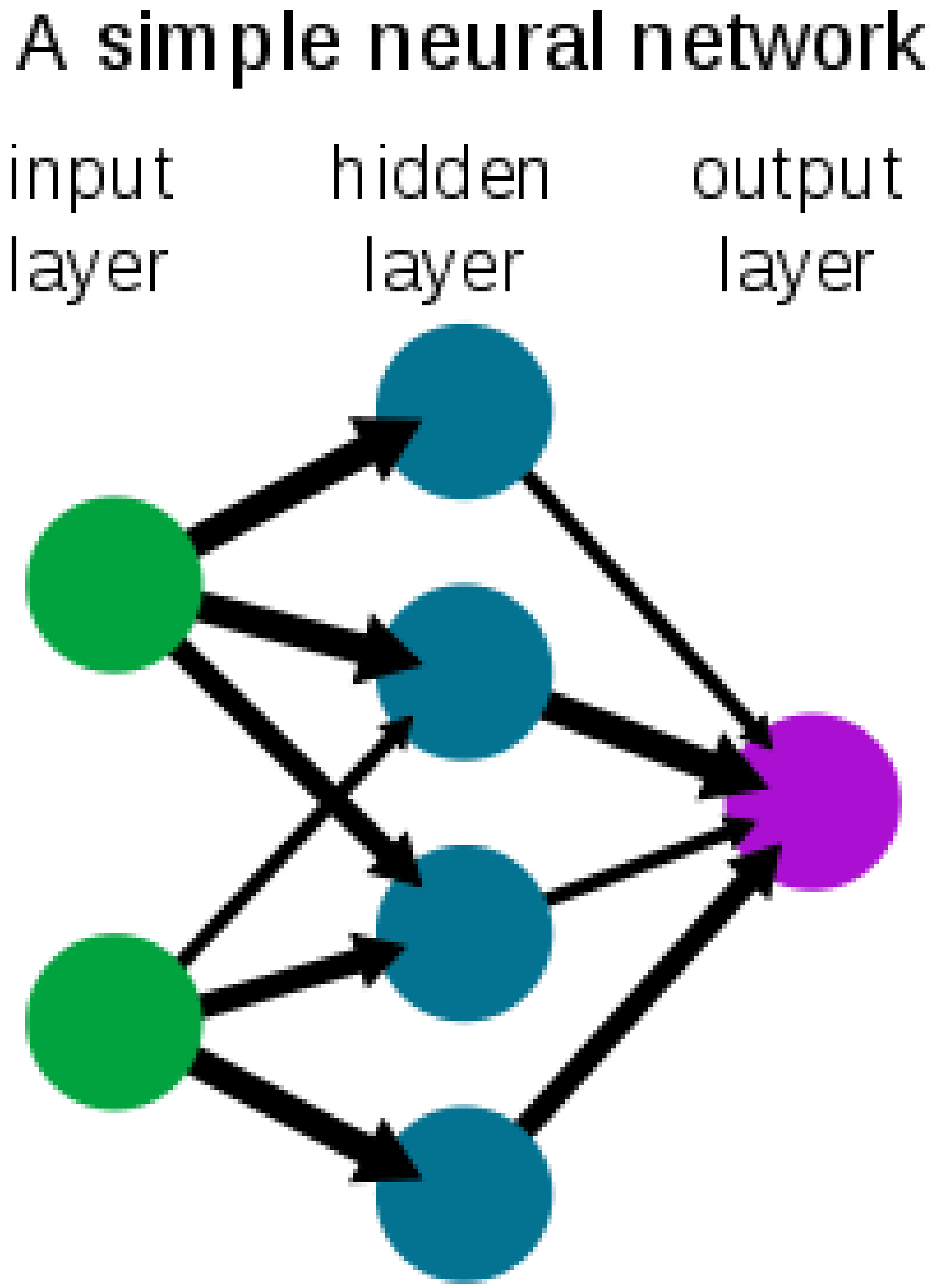
Backpropagation Algorithm

(Concept)

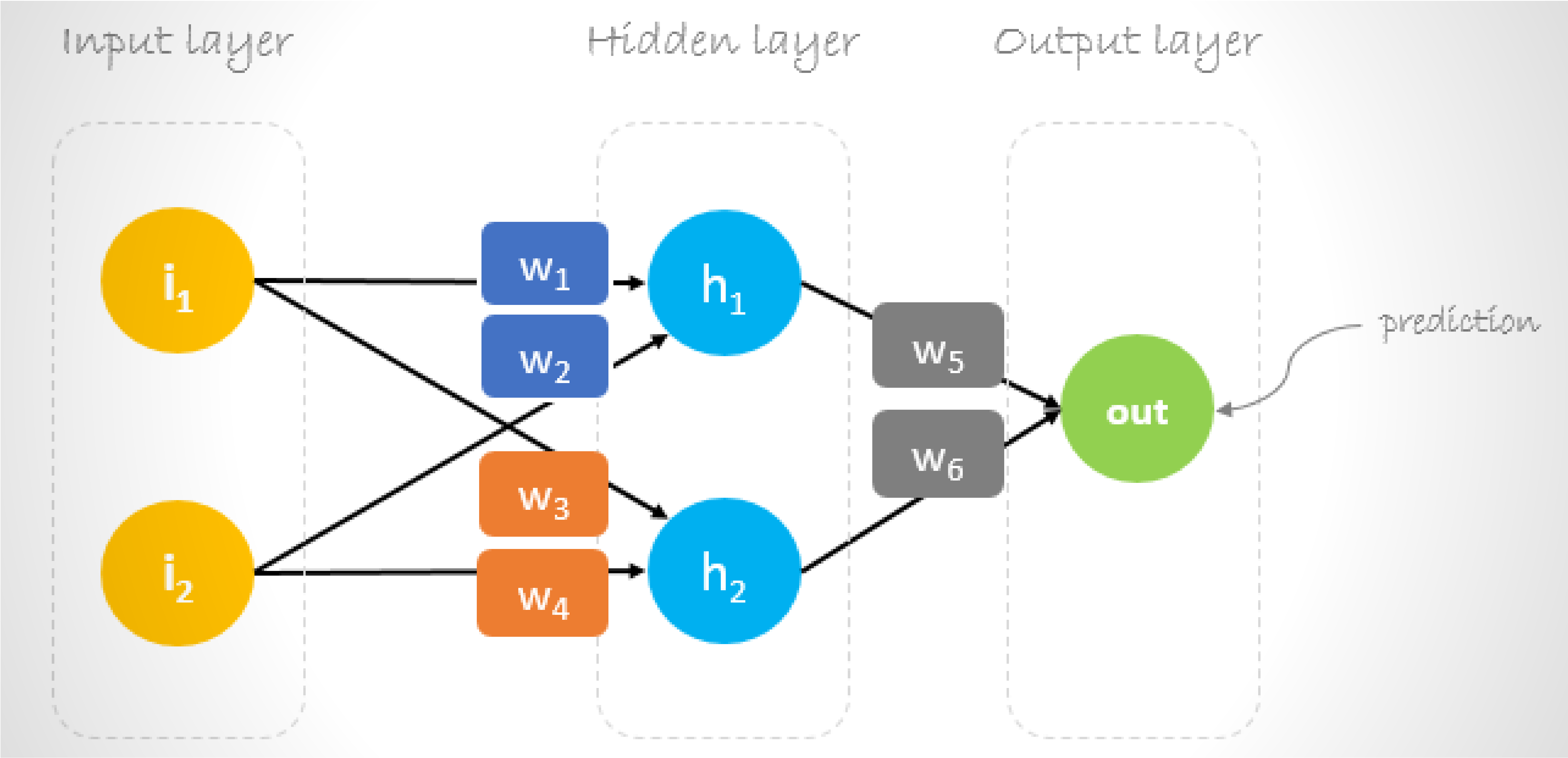
Backpropagation Algorithm

What is Backpropagation?

