Neural Networks

Stephen G. Ware

CSCI 4525 / 5525

Narrative Intelligence LAB

THE UNIVERSITY of NEW ORLEANS

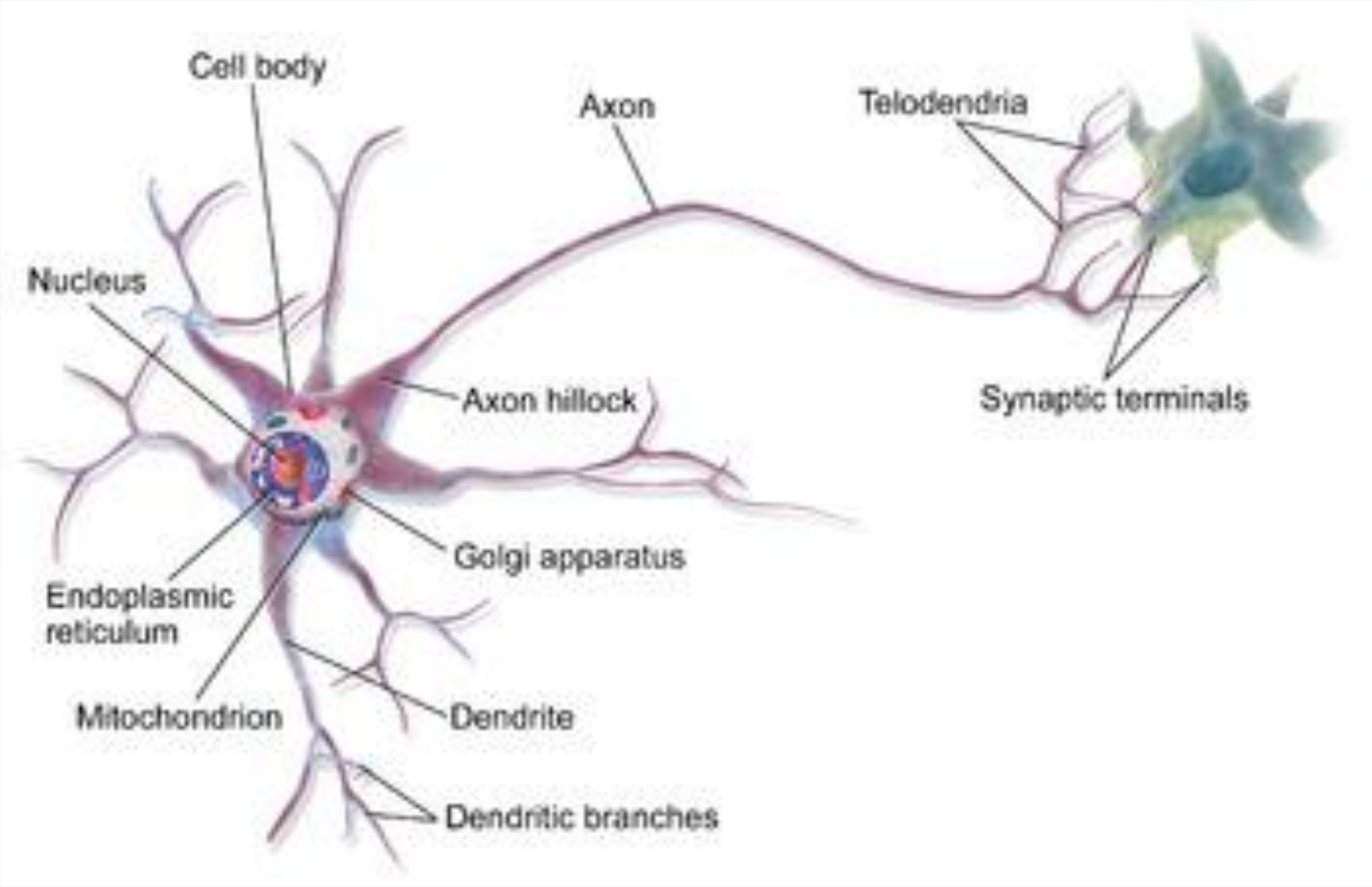
Neural Networks

A **neural network** is a complex system of simple, interconnected units inspired by animal neurons.

Each neuron has a simple rule which defines its value. It communicates this value to other neurons via weighted directed edges, and these values affect the nodes to which they are communicated.

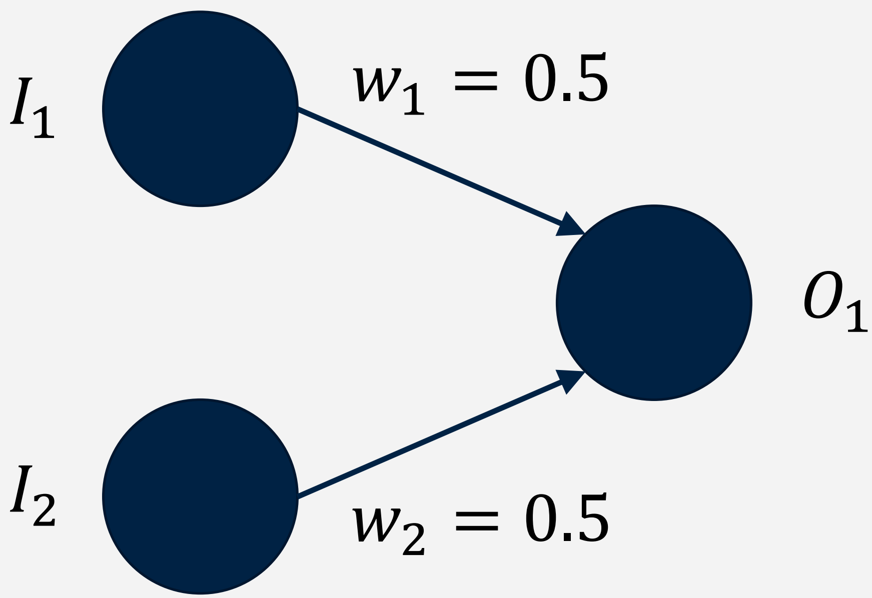
Neural nets are capable of approximating complex functions.

Animal Nervous Systems



NIL

Simple Example 1

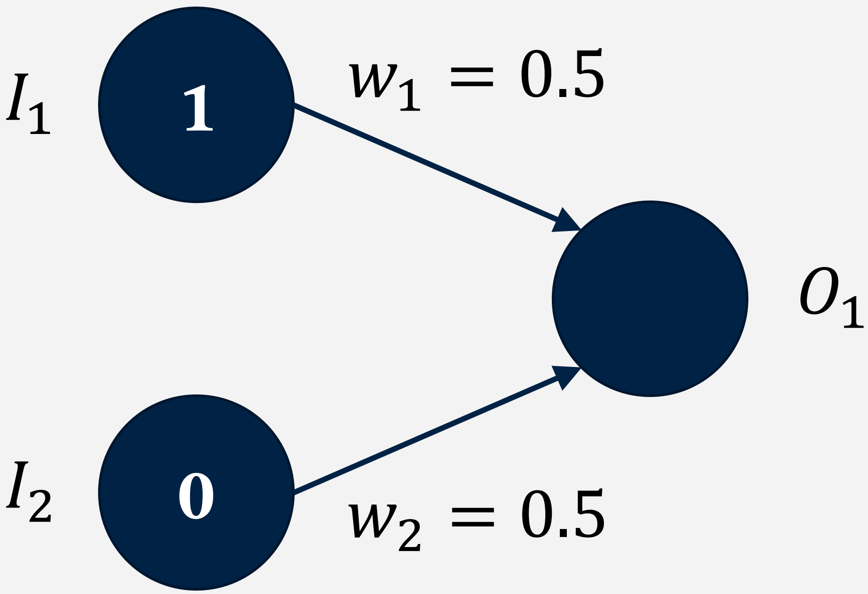


Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = I_1 w_1 + I_2 w_2$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Simple Example 1

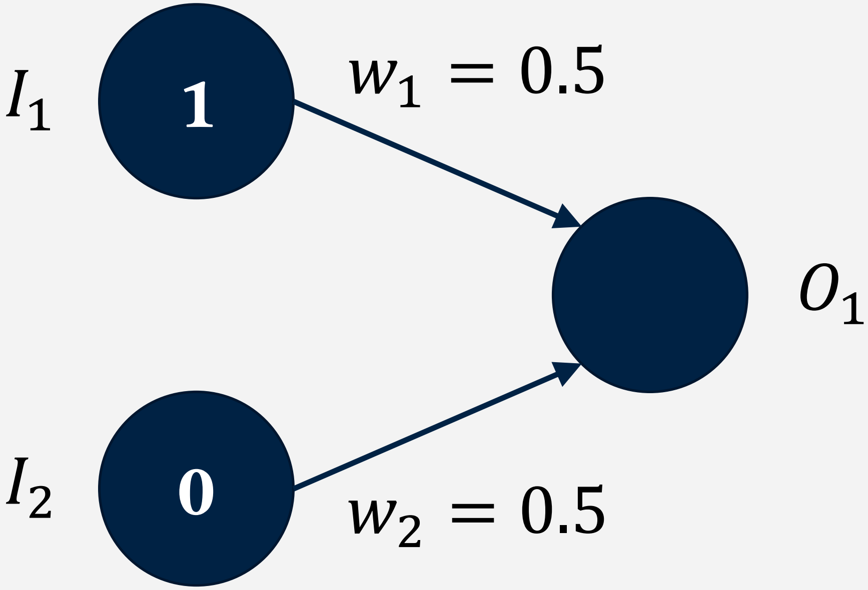


Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = I_1 w_1 + I_2 w_2$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Simple Example 1

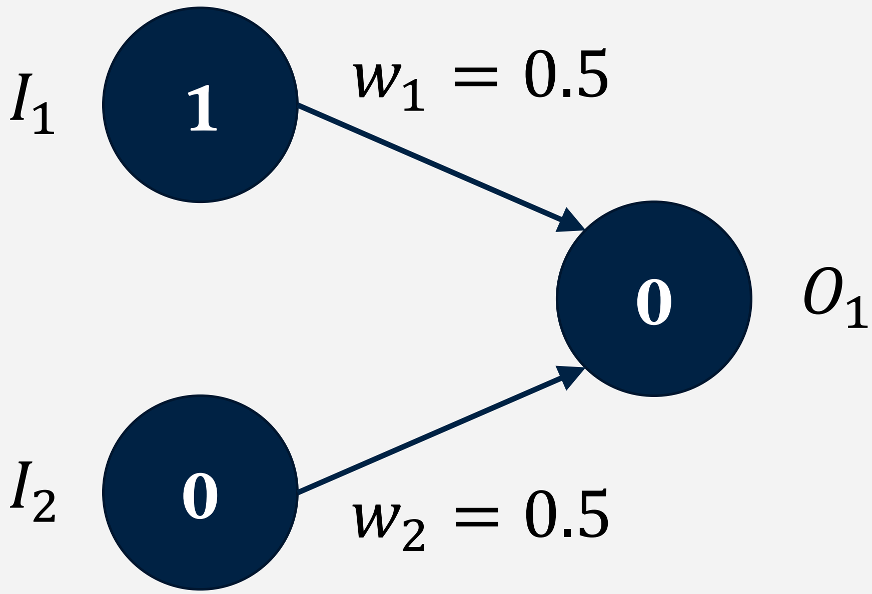


Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = 1 \cdot 0.5 + 0 \cdot 0.5$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Simple Example 1

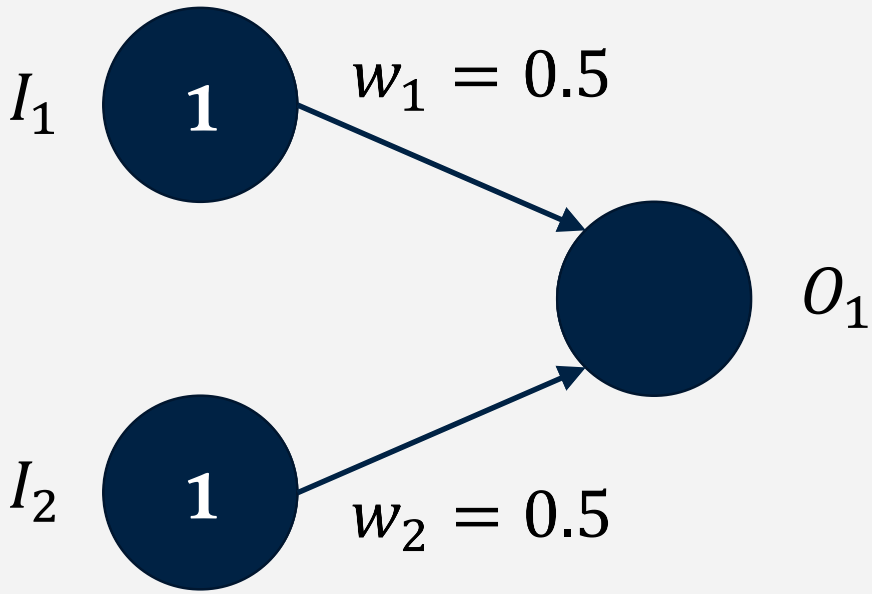


Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = 1 \cdot 0.5 + 0 \cdot 0.5$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Simple Example 1

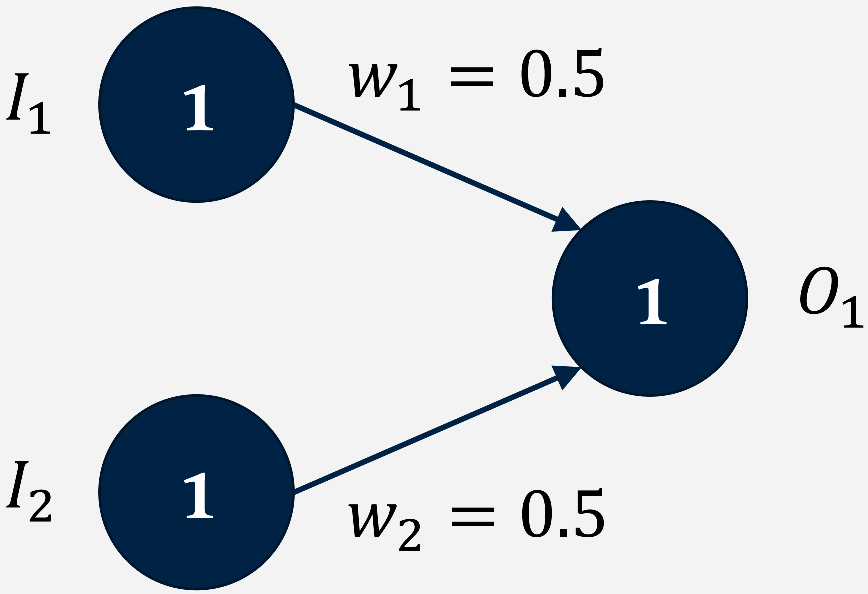


Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = 1 \cdot 0.5 + 1 \cdot 0.5$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Simple Example 1

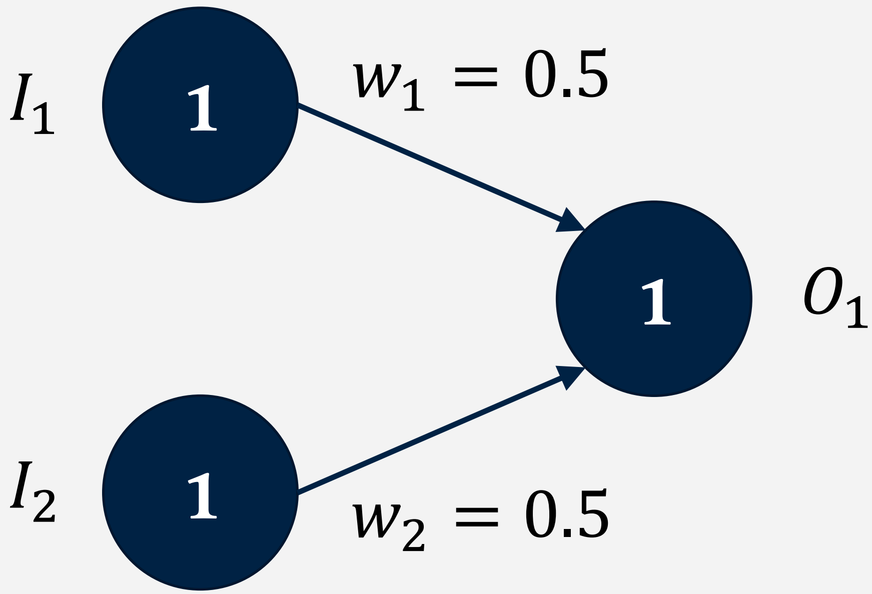


Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = 1 \cdot 0.5 + 1 \cdot 0.5$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Simple Example 1