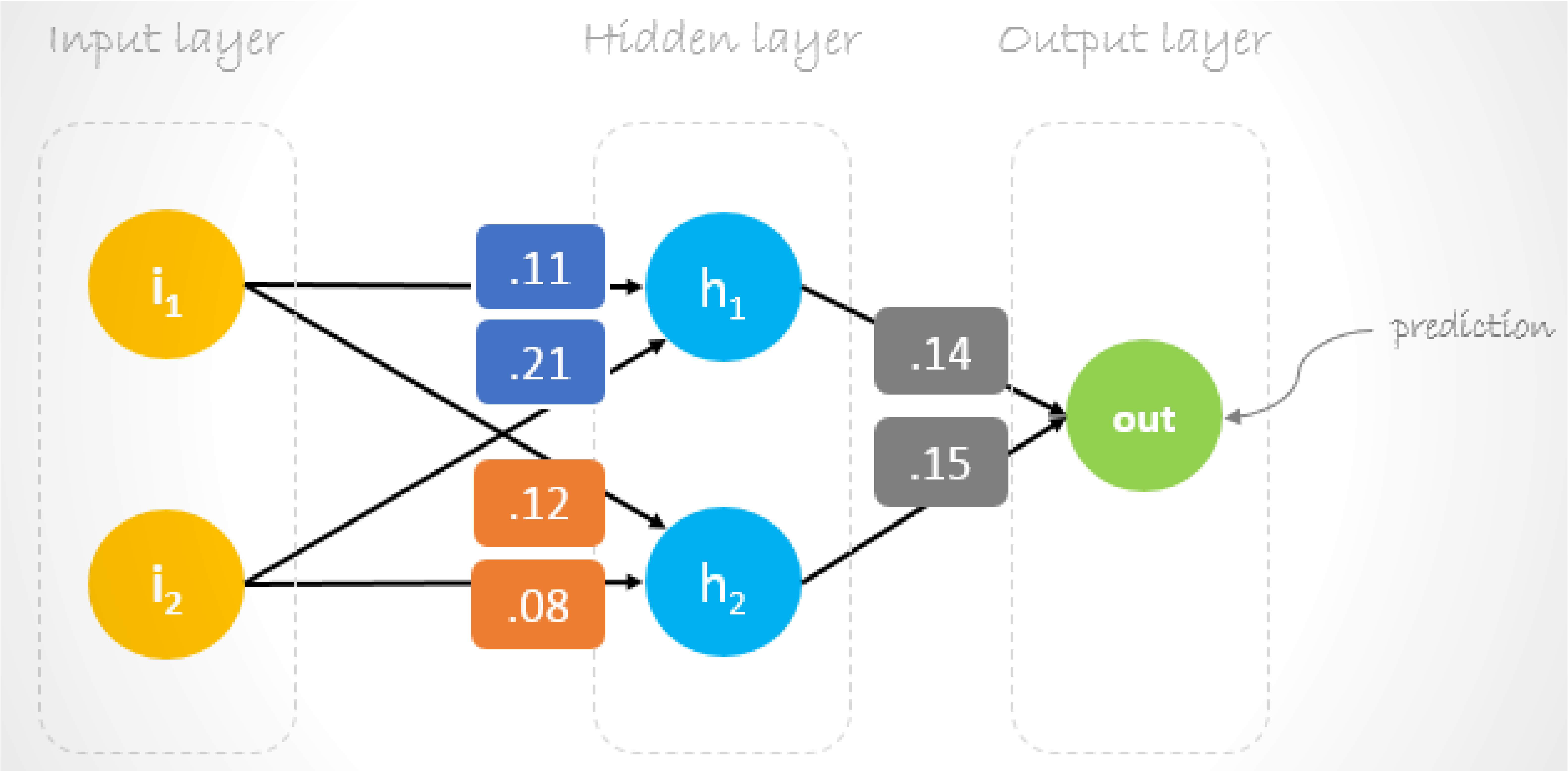
Backpropagation is a supervised learning algorithm, for training Neural Networks.

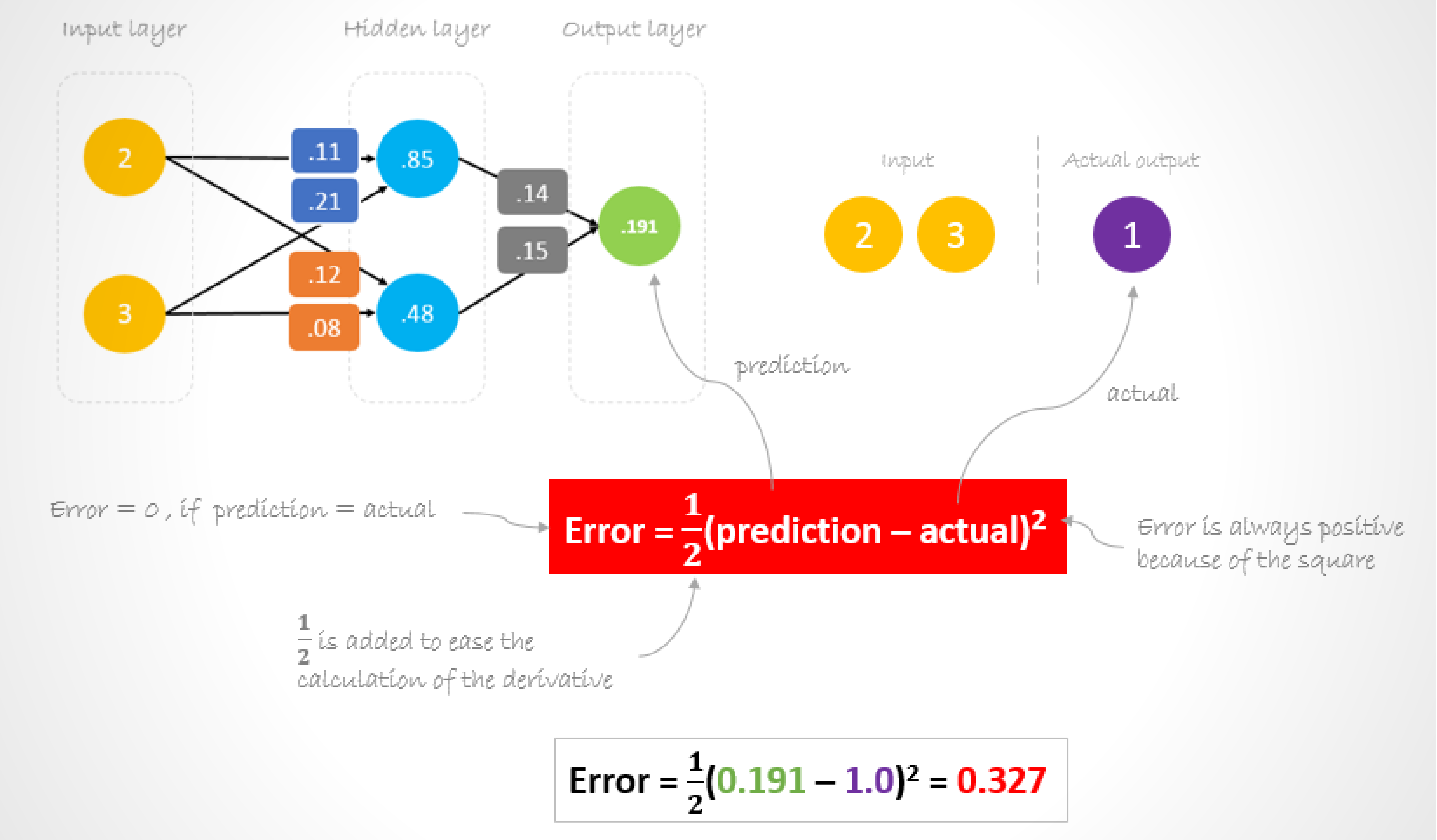


Why We Need Backpropagation?