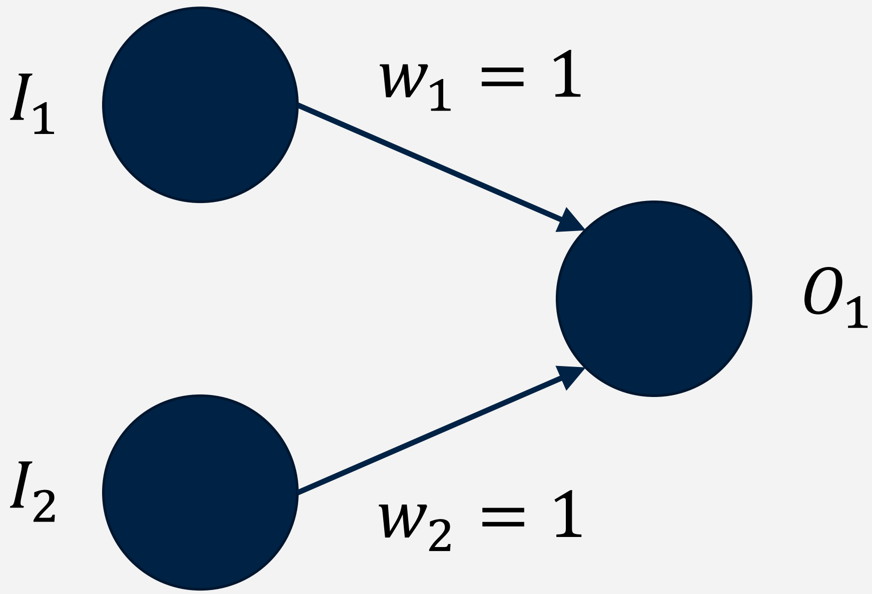
Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = 1 \cdot 0.5 + 1 \cdot 0.5$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Boolean AND

Simple Example 2

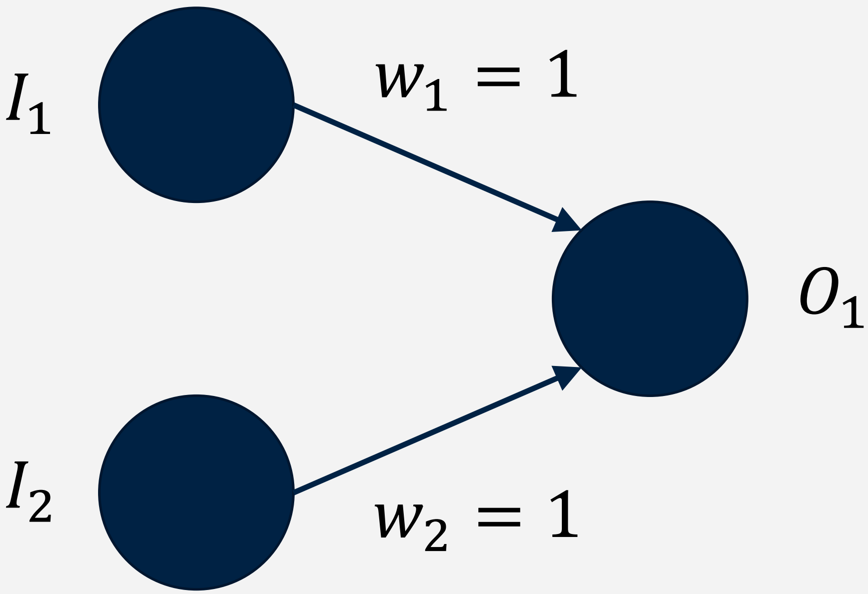


Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = I_1 w_1 + I_2 w_2$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Simple Example 2



Input nodes I_1 and I_2 take 0 or 1 as input.

$$O_1 = I_1 w_1 + I_2 w_2$$

Output node O_1 outputs 1 if its value is ≥ 1, 0 otherwise.

Boolean OR

Structure of a Neural Net

- 1 or more input nodes in the input layer.
- 1 or more output nodes in the output layer.
- Directed, weighted edges between nodes.
- Nodes have an activation function the defines its output given the sum of its inputs.
- Optionally, the network has 1 or more hidden nodes, possibly organized into hidden layers.

Types of Neural Nets

A neural net with no hidden layers or cycles and a threshold activation function is called a **perceptron**. Perceptrons are one popular kind of linear classifier.

A neural net with no cycles is called a **feed-forward** network. These are simple and probably the most common kind of neural nets.

A neural net which contains cycles is called **recurrent**, and can model a kind of short term memory.

Neural Nets in AI History

- First work on neural nets (perceptrons) started 1957 and was a source of much excitement.
- In 1969, Minsky and Papert published *Perceptrons*, which demonstrated that many kinds of functions could not be learned by perceptrons.
- Excitement for neural nets decreased dramatically for more than a decade. Some consider this an inciting incident for AI Winter.
- Neural nets experiencing a resurgence since 80's.

Value of Neural Nets

- Input and output can be thought of as vectors.
    - This is ideal for applications like computer vision, where the input might be a camera's pixels and the output might be a vector of all the possible things the network is trained to recognize.
- Neural nets with hidden layers are an excellent tool for performing non-linear regression.

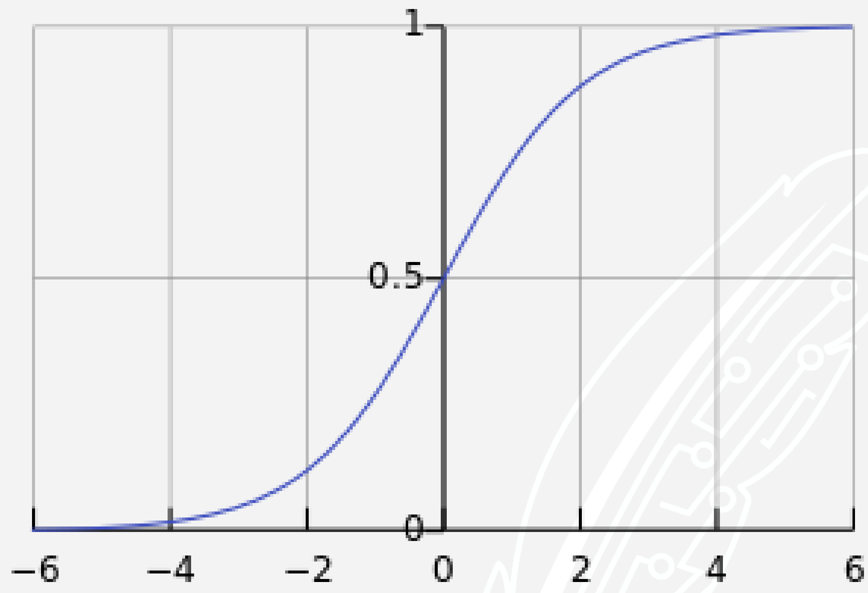
Example Neural Net

We will consider a simple feed-forward example network with:

- 2 input nodes
- 1 output node
- 1 hidden layer with 2 nodes
- Sigmoid activation function

Sigmoid Activation Function

Maps any input number x to a number between 0 and 1.



$f(-4) \approx 0.02$
$f(0) \approx 0.50$
$f(2) \approx 0.88$

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

Training a Neural Net

Like other machine learning tools, neural nets can be trained on a set of examples.

Training occurs by adjusting the weights of edges.

Back Propagation Learning

```
Set all edge weights to random values.
Calculate the network's output.
Calculate the error of this output.
Until the error cannot be reduced further:
    For each layer, from output back to input:
        Adjust the weights of the edges going to the nodes
        in this layer to reduce the error in the last layer.
```

Updates Based on Error

Given some function, the **gradient** of that function is a vector that points in the direction of the greatest rate of increase, and its magnitude is the slope of the graph in that direction.

Practically speaking, we find the gradient of $f(x)$ by differentiating with respect to x.

Updating Based on Error

In general, error is the difference between what you want and what you actually got:

$$correct - output$$

When we update the weights in a neural network, we want to update them to reduce error. We want to raise or lower them appropriately (direction) and we want to raise or lower them as much as is needed (magnitude). Hence, we need the gradient of the activation function.

Sigmoid Gradient

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

$$f'(x) = \frac{\partial}{\partial x}\left(\frac{1}{1 + e^{-x}}\right)$$

$$f'(x) = f(x)\big(1 - f(x)\big)$$

Sigmoid Error

When adjusting the weight of an edge in a network whose nodes use the sigmoid activation function, we need to calculate error like so:

$$\underbrace{output(1 - output)}_{\text{error gradient}}\underbrace{(correct - output)}_{\text{error}}$$

Error Values

For each node X, we need to calculate its error value Δ by considering all its outgoing edges and the error values of the nodes in the next level:

$$\Delta(X) = X(1 - X) \sum_{Y} w_{X \to Y} \Delta(Y)$$

The idea here is that each edge is responsible for some of the error of the nodes that take its input.

Edge Weight Updates

- Consider some training example.
- Calculate the output of the network for the example.
- Calculate the error of the network.
- The output nodes are responsible for the error, so adjust the weights going into them.
- The node in the last hidden layer are responsible for some of the error of the edges going into the output, so adjust the weights of the edges going into them.
- So on back through the whole network.

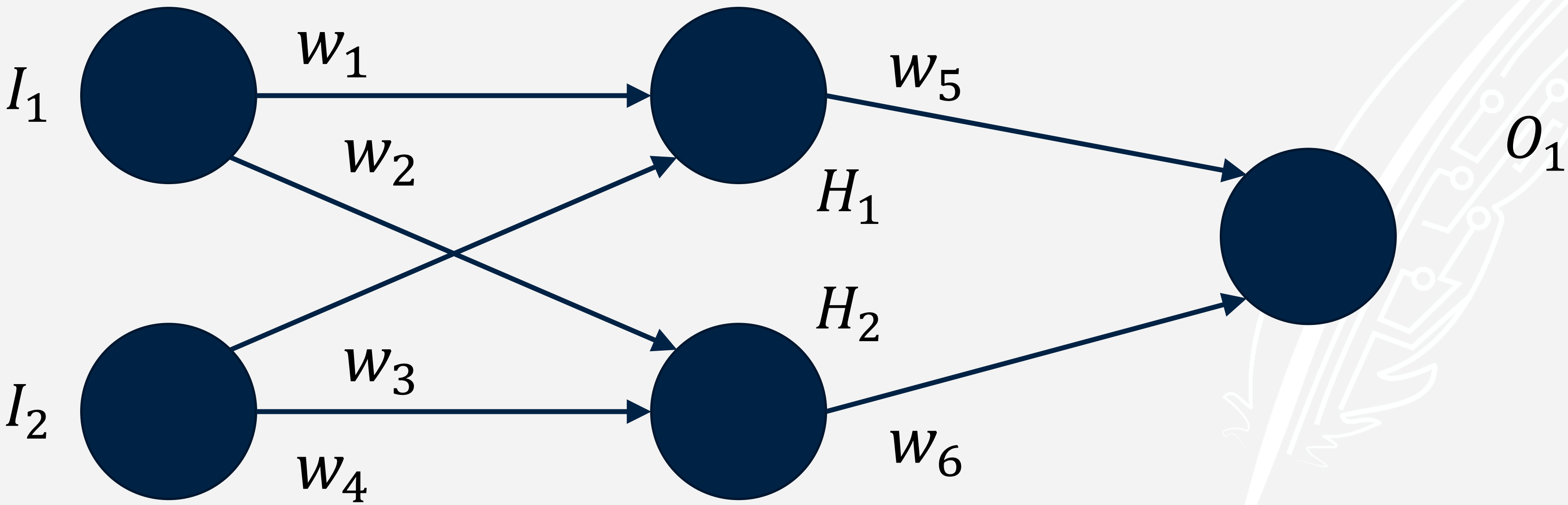
Edge Weight Updates

When updating an edge from node X to node Y with weight $w_{X \to Y}$, we assume that the edge is partially responsible for some of the error in Y. We need to adjust it based on that error and based on the weight of node X:

$$w_{X \to Y} = w_{X \to Y} + \left(X \cdot \Delta(Y)\right)$$

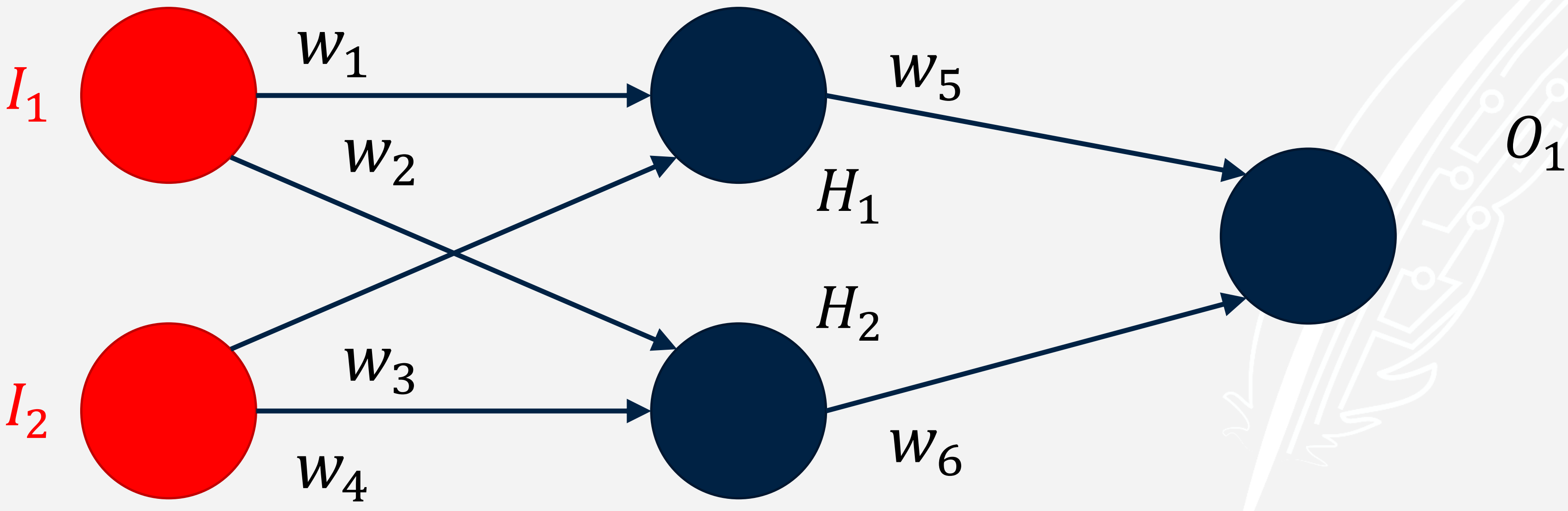
Backpropagation Example

This network has 5 nodes and 3 layers.



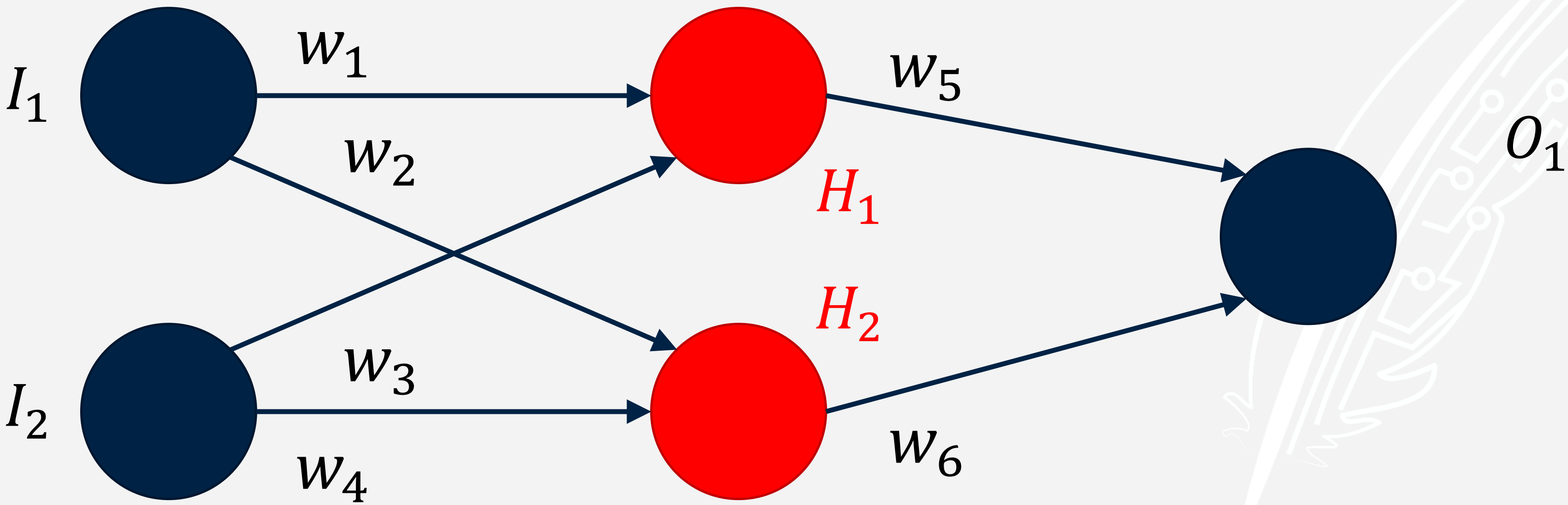
Backpropagation Example

This network has 2 input nodes, I_1 and I_2.



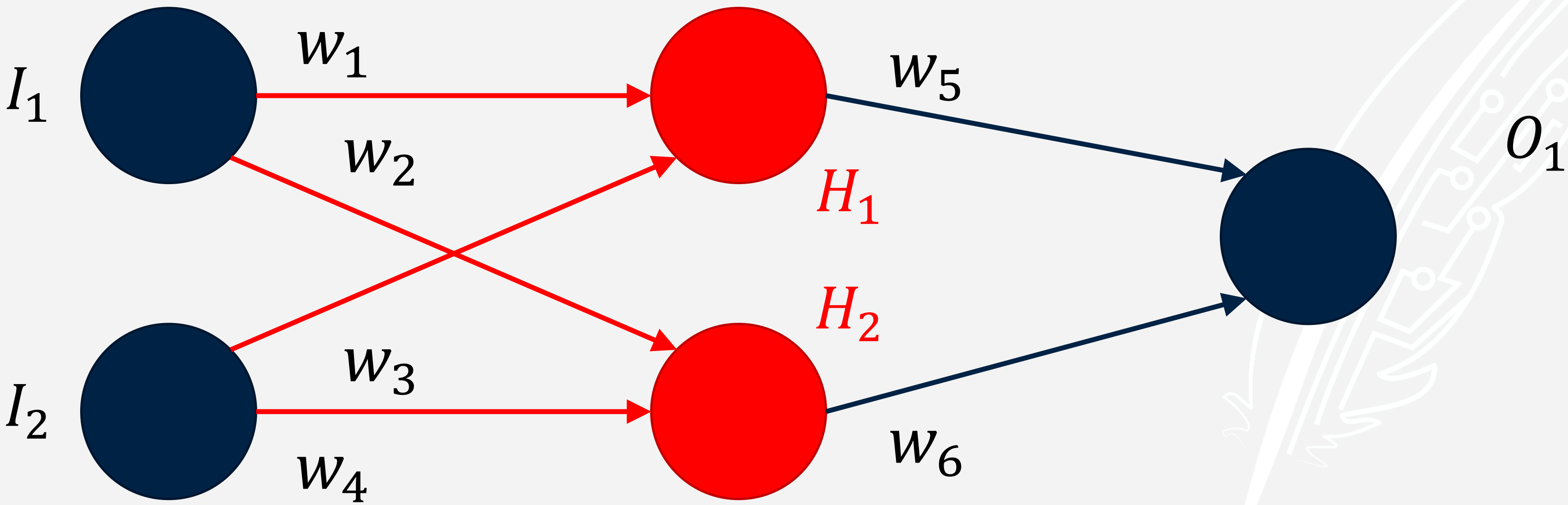
Backpropagation Example

This network has 2 hidden nodes, H_1 and H_2.



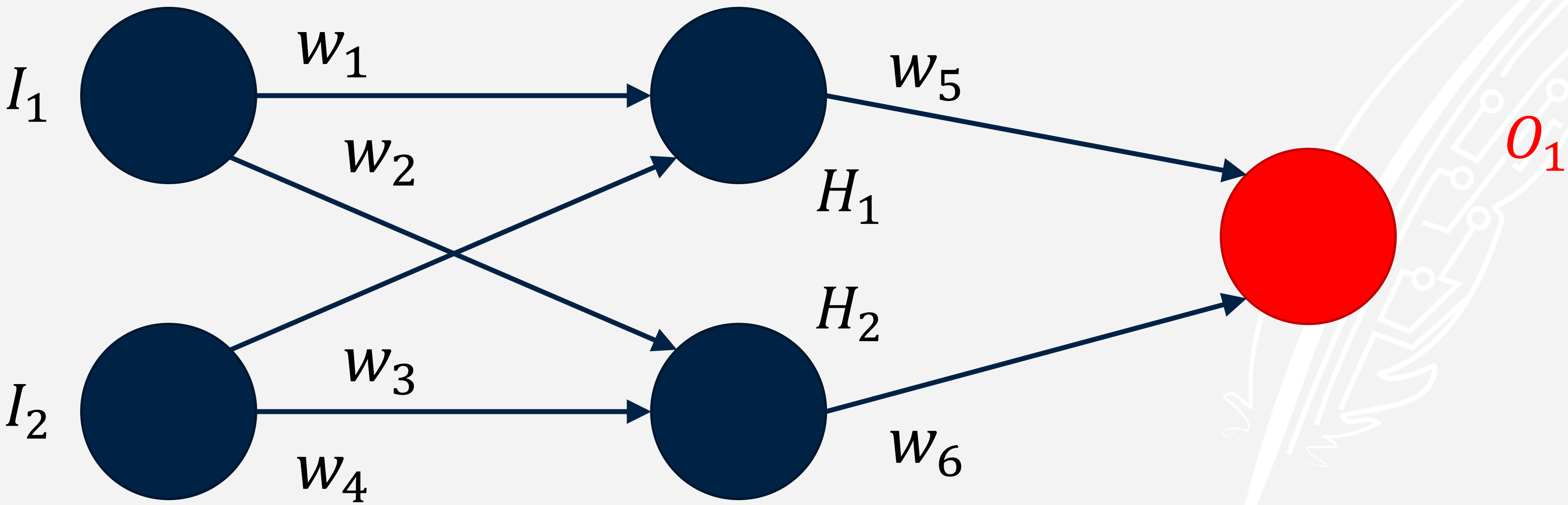
Backpropagation Example

Each hidden node receives input from all input nodes.



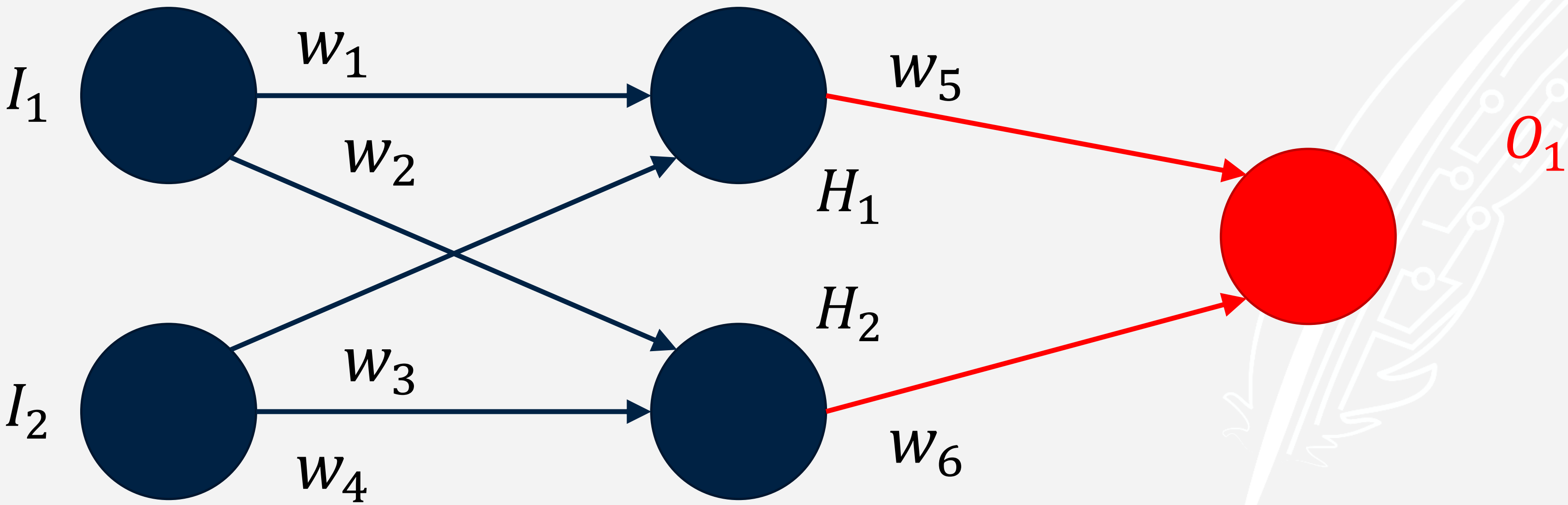
Backpropagation Example

This network has 1 output node, O_1.



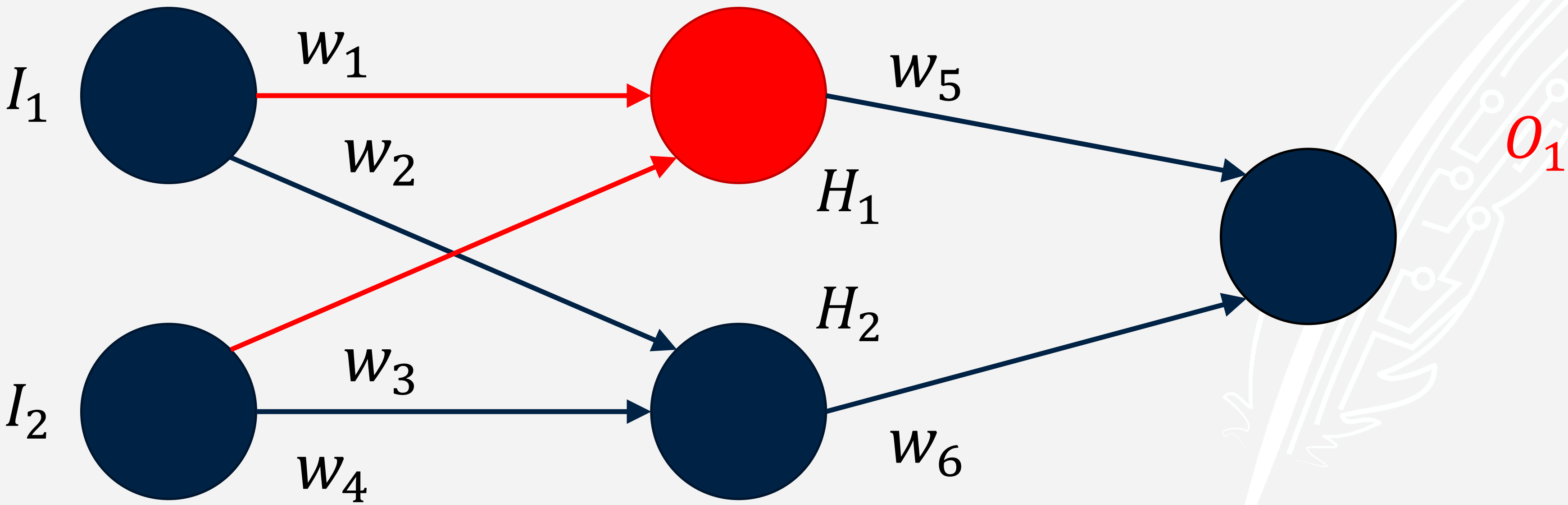
Backpropagation Example

Each output node receives input from each hidden node in the layer before it.



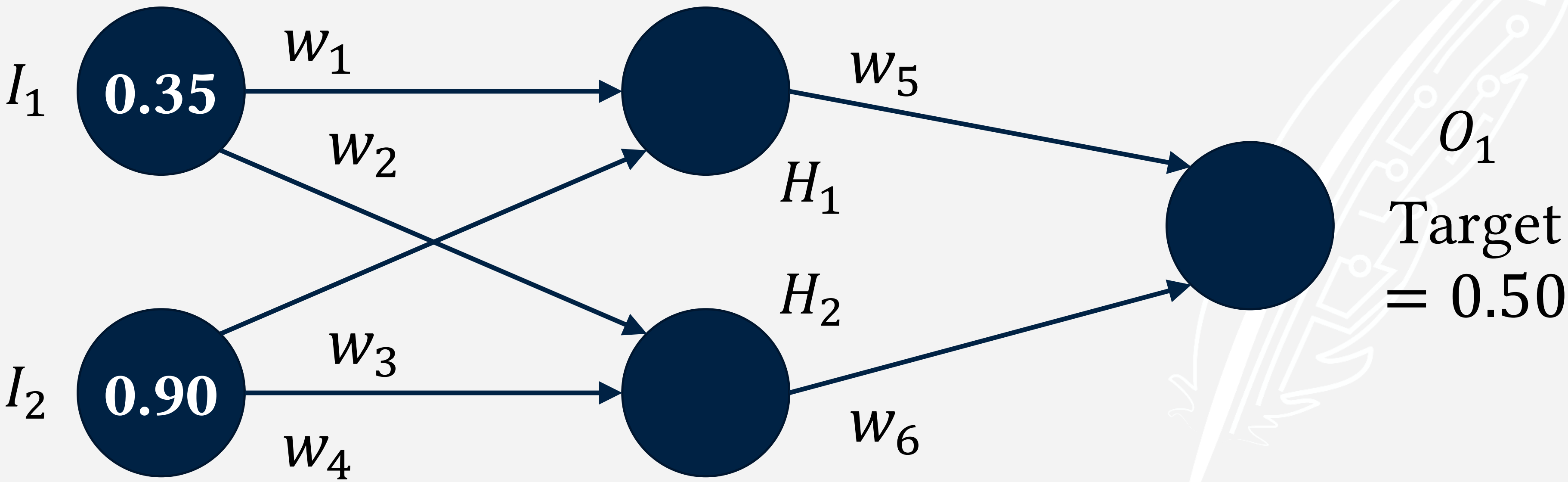
Backpropagation Example

Each hidden node receives input from each node in the layer before it.



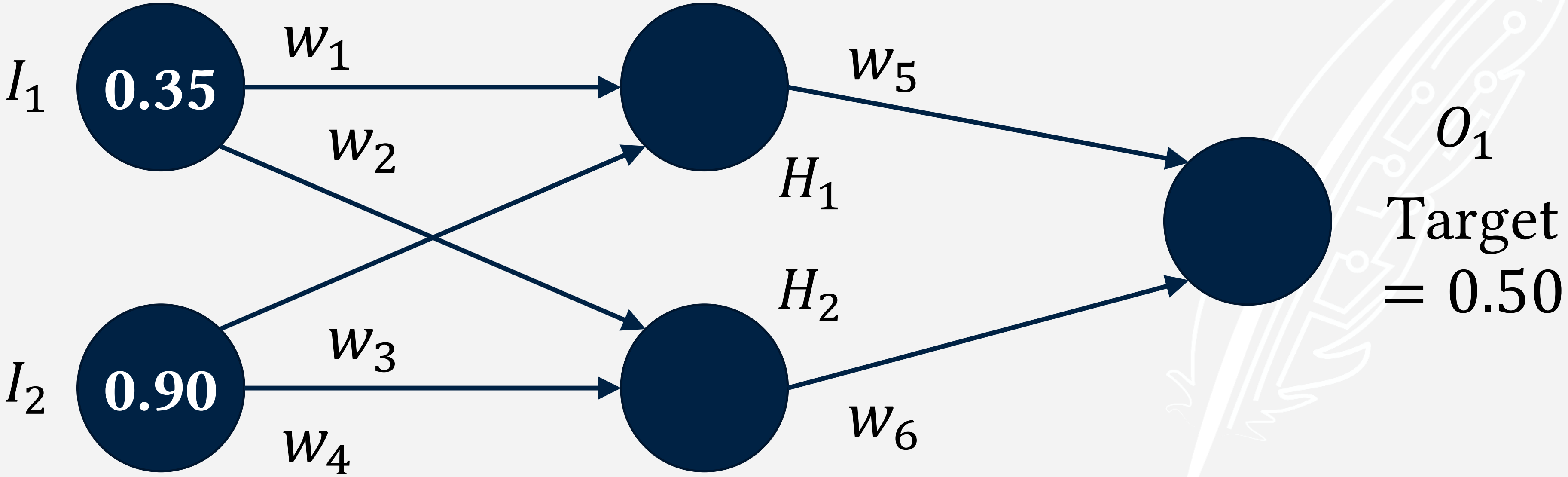
Backpropagation Example

We want to train this network to output 0.5 given the inputs 0.35 and 0.9.



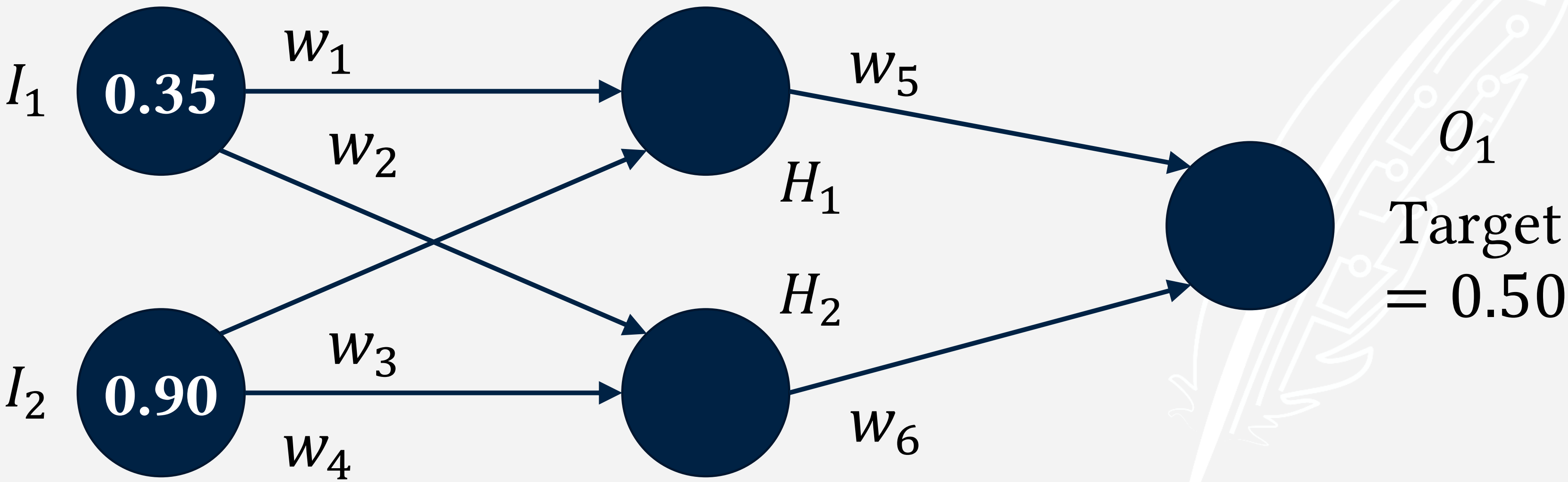
Backpropagation Example

We will do a single iteration of back propagation learning.