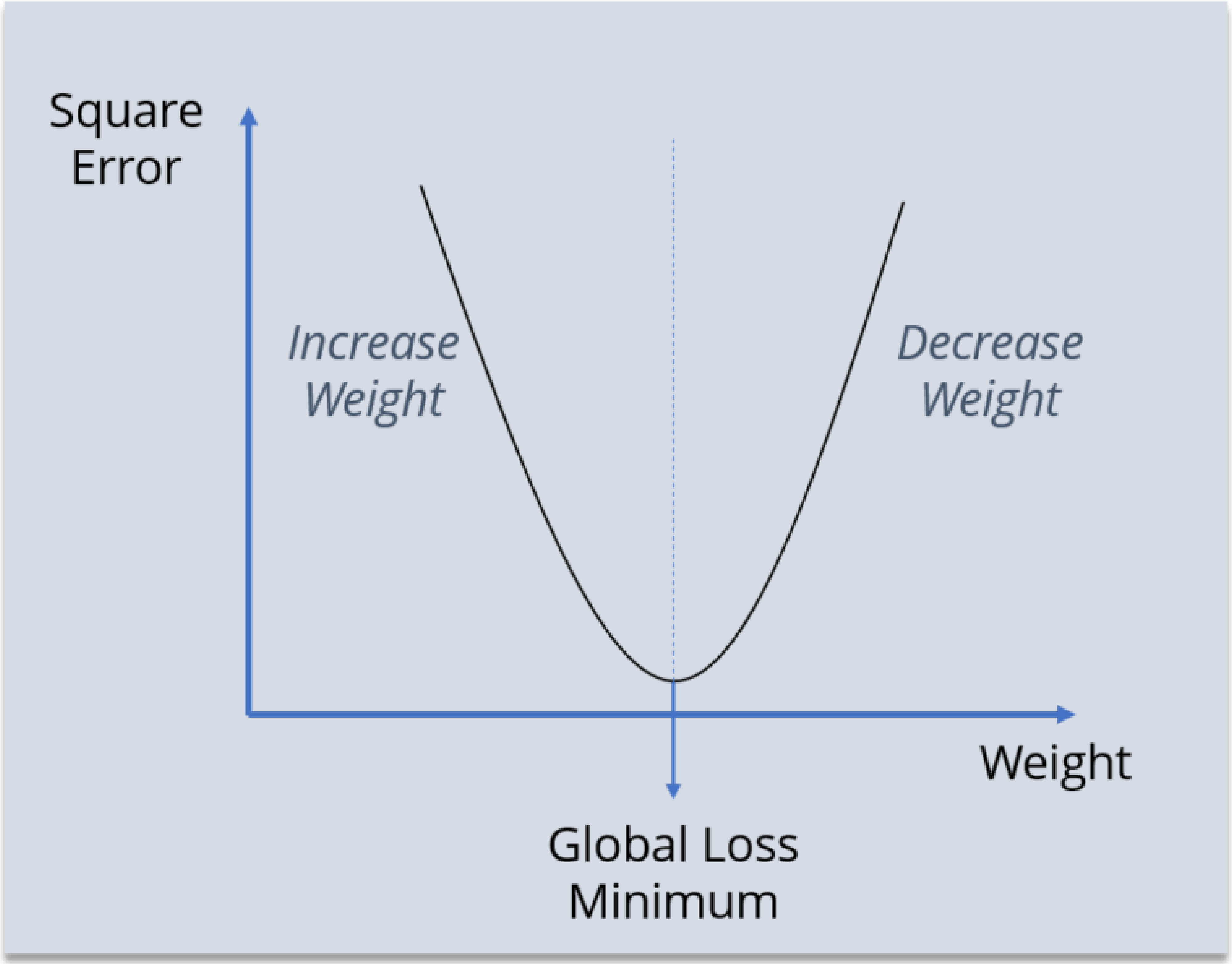
Why We Need Backpropagation?



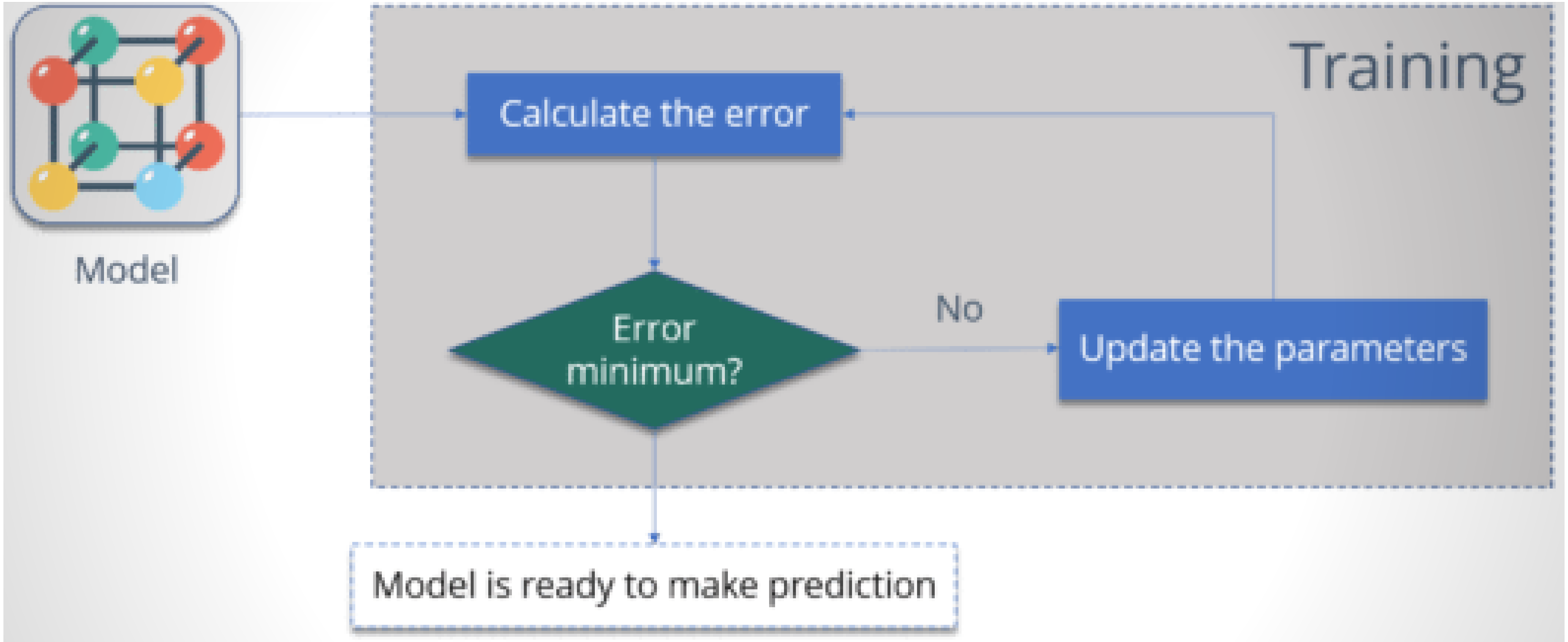

Input layer
Hidden layer
Output layer
2
3
.11
.21
.12
.08
.85
.48
.14
.15
.191
Input
Actual output
2
3
1
prediction
actual
Error = 0 , if prediction = actual
$\text{Error} = \frac{1}{2}(\text{prediction} - \text{actual})^2$
Error is always positive because of the square
$\frac{1}{2}$ is added to ease the calculation of the derivative
$\text{Error} = \frac{1}{2}(0.191 - 1.0)^2 = 0.327$

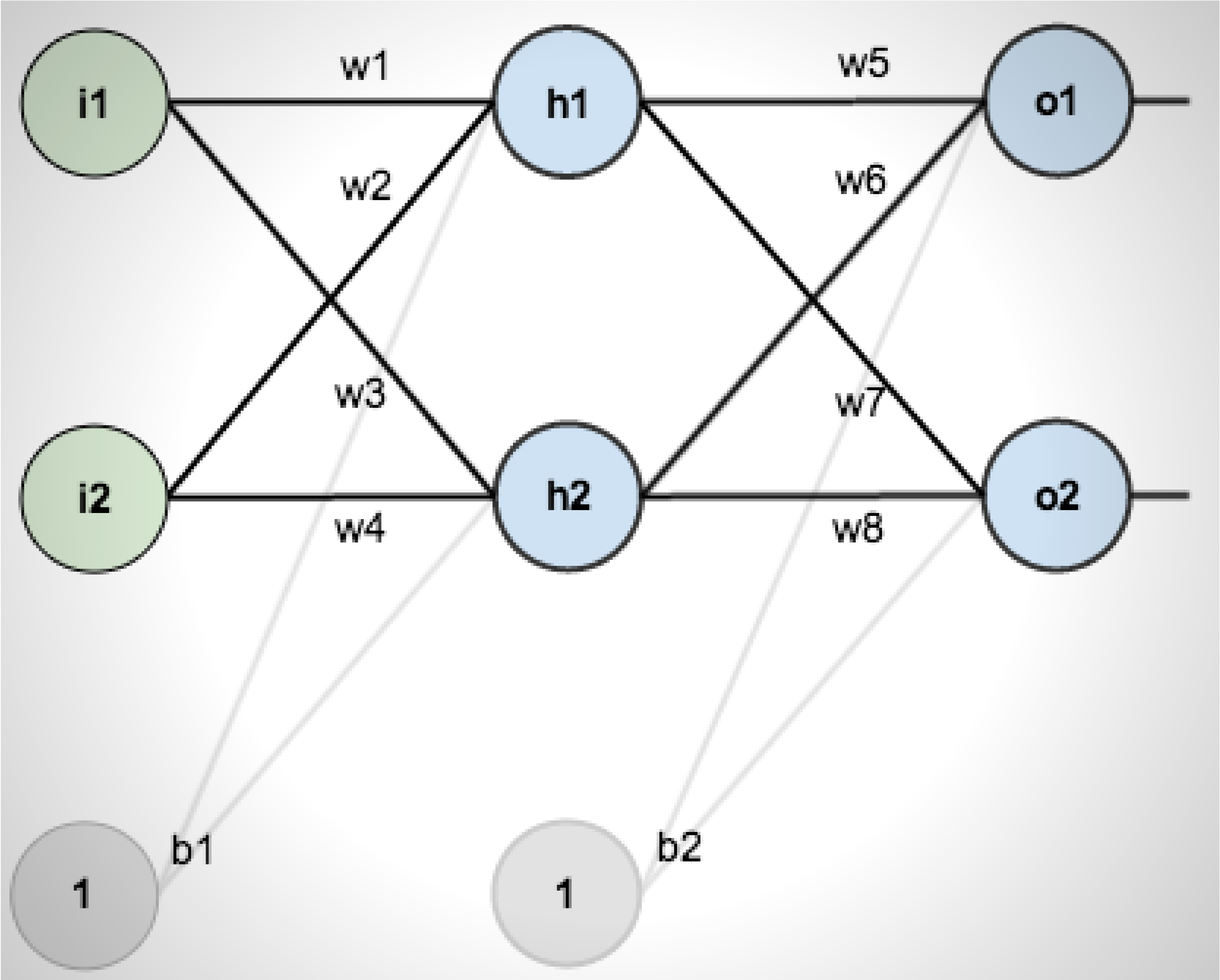
Gradient Descent

- Update the weights using gradient descent.
- Gradient descent is used for finding the minimum of a function.
- In our case we want to minimize the error function.