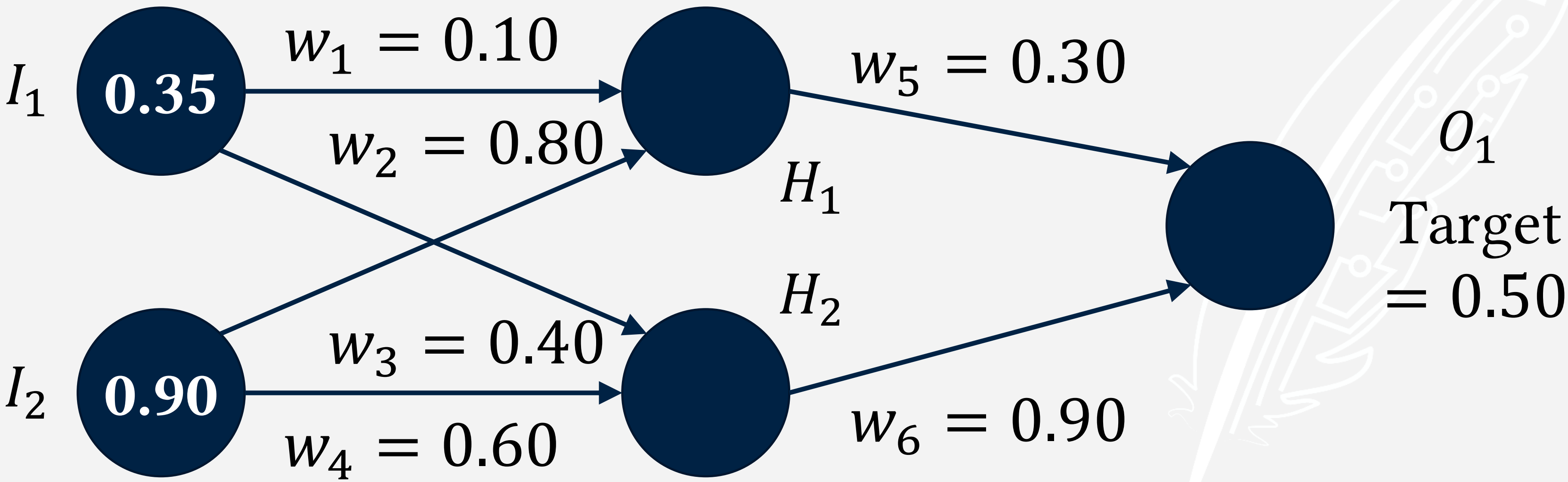
Backpropagation Example

Start by setting all weights in the network to random numbers.



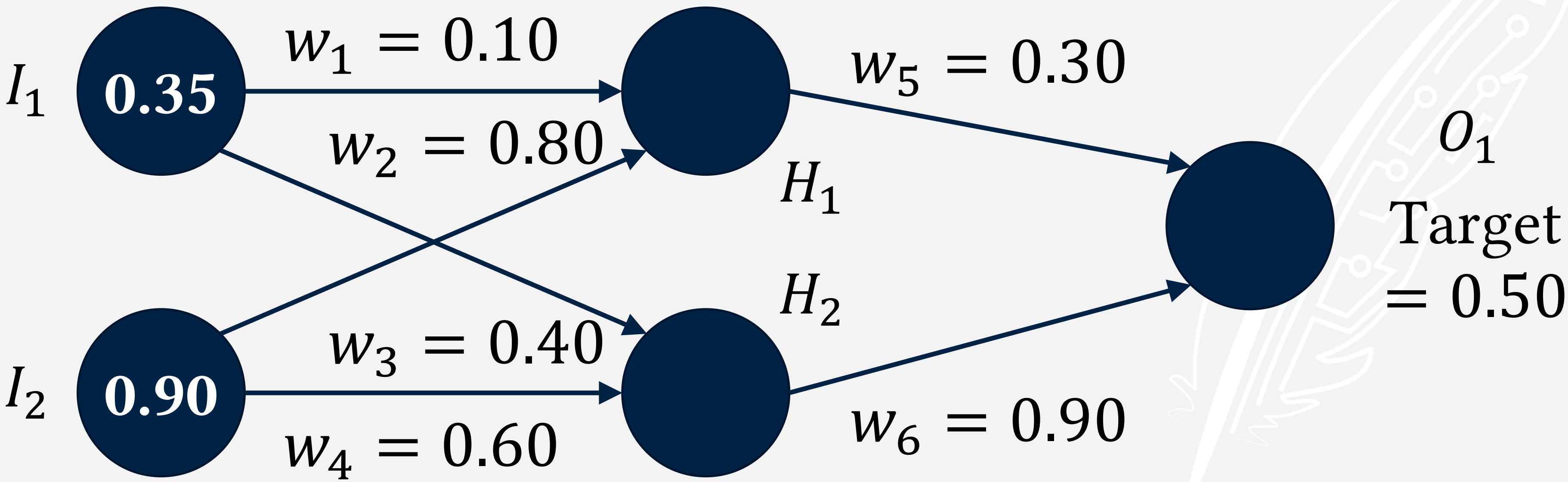
Backpropagation Example

Start by setting all weights in the network to random numbers.



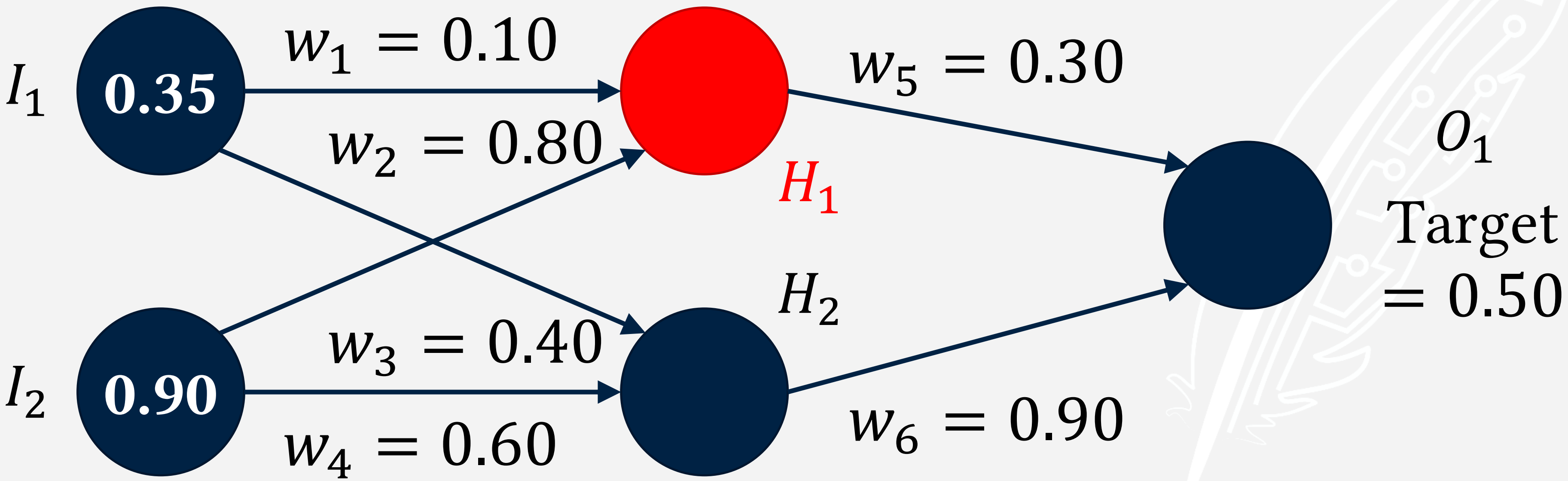
Backpropagation Example

Feed values forward from the input nodes to the hidden nodes to calculate the output node value.