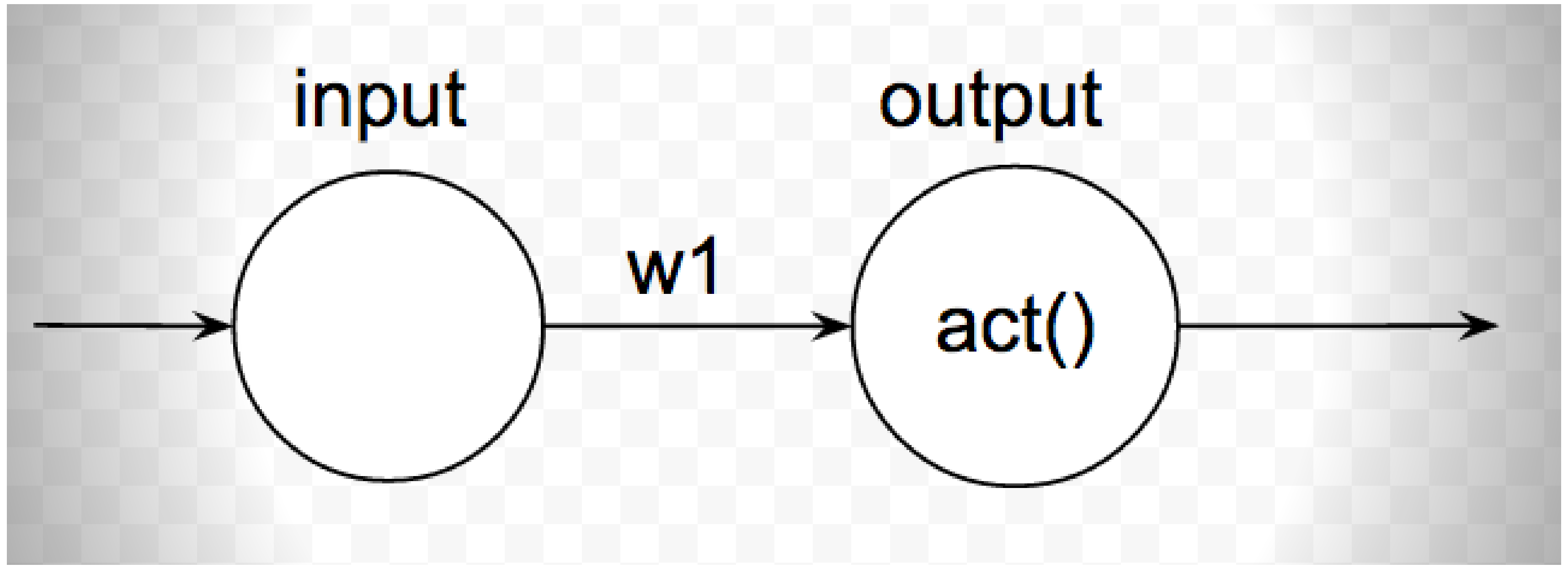


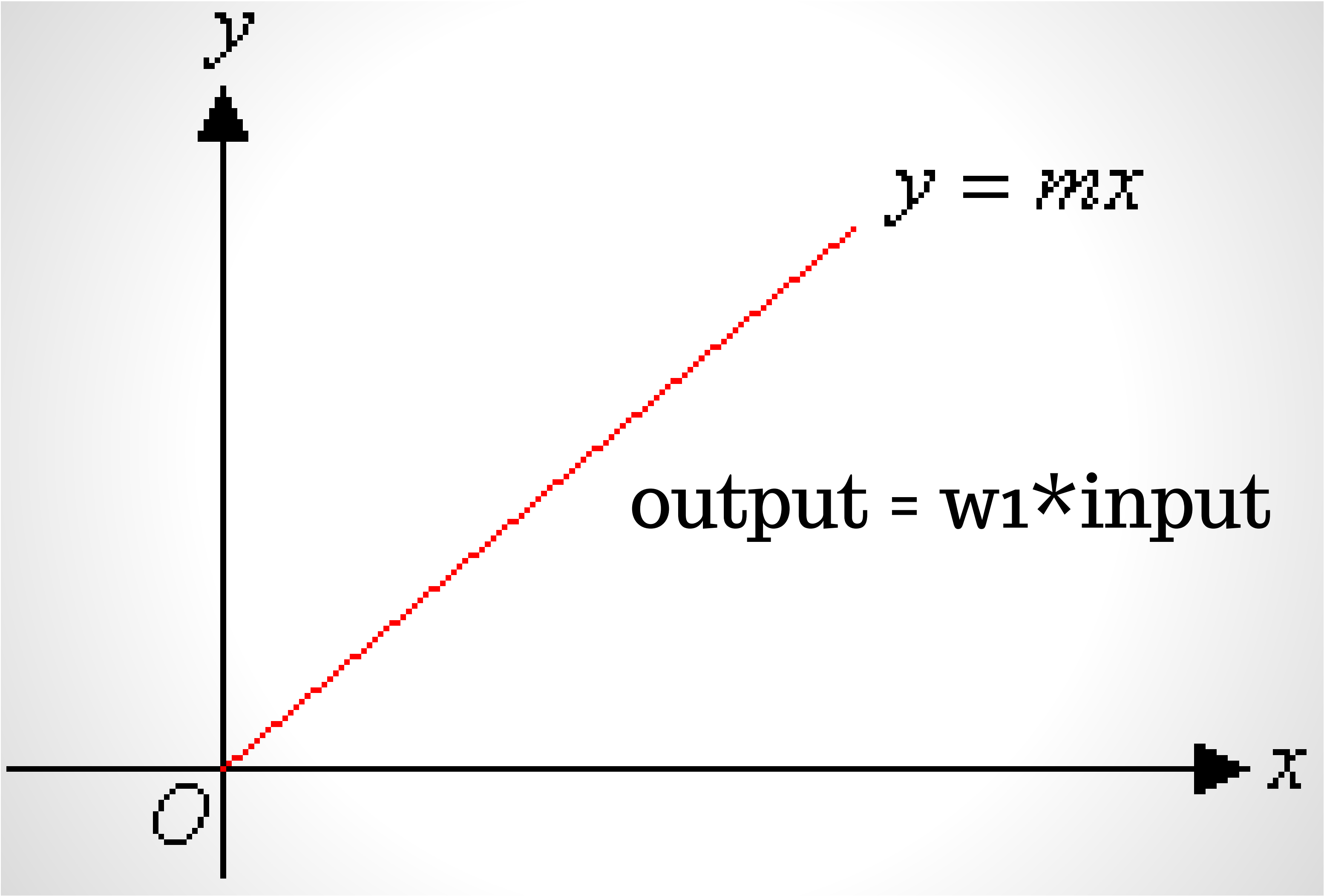
OVERVIEW:

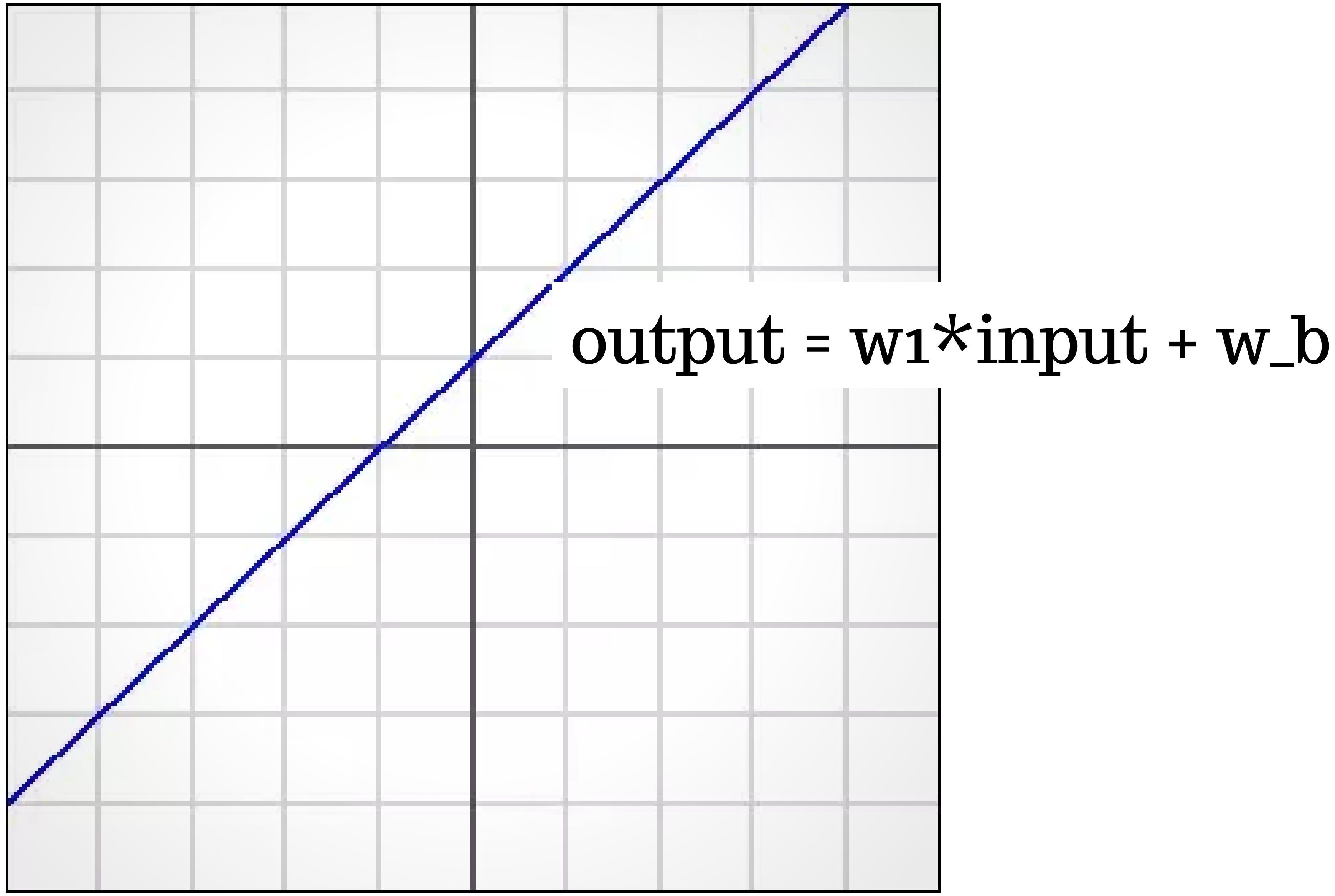

Training
Model
Calculate the error
Error minimum?
No
Update the parameters
Model is ready to make prediction