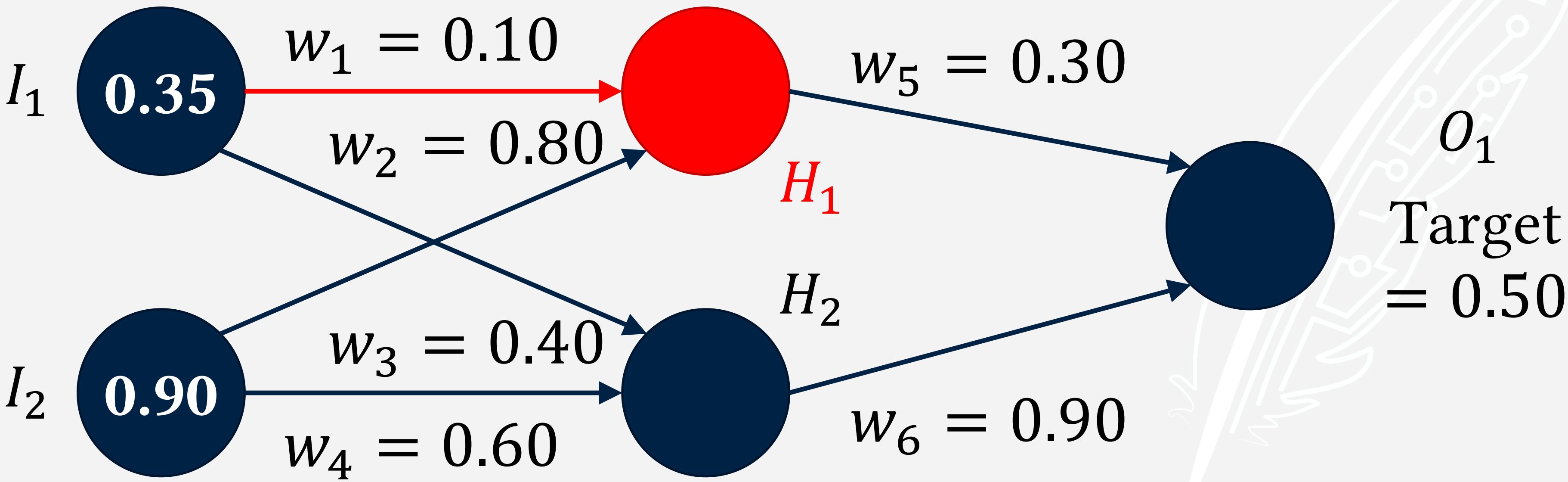
Backpropagation Example

$H_1 = ?$



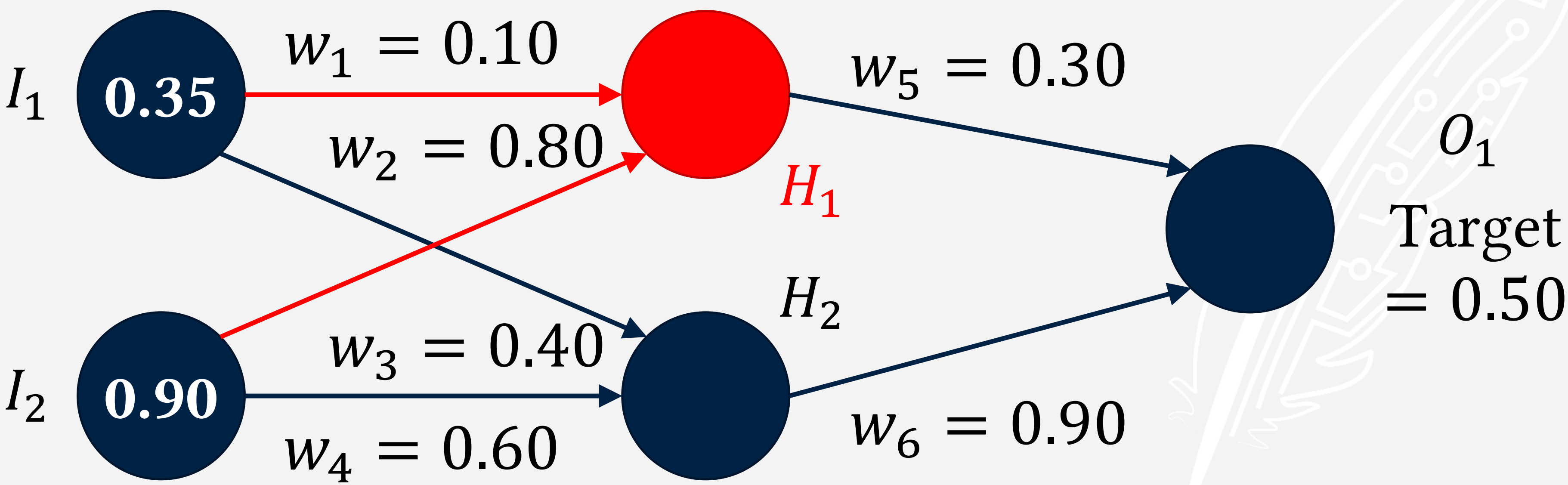
Backpropagation Example

$H_1 = (0.35 \cdot 0.10)$



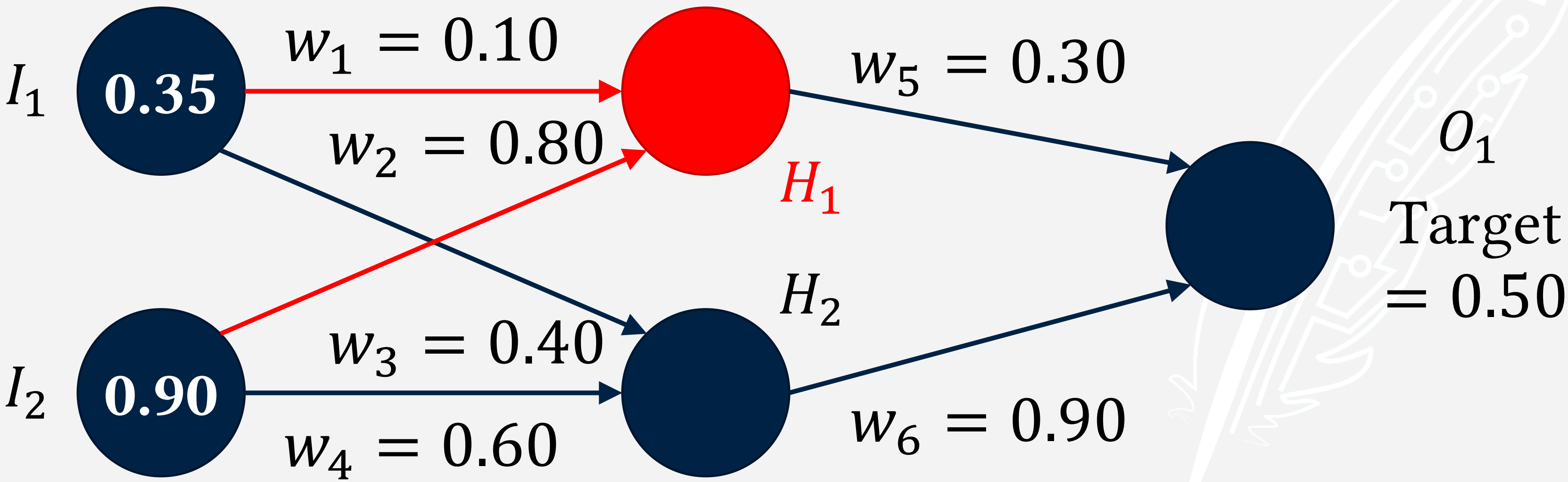
Backpropagation Example

$$H_1 = (0.35 \cdot 0.10) + (0.90 \cdot 0.80)$$



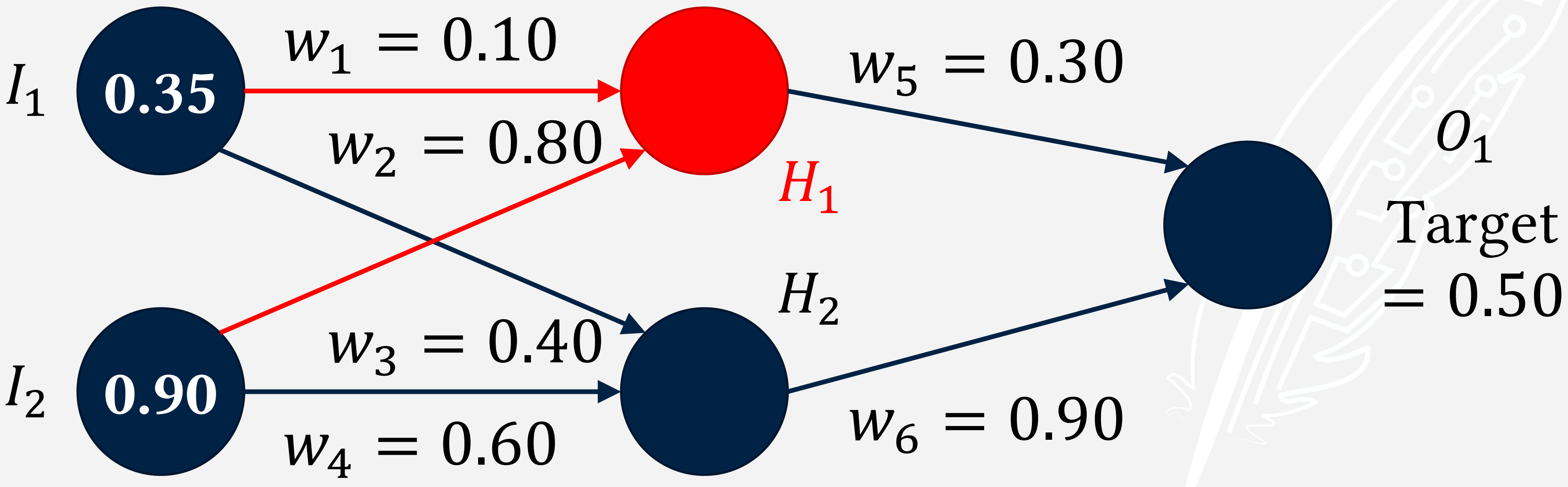
Backpropagation Example

$H_1 = 0.755$



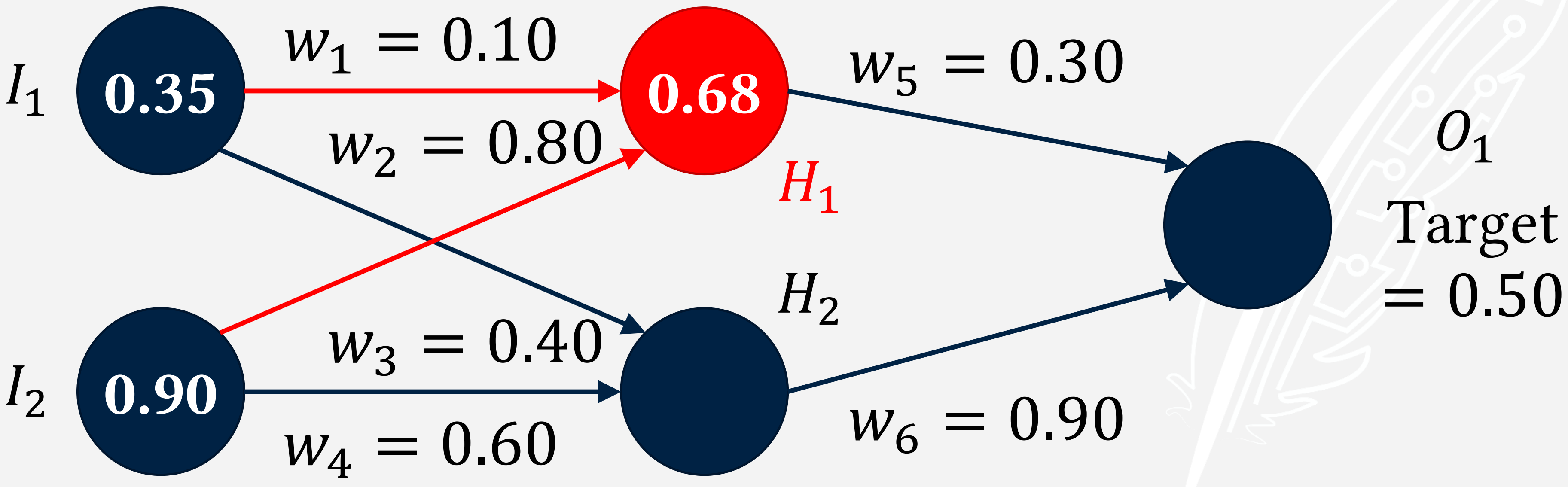
Backpropagation Example

$$H_1 = \frac{1}{1 + e^{-0.755}}$$



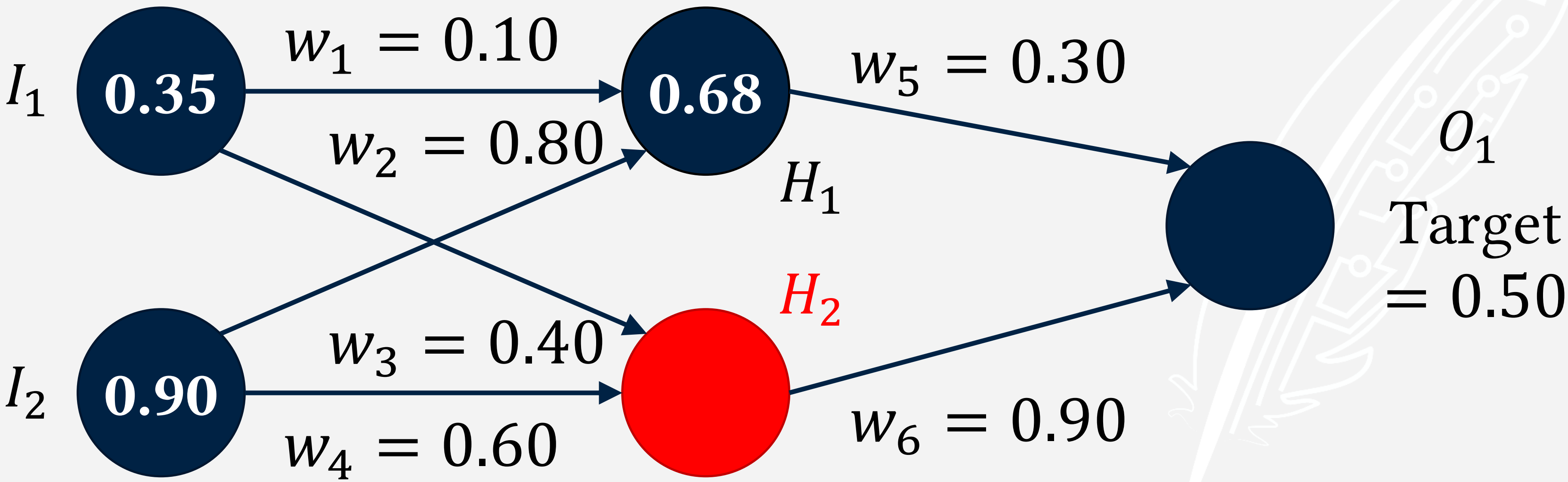
Backpropagation Example

$H_1 \approx 0.68$



Backpropagation Example

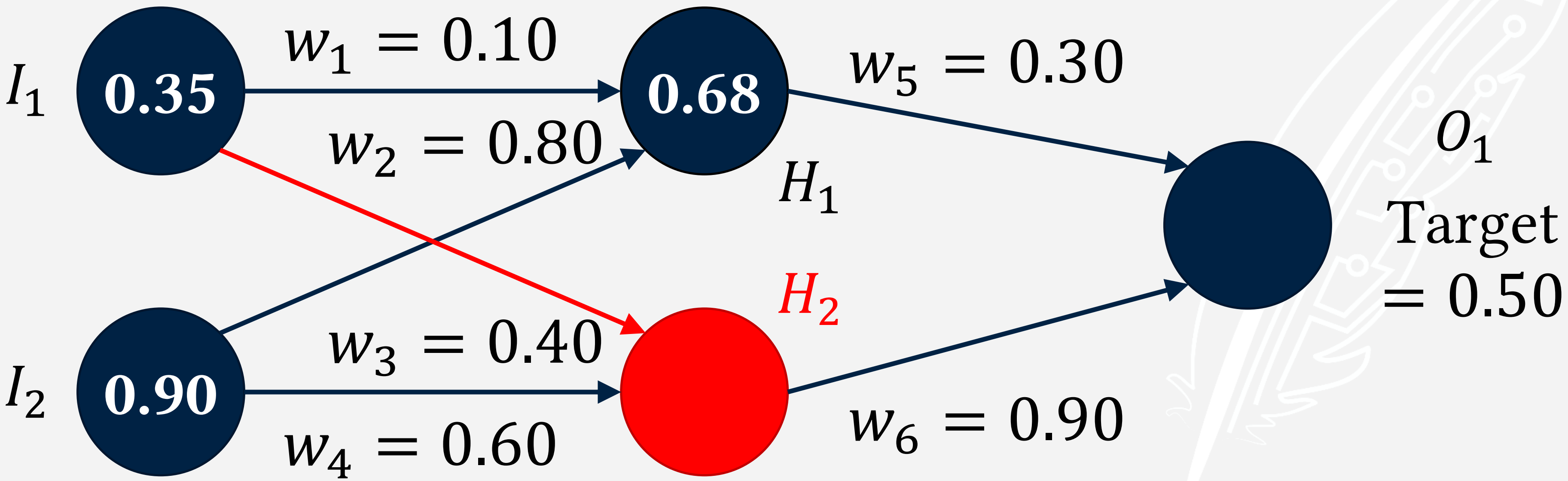
$H_2 = ?$



NIL

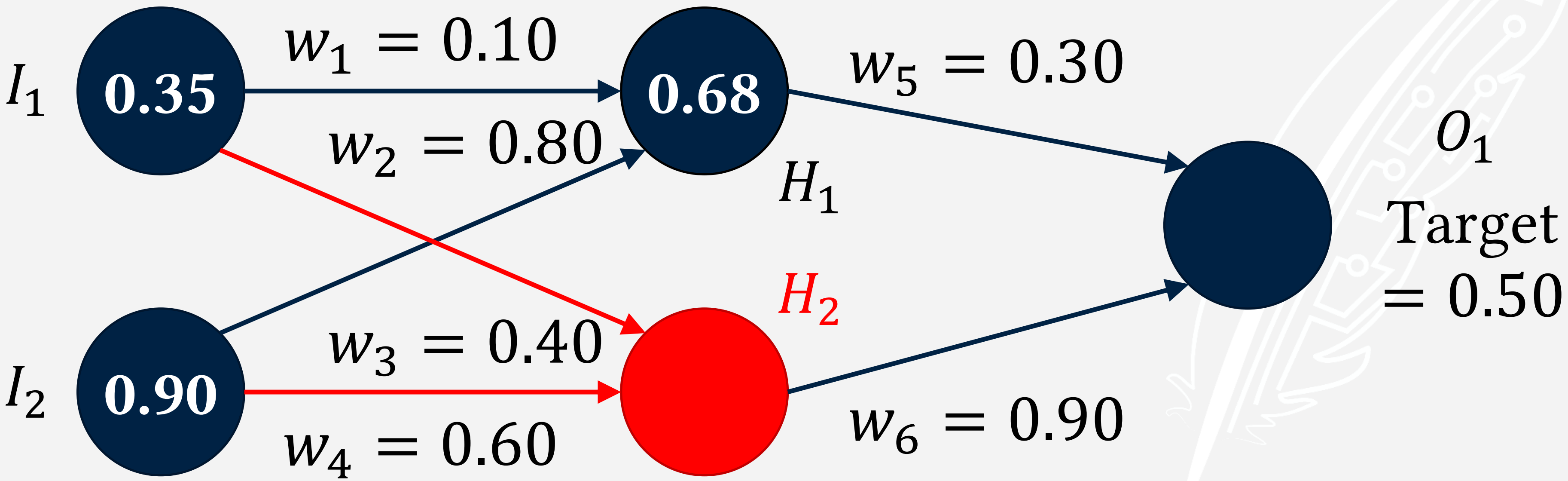
Backpropagation Example

$H_2 = (0.35 \cdot 0.40)$



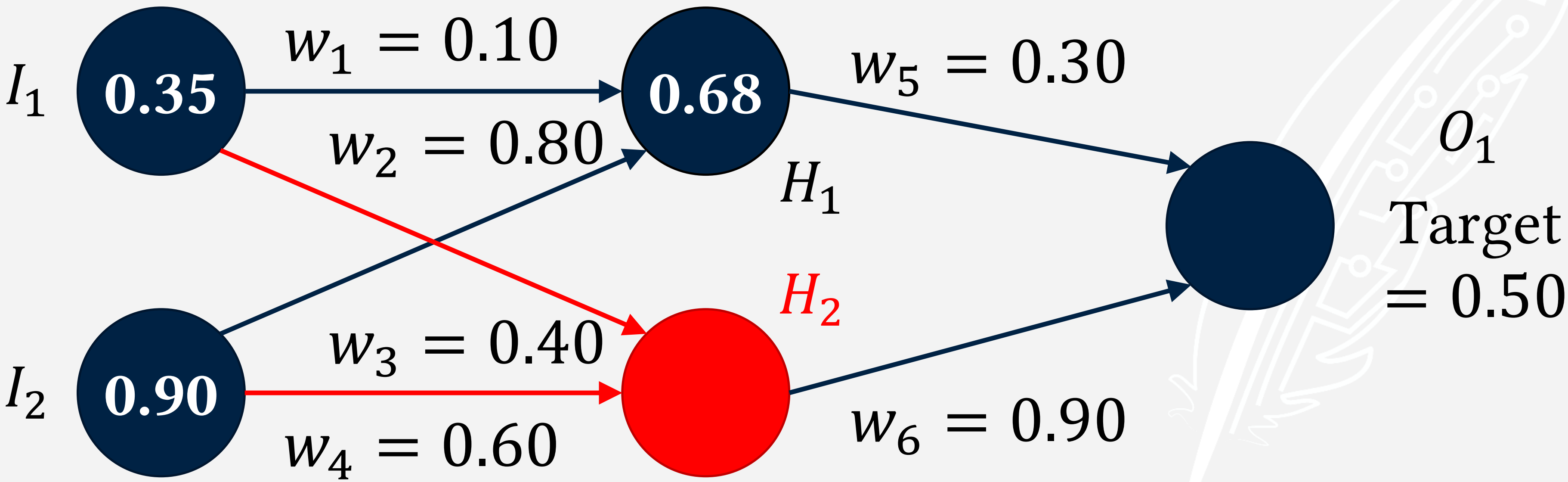