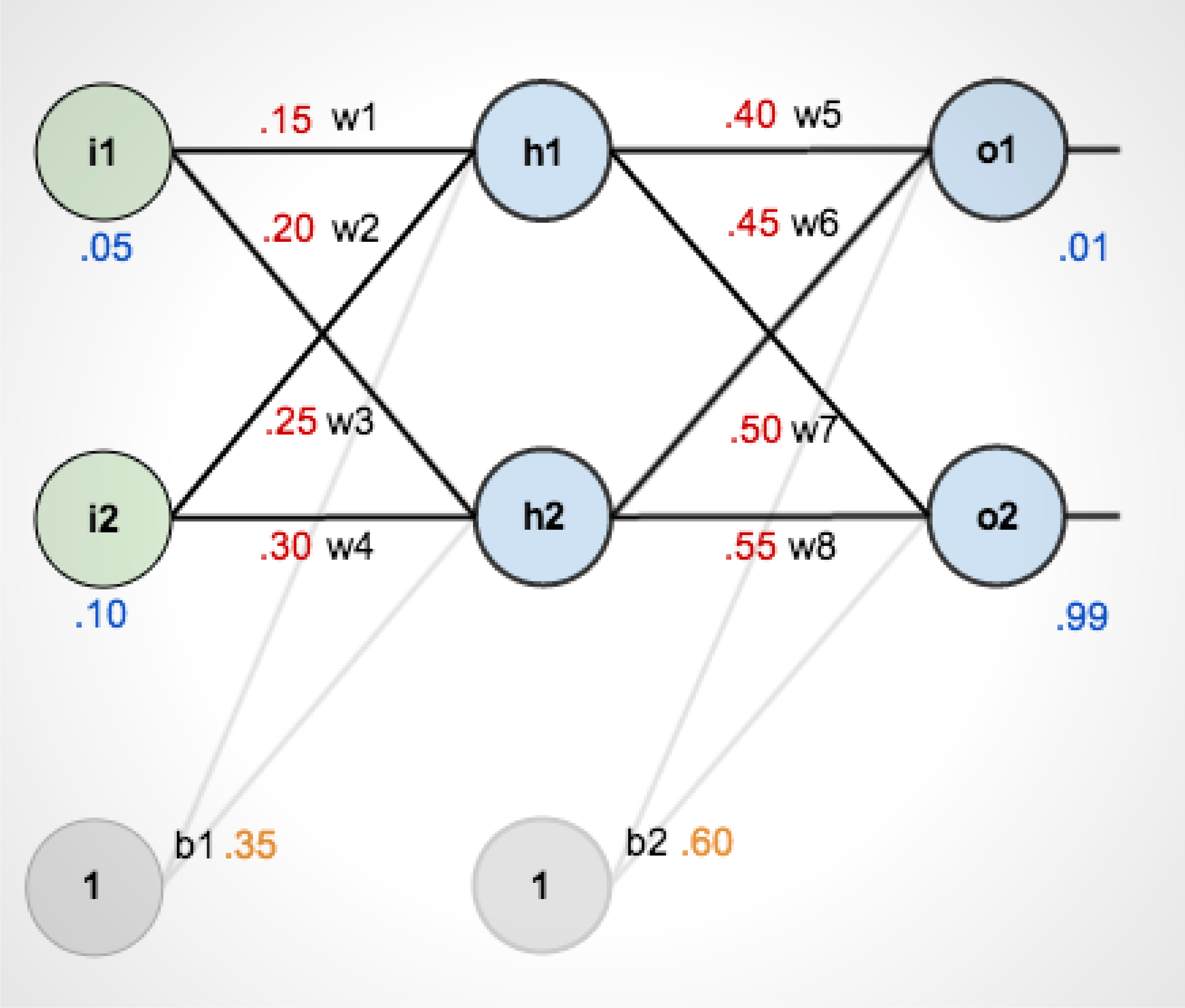

i1
i2
1
w1
w2
w3
w4
b1
h1
h2
1
w5
w6
w7
w8
b2
o1
o2

- y = w1 * x
- Output of bias is **w_b**.
- w_b = 1 * b1









i1=0.05, i2=0.10

w1=0.15, w2=0.20

w3=0.25, w4=0.30

b1=0.3

w5=0.4. w6=0.45

w7=0.5, w8=0.55

b2=0.6

o1=0.01, o2=0.99

Forward Pass

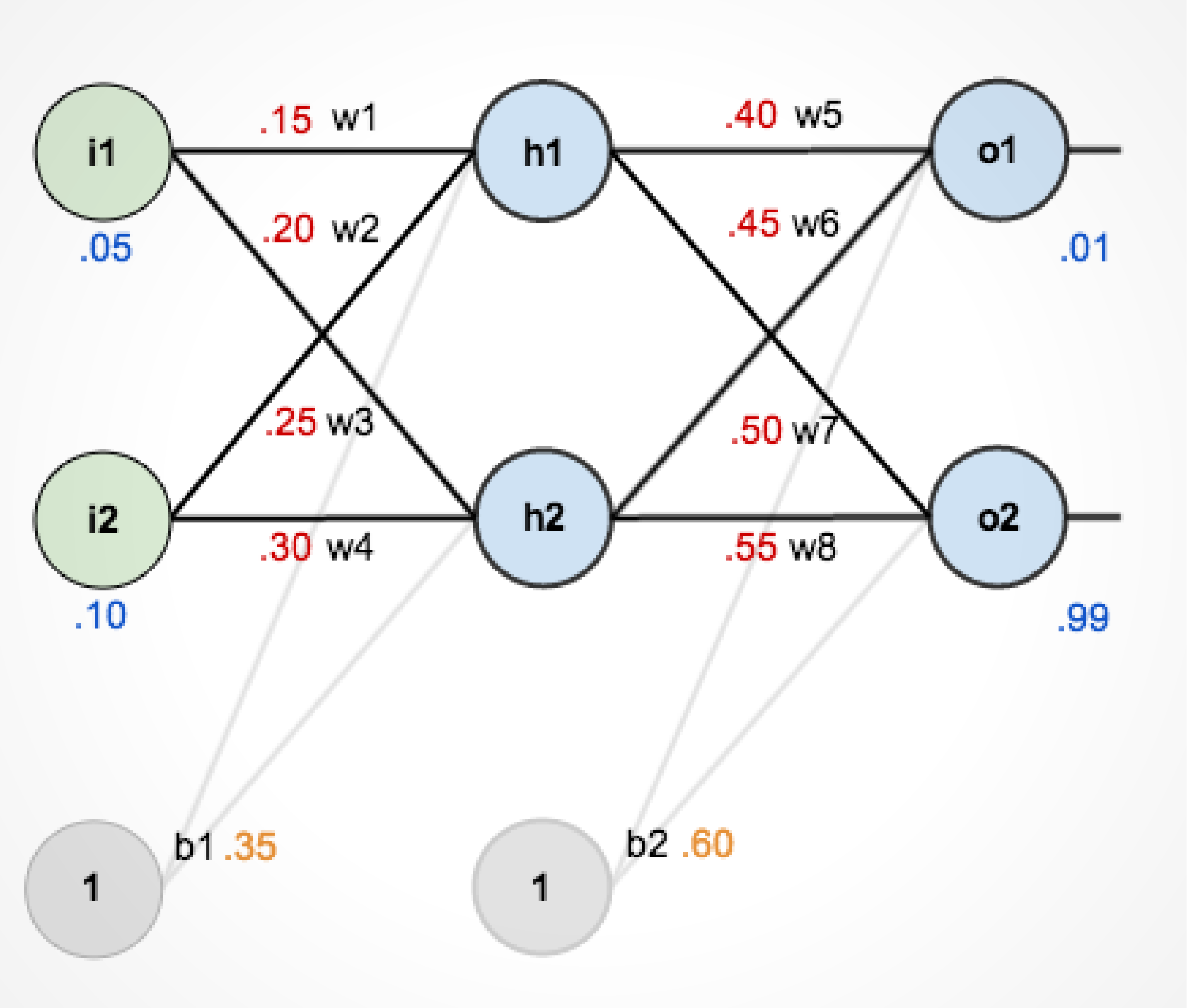
$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

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

Squash it using logistic function to get output

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

$$out_{h2} = 0.596884378$$



Forward Pass

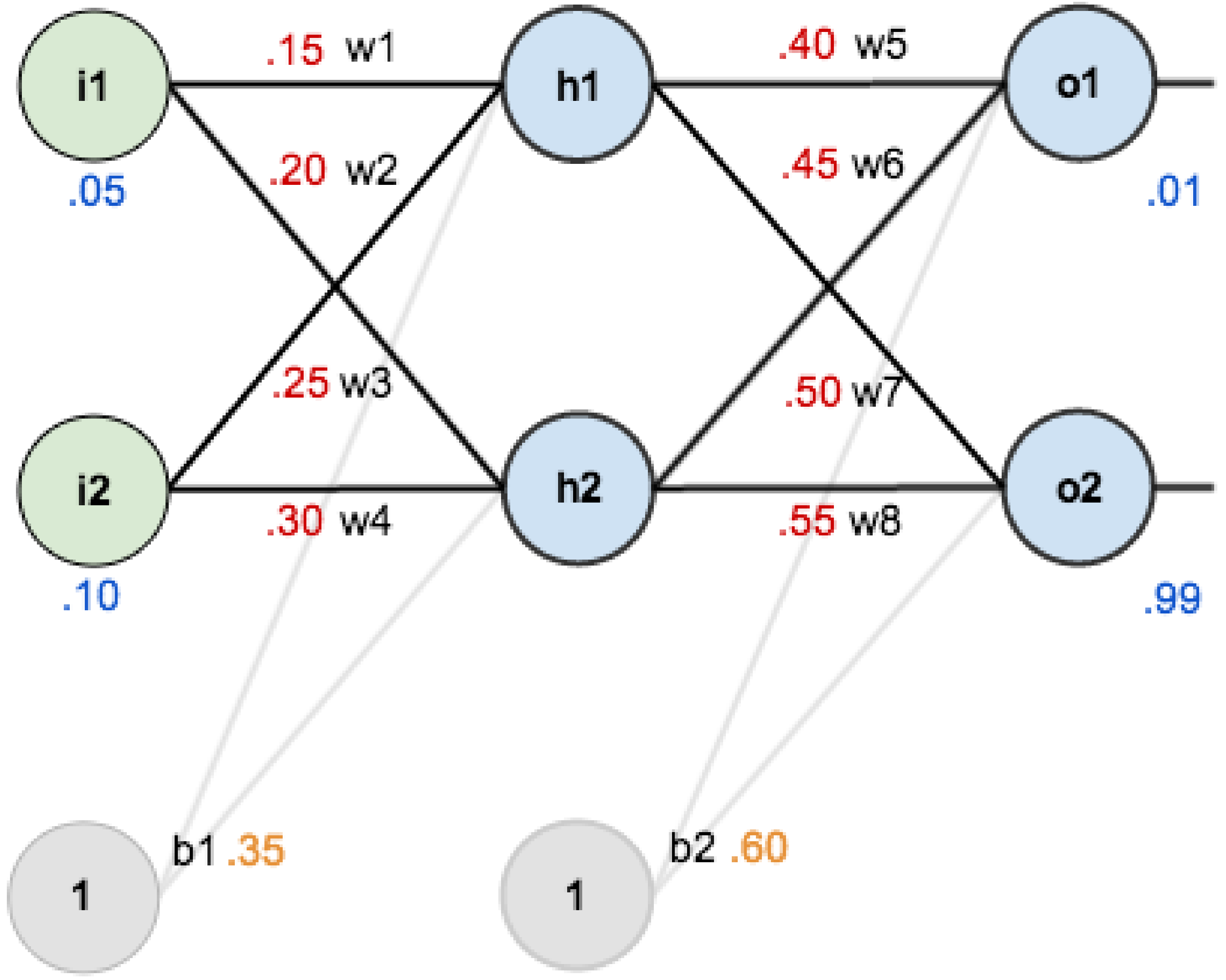
$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

And carrying out the same process for o_2 we get:

$$out_{o2} = 0.772928465$$



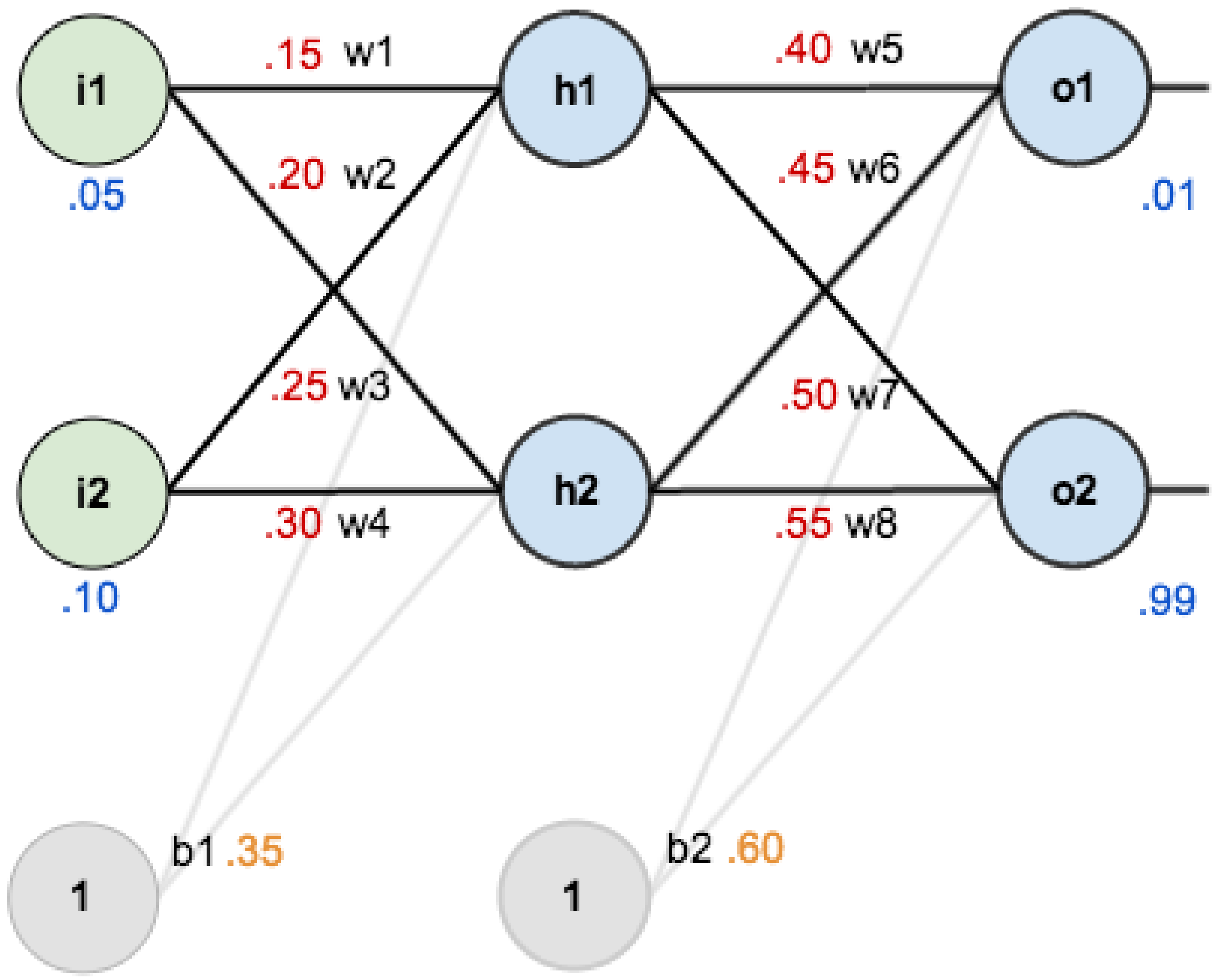
Calculating Total Error

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

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

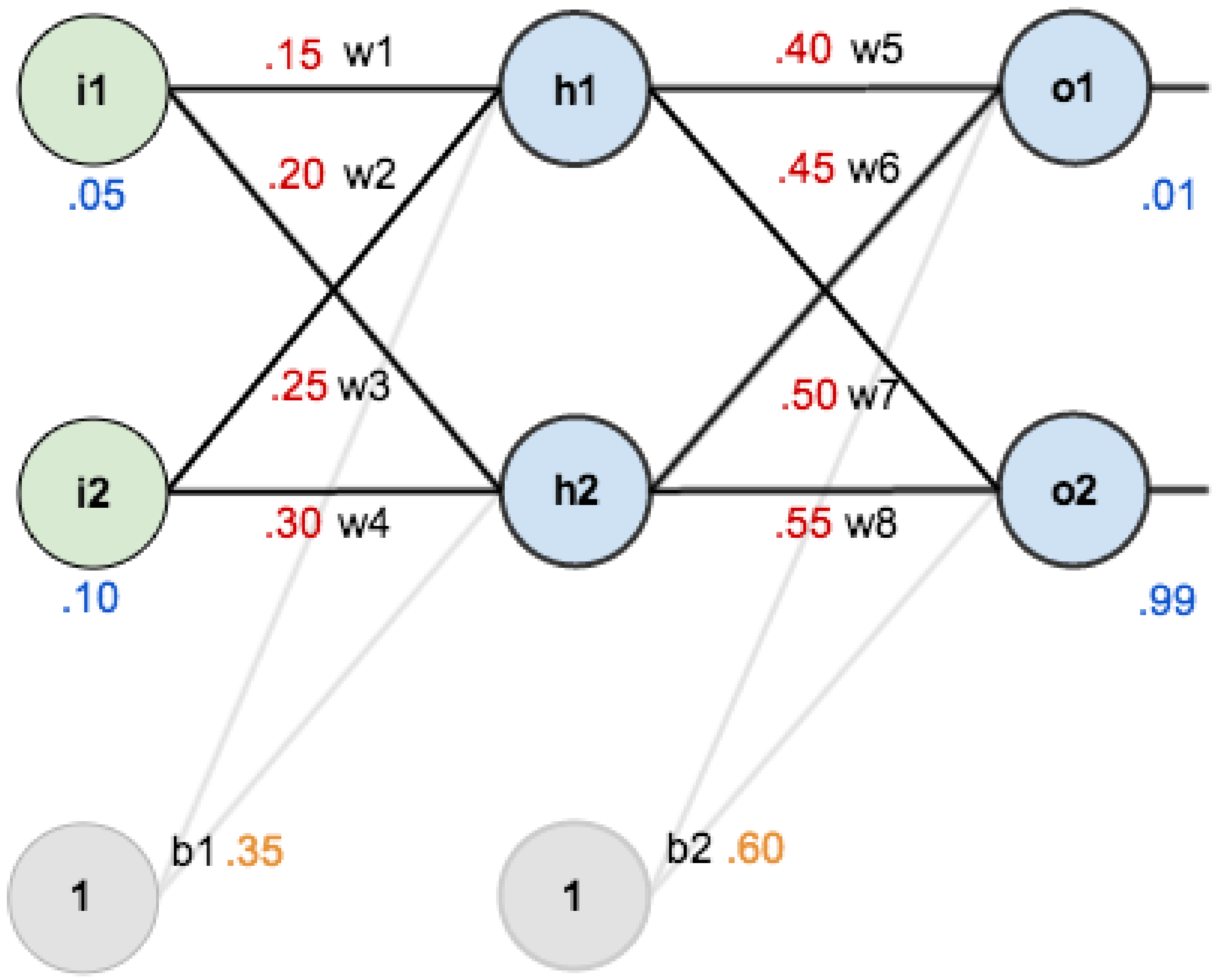
$$E_{o2} = 0.023560026$$

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



The Backwards Pass

- Update the weights using gradient descent.
- 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:

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

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

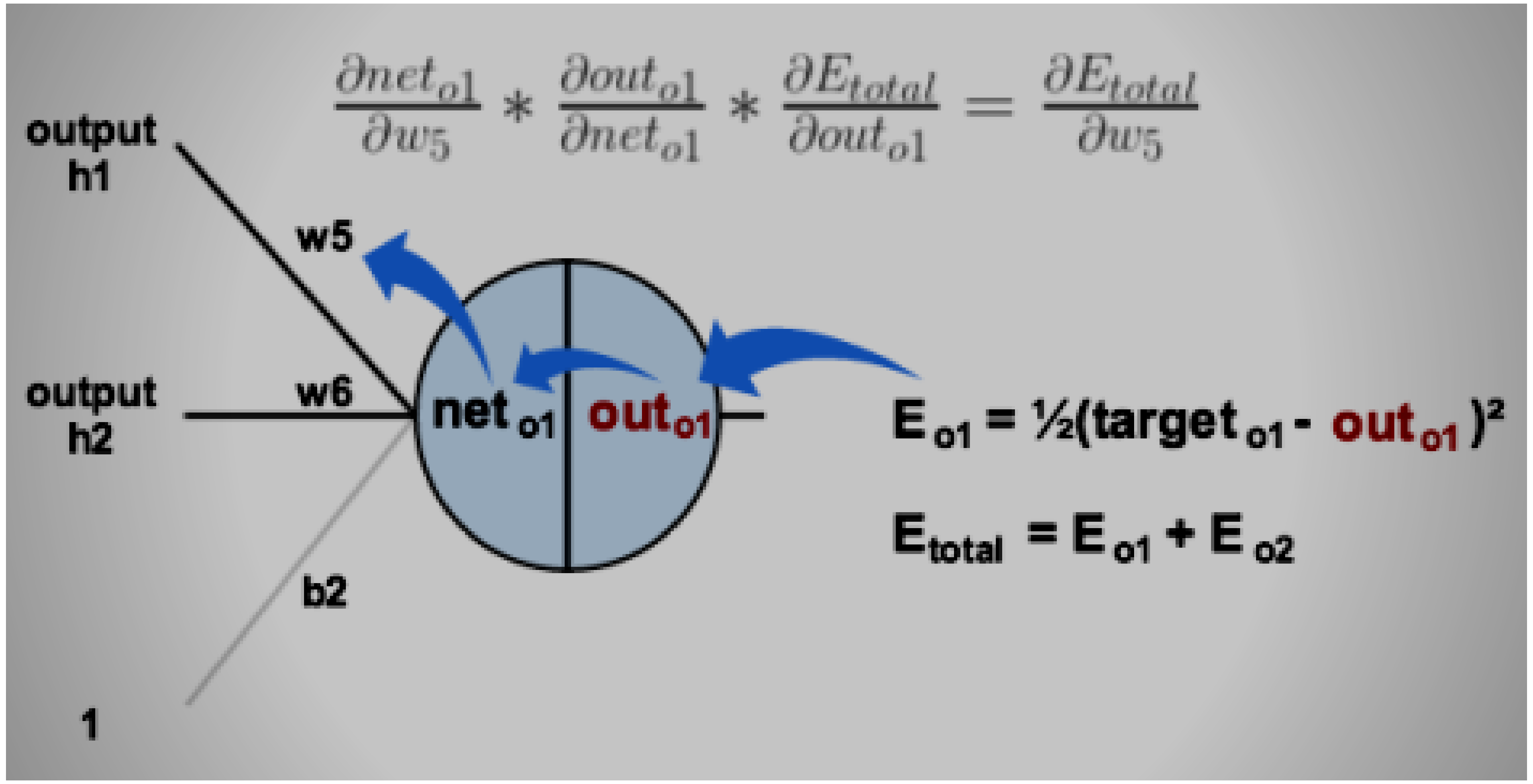
Consider w5.