Backpropagation Example

$$H_2 = (0.35 \cdot 0.40) + (0.90 \cdot 0.60)$$



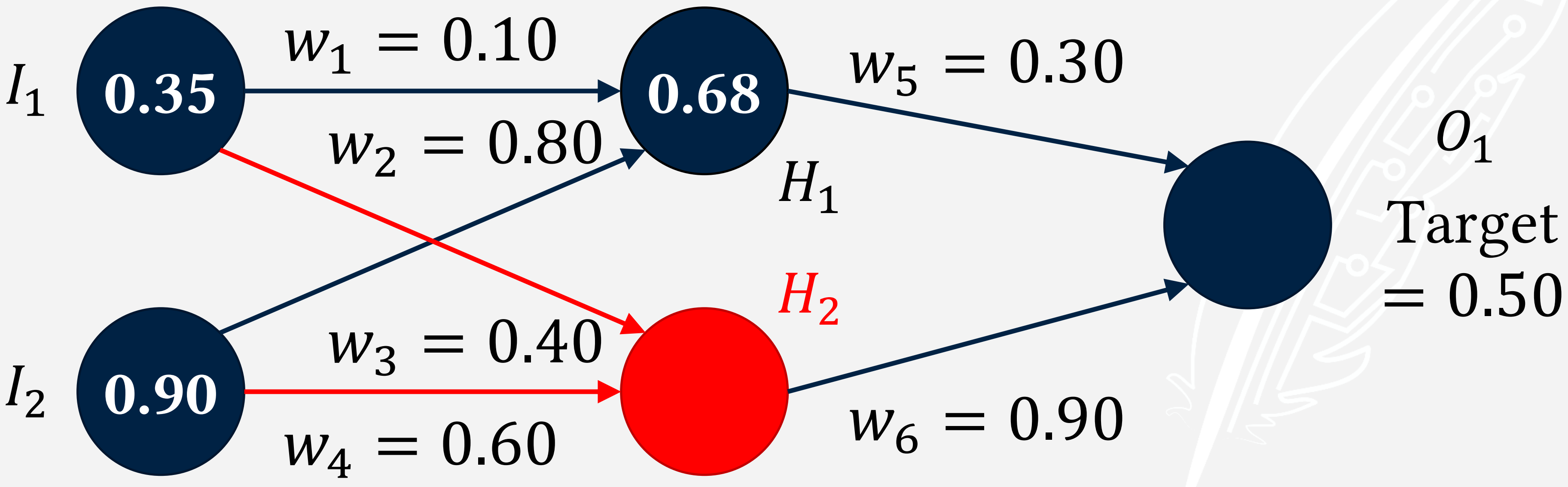
Backpropagation Example

$H_2 = 0.68$



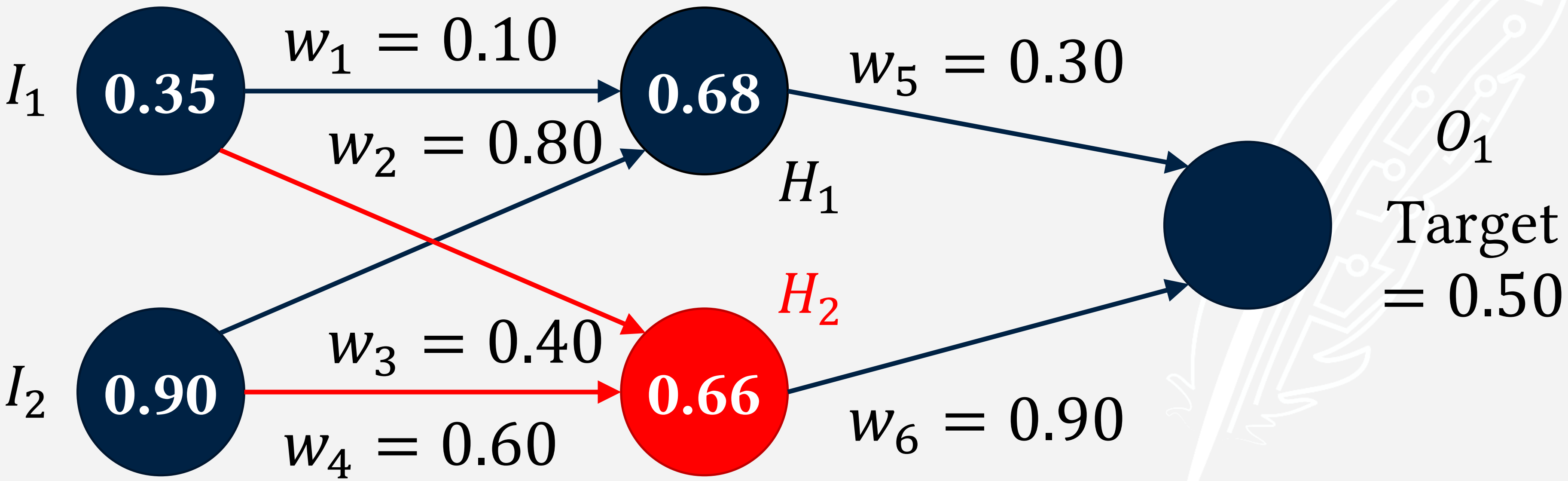
Backpropagation Example

$$H_2 = \frac{1}{1 + e^{-0.68}}$$



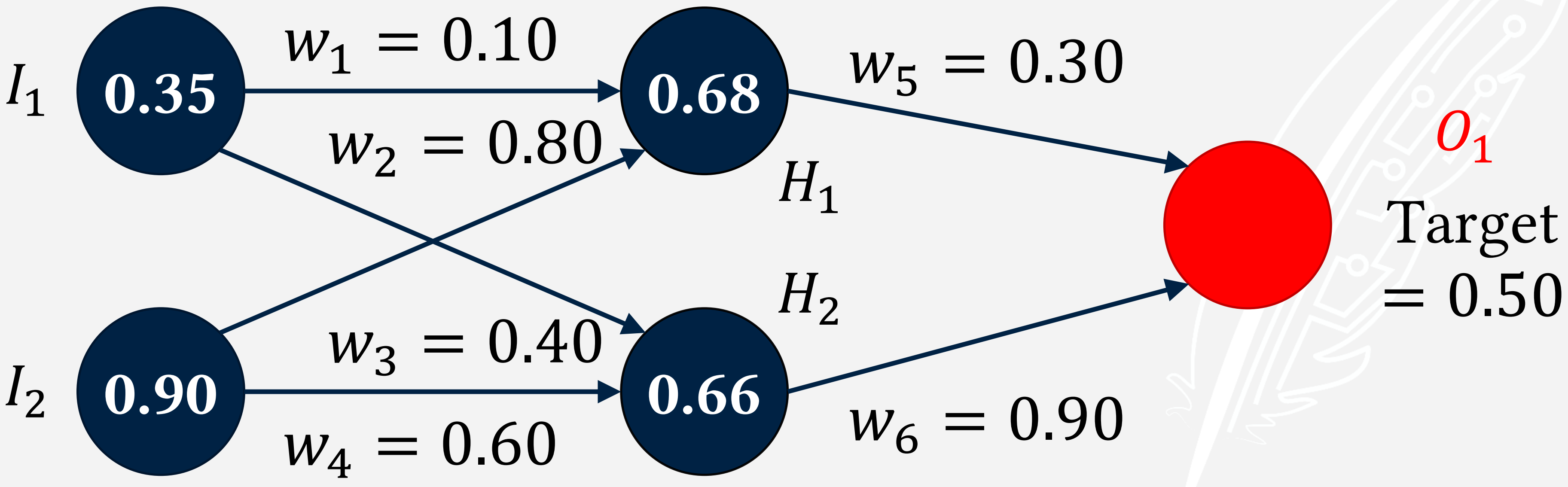
Backpropagation Example

$H_2 \approx 0.66$



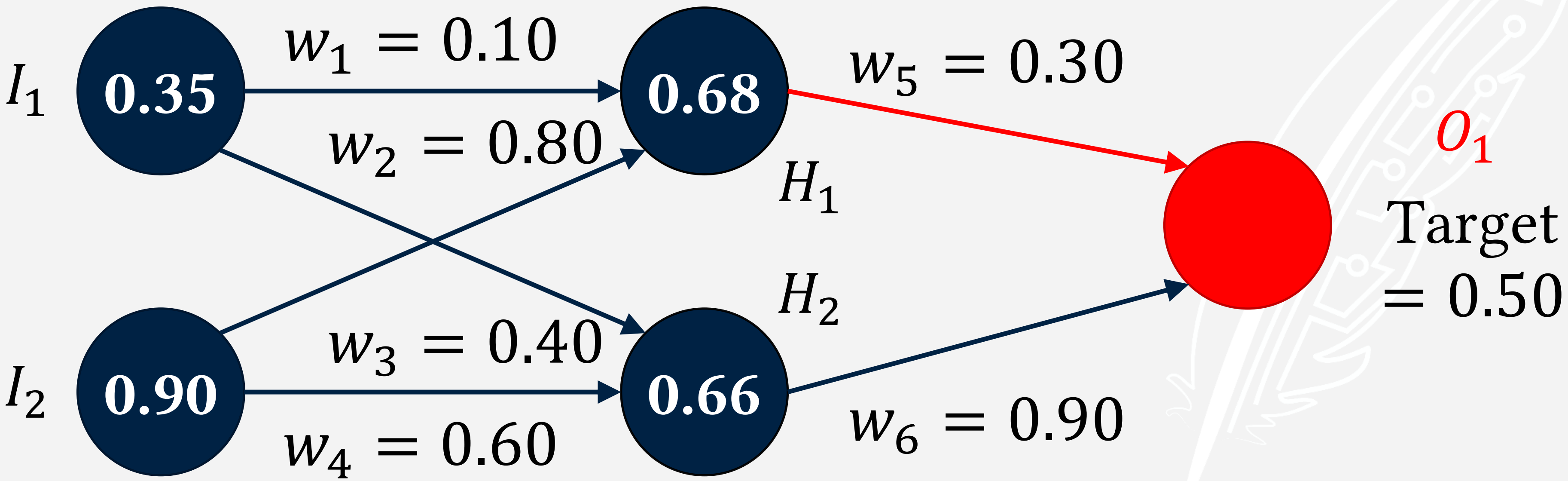
Backpropagation Example

$O_1 = ?$



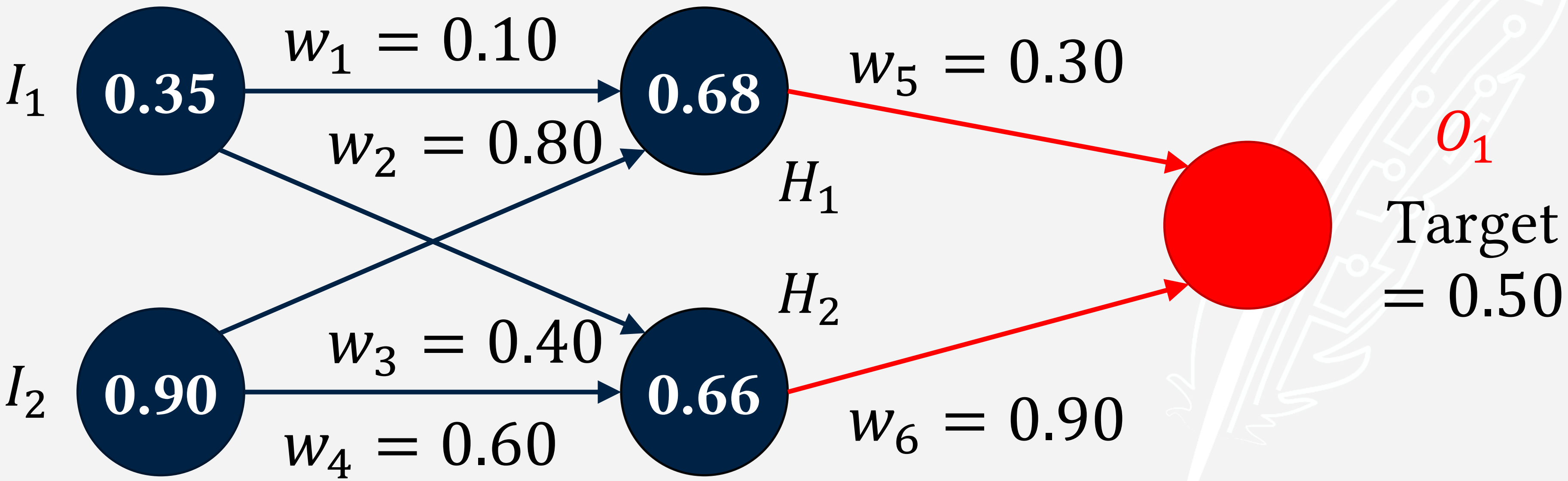
Backpropagation Example

$$O_1 = (0.68 \cdot 0.30)$$



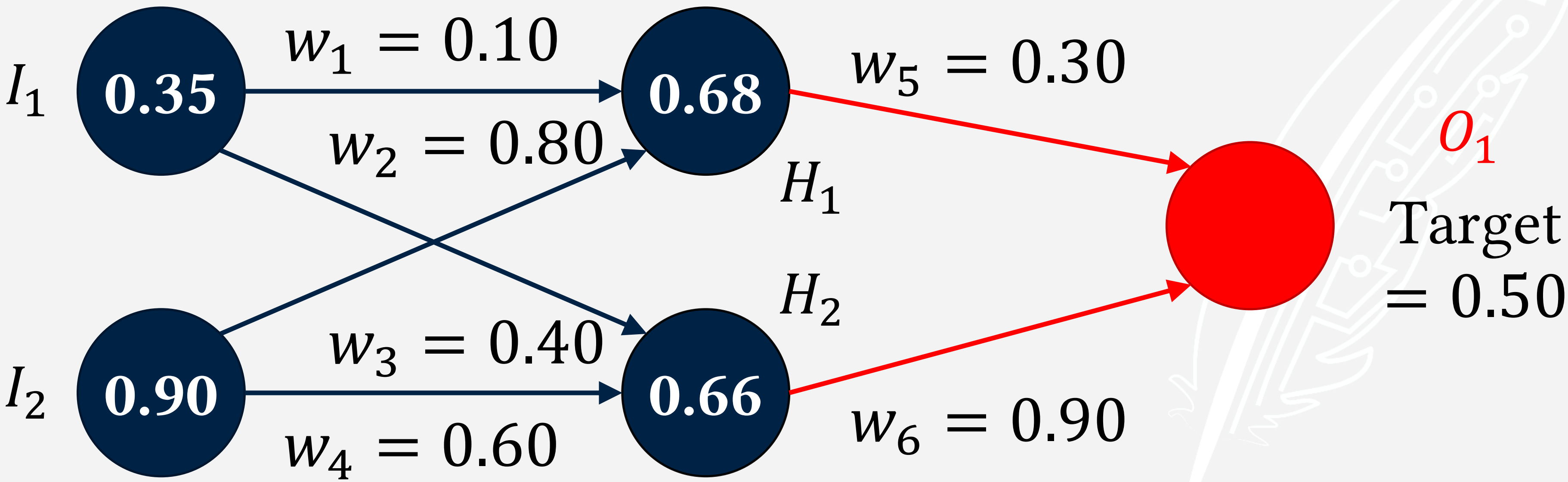
Backpropagation Example

$$O_1 = (0.68 \cdot 0.30) + (0.66 \cdot 0.90)$$



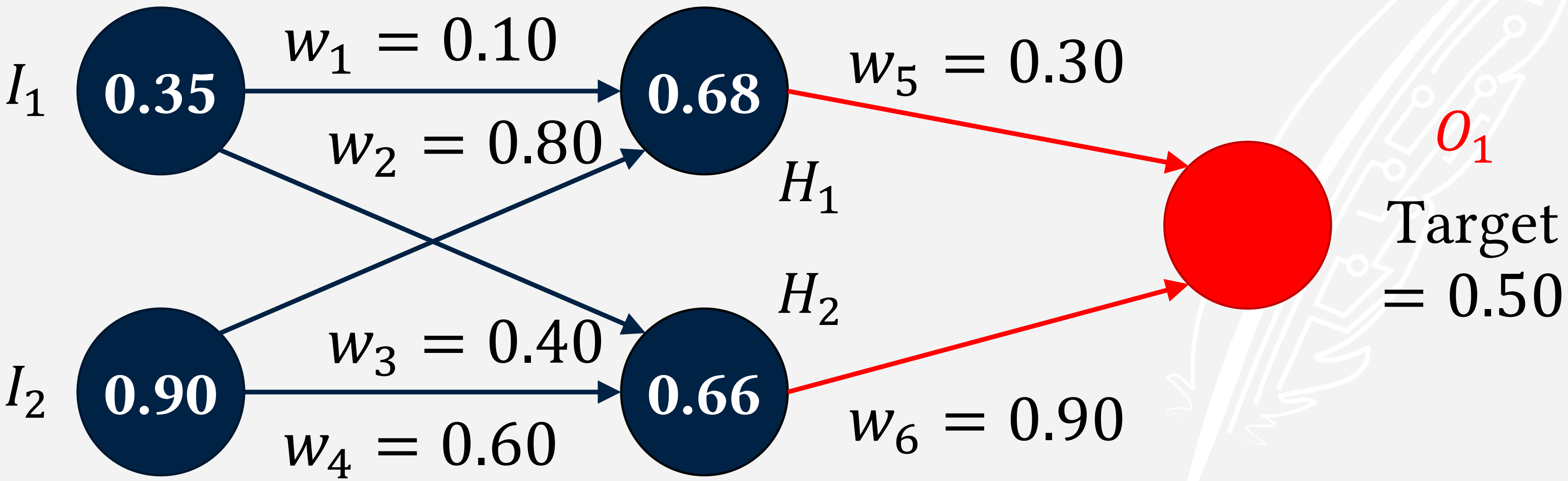
Backpropagation Example

$O_1 \approx 0.80$



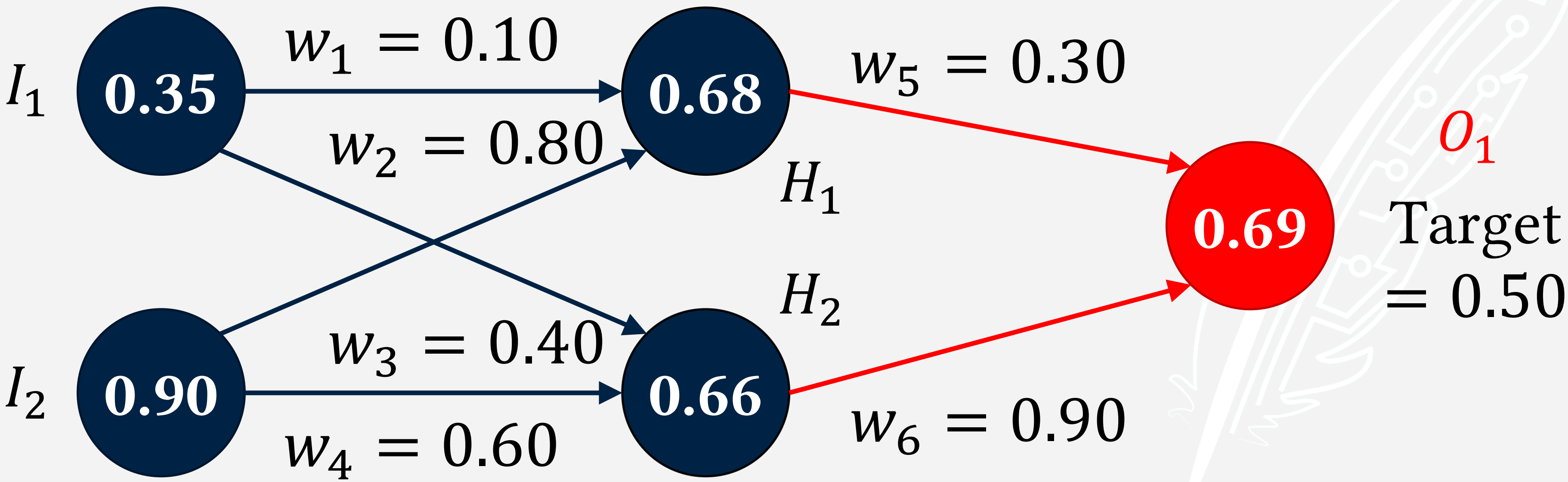
Backpropagation Example

$$O_1 \approx \frac{1}{1 + e^{-0.80}}$$



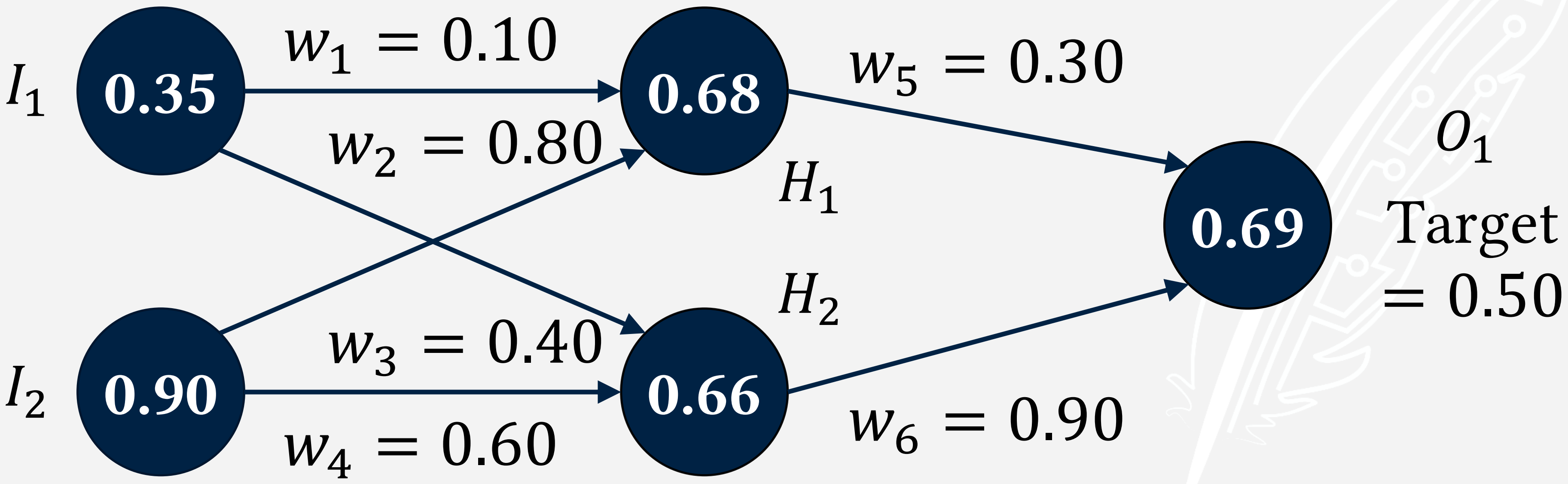
Backpropagation Example

$O_1 \approx 0.69$



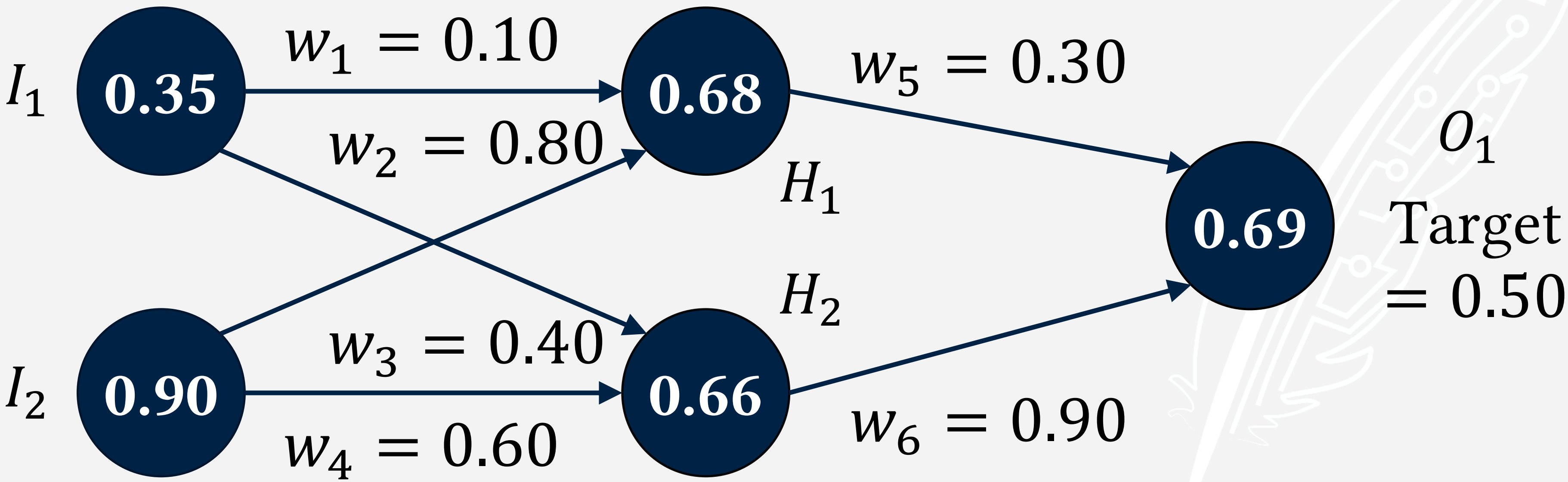
Backpropagation Example

Now we calculate the error of our network.



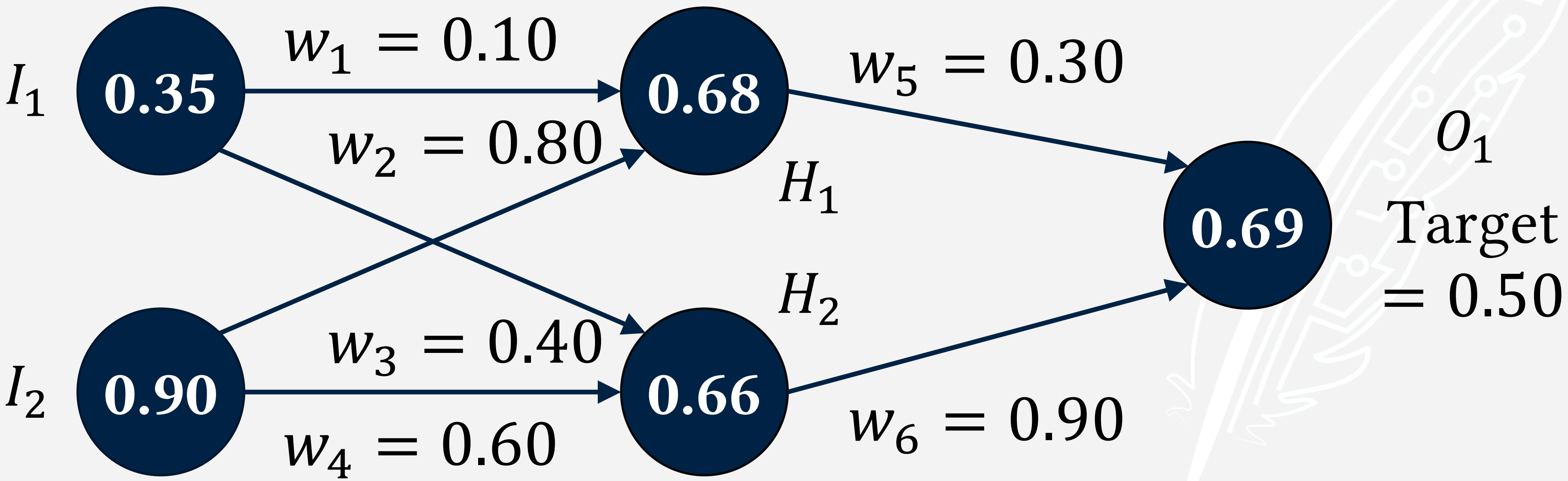
Backpropagation Example

$$error = correct - output$$



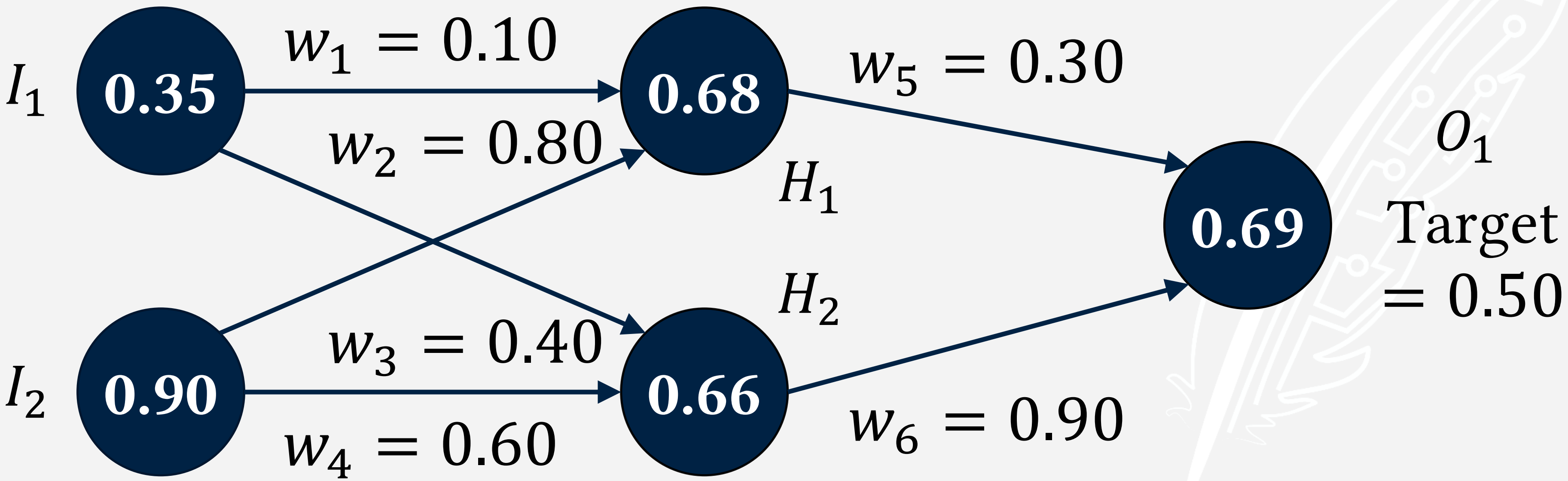
Backpropagation Example

$$error = 0.5 - 0.69$$



Backpropagation Example

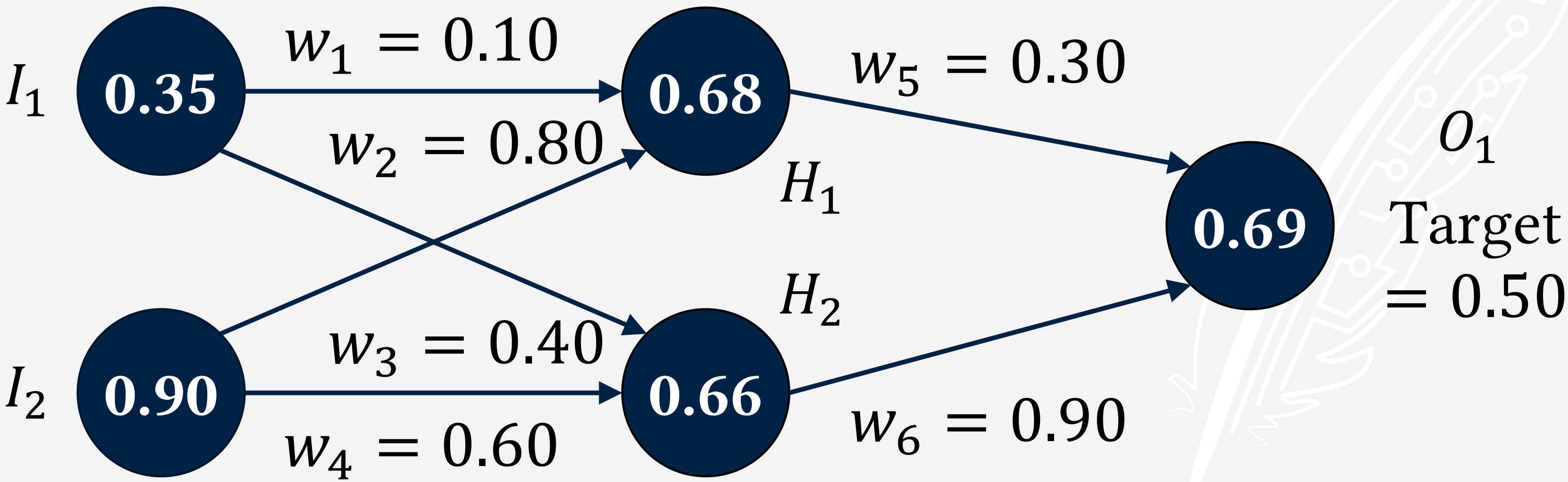
$error = -0.19$



NIL

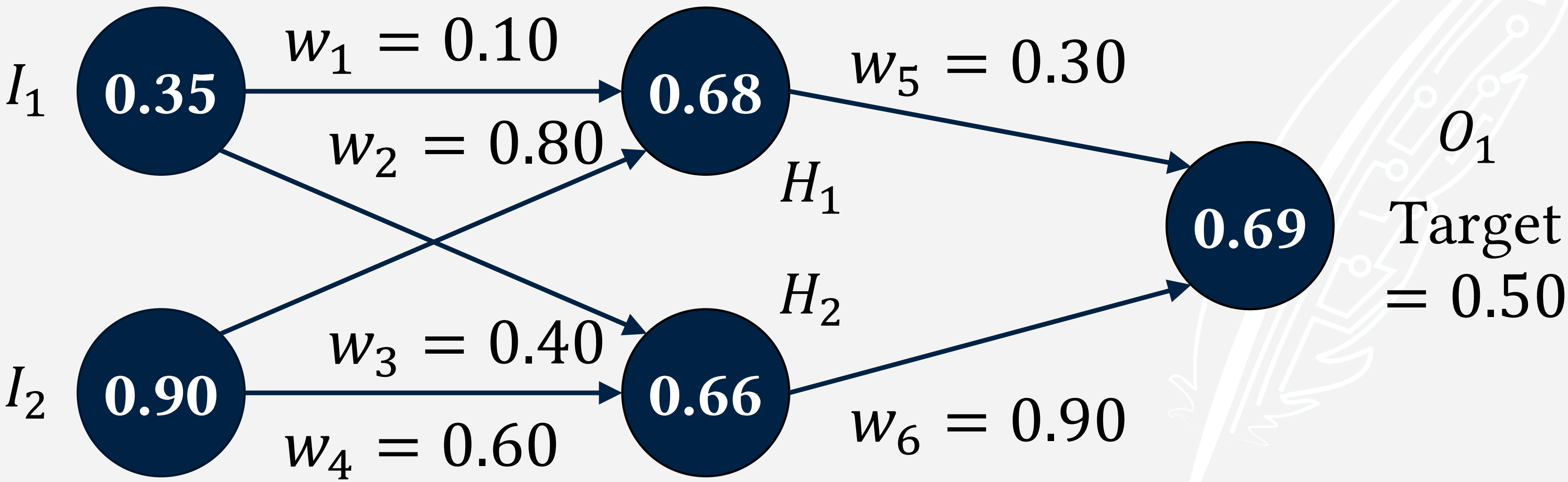
Backpropagation Example

Now we need to adjust the weights to reduce the error in the network.



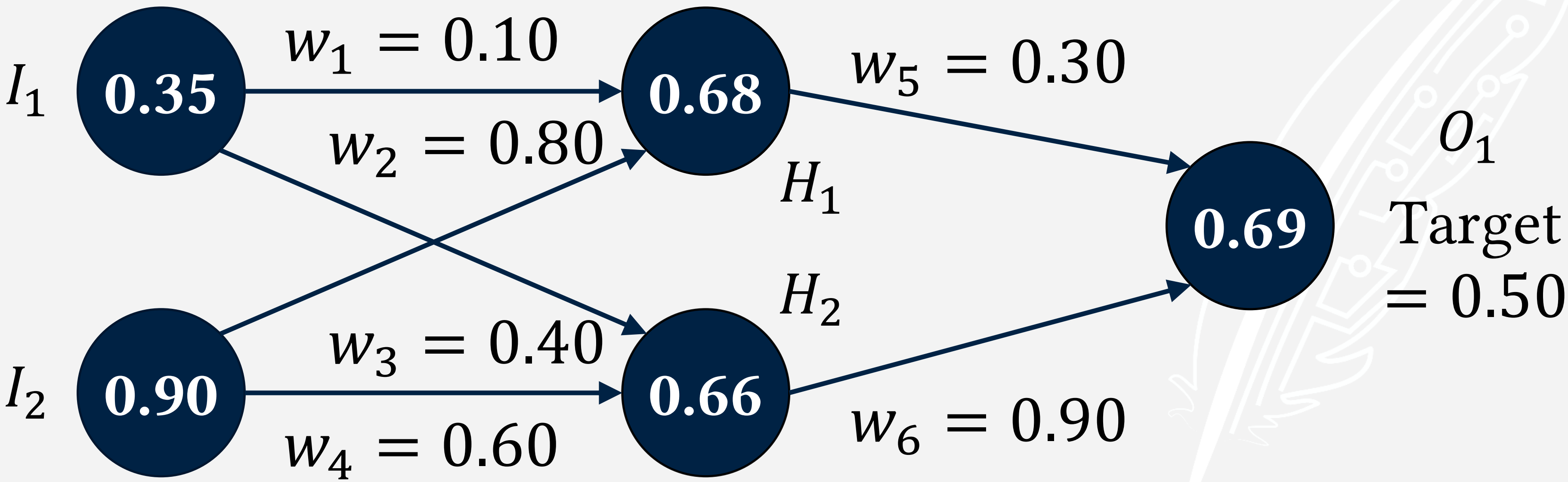
Backpropagation Example

We start with the output node and work backwards to the input nodes.