$\frac{\partial E_{total}}{\partial w_5}$ partial derivative of E_{total} wrt w5.

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

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

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



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

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

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

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

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

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

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} \qquad \frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1})$$

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

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

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

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

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

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

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

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.08216704$$

- Update the weight.
- To decrease the error, we then subtract this value from the current weight.

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

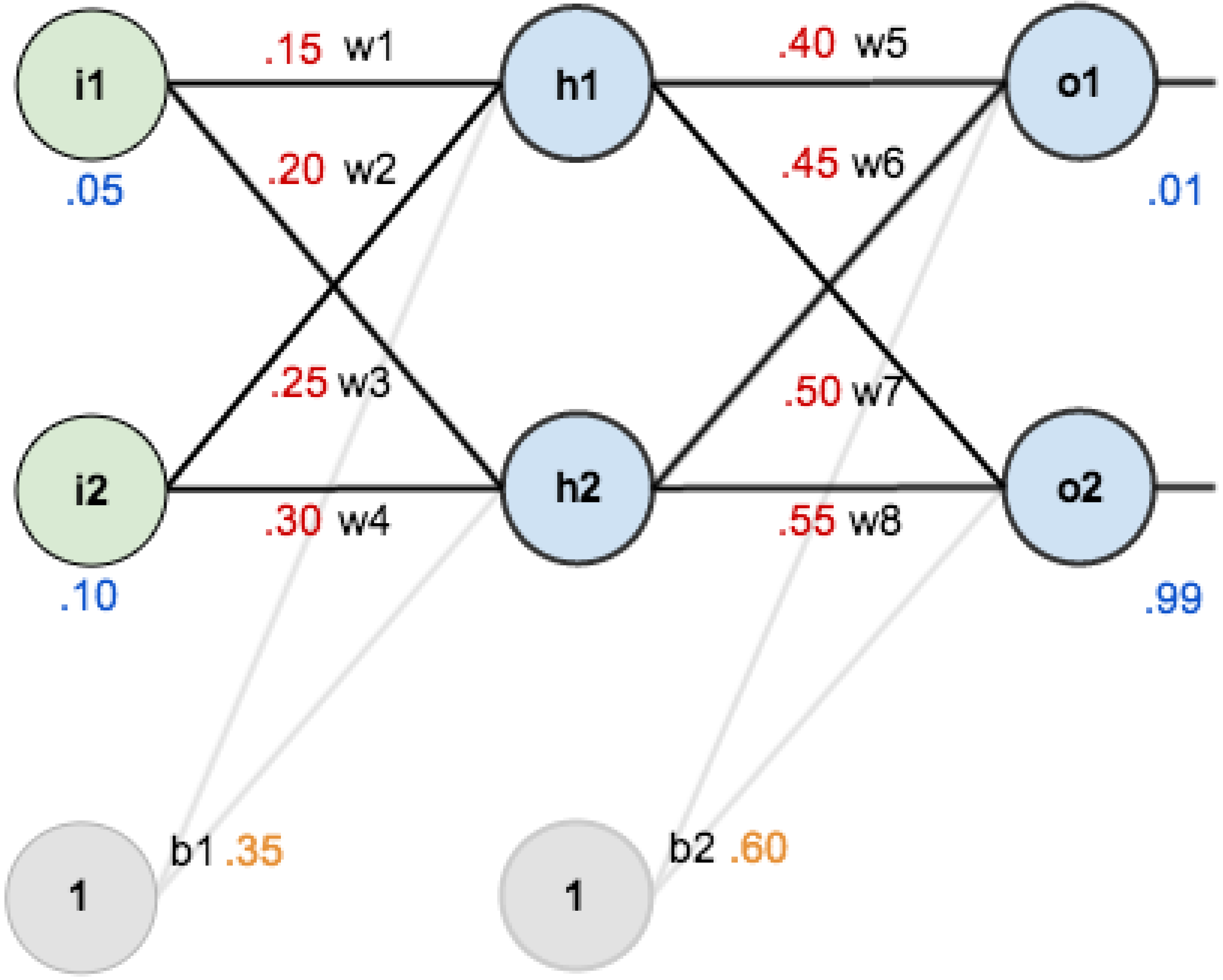
η = learning rate (eta)

$$w_6^+ = 0.408666186$$

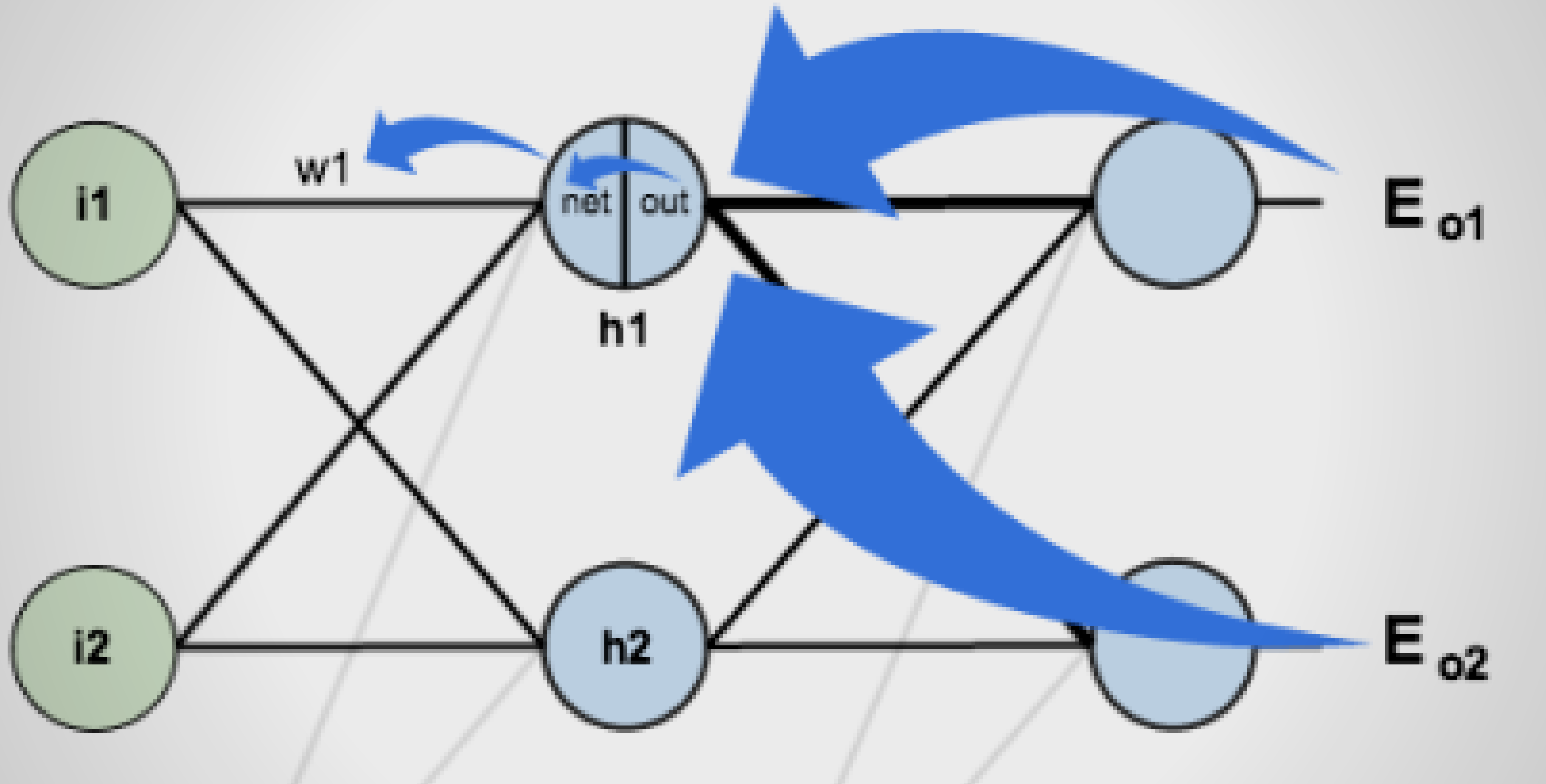
$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

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 below).



Hidden Layer:



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

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

$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2$ $out_{o1} = \frac{1}{1+e^{-net_{o1}}}$ $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

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

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

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

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

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

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

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

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

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

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

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

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

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

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 backpropagation, 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.

Thank You