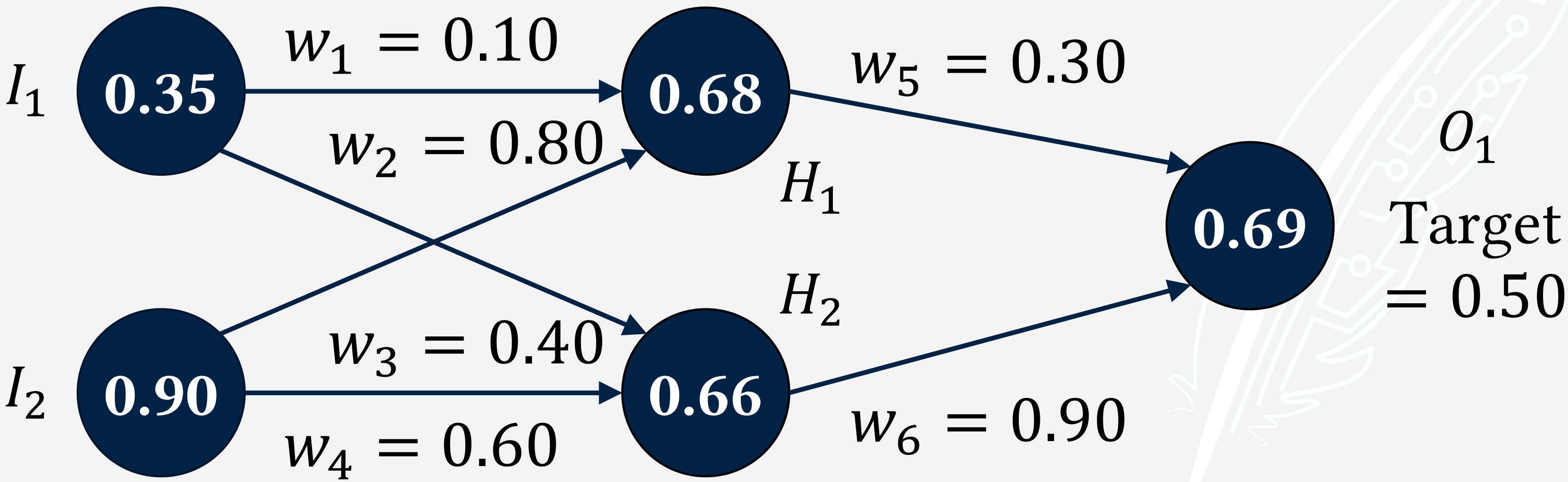
Backpropagation Example

$$\Delta(X) = X(1 - X)\sum_{Y} w_{X \to Y}\Delta(Y)$$



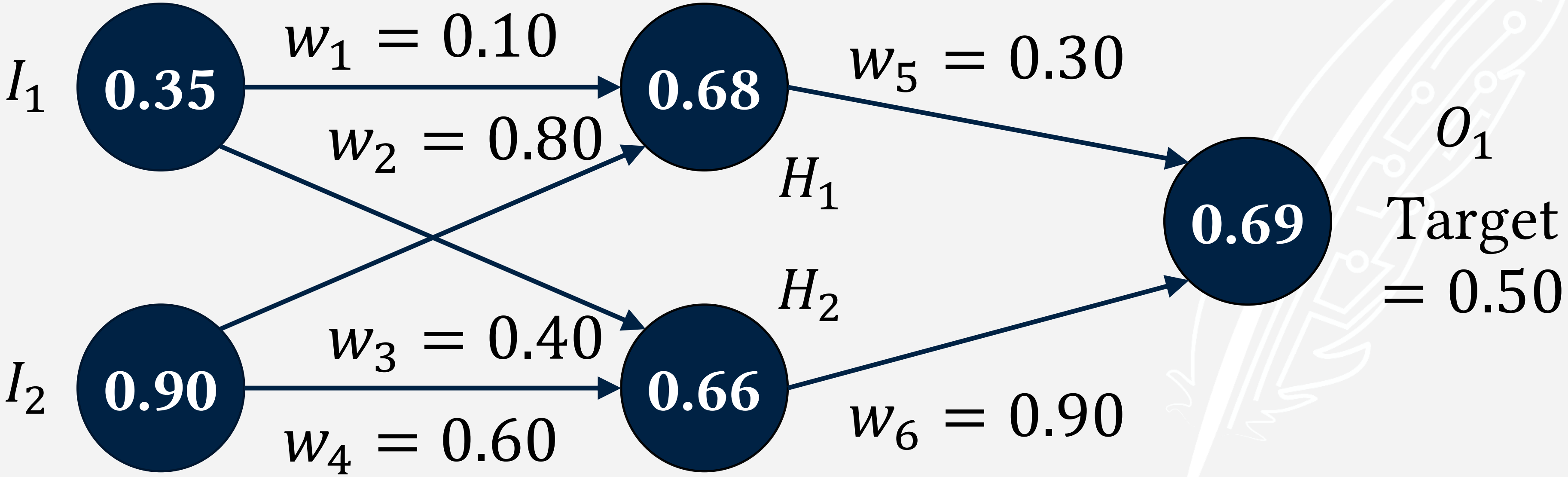
Backpropagation Example

$$\Delta(O_1) = O_1(1 - O_1)\sum_{Y} w_{O_1\to Y}\Delta(Y)$$



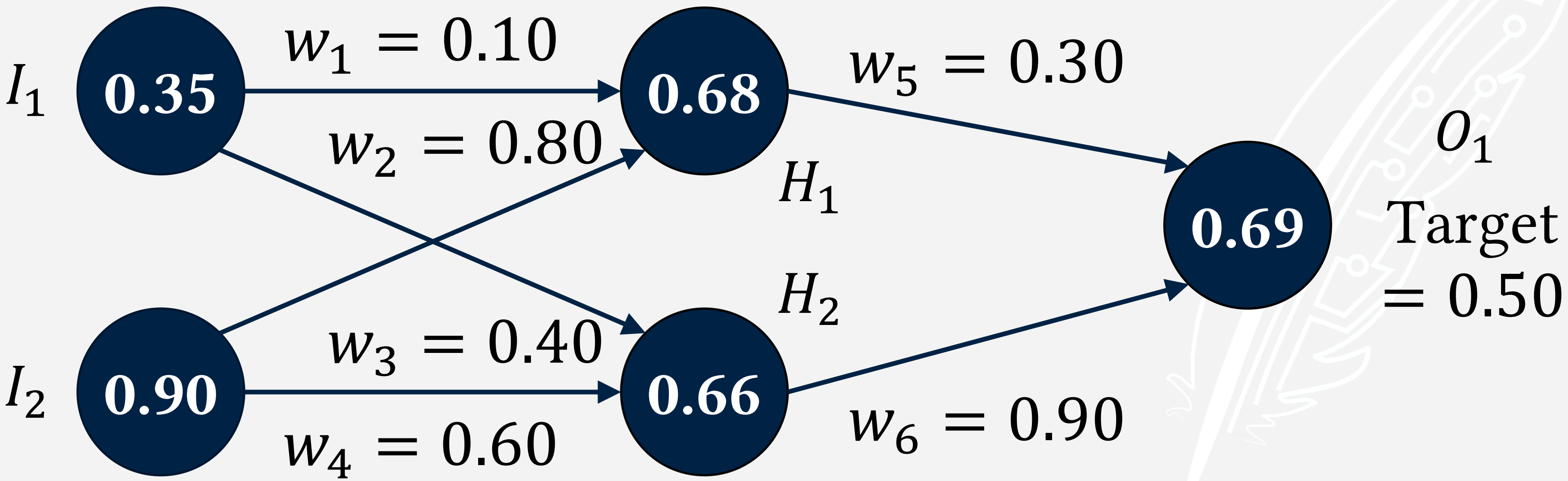
Backpropagation Example

$$\Delta(O_1) = O_1(1 - O_1)(-0.19)$$



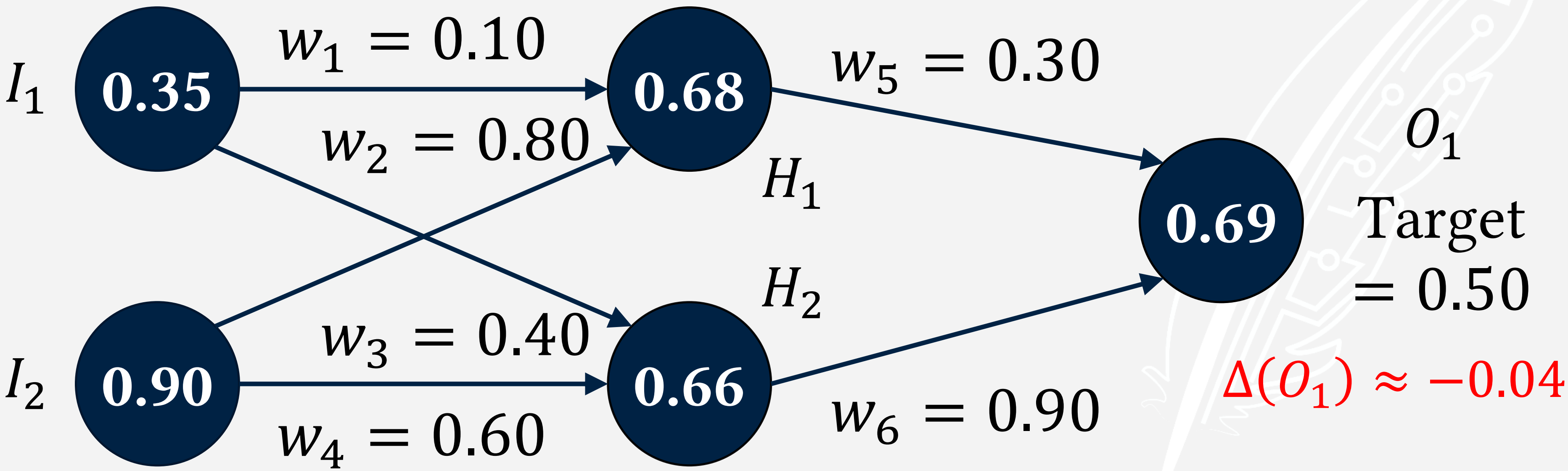
Backpropagation Example

$$\Delta(O_1) = 0.69(1 - 0.69)(-0.19)$$



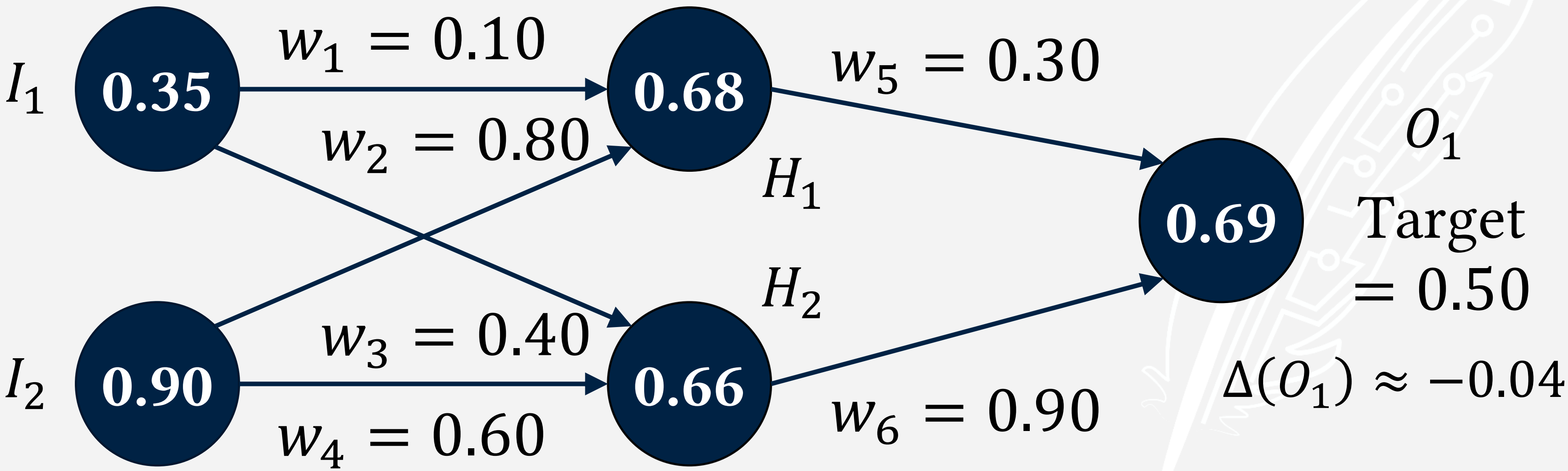
Backpropagation Example

$\Delta(O_1) \approx -0.04$



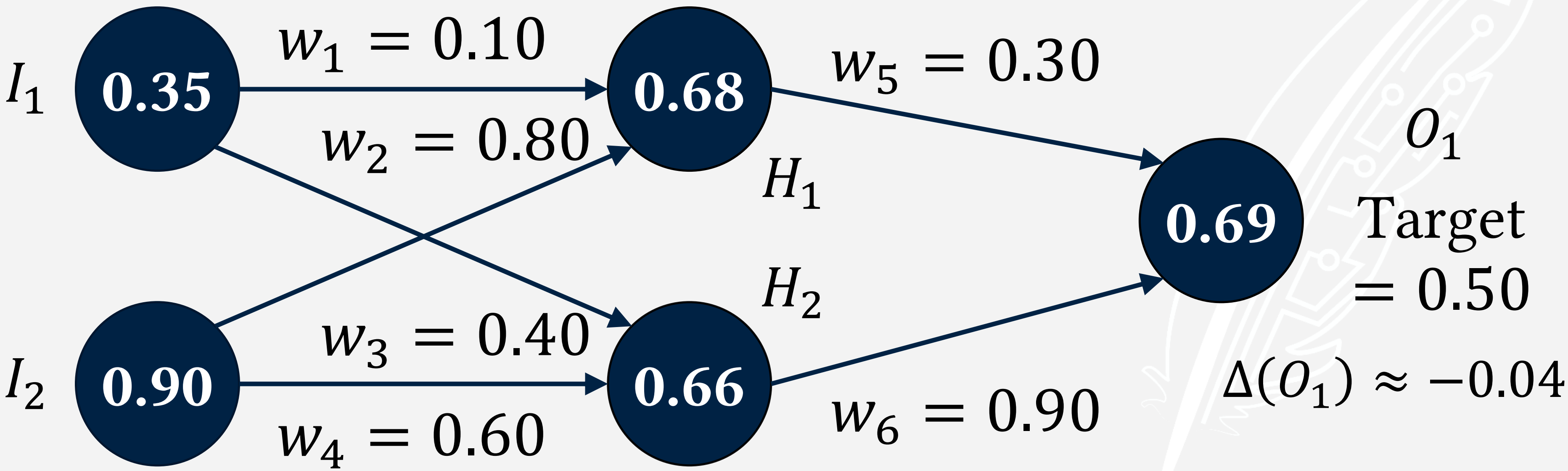
Backpropagation Example

Now we need to update the weights of the edges leading to the output nodes.



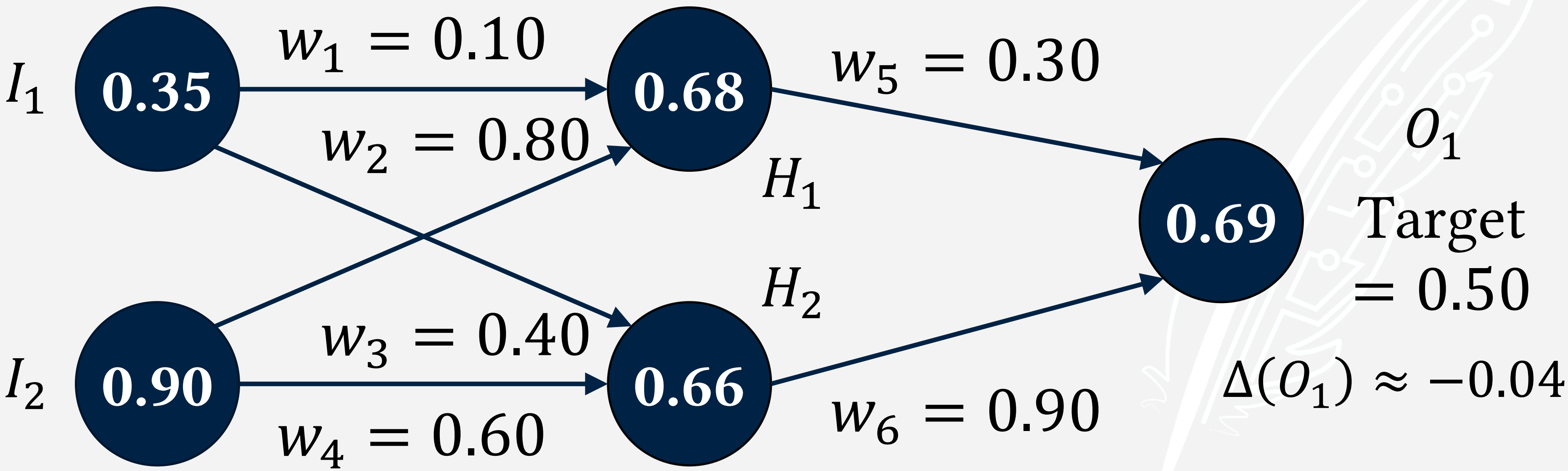
Backpropagation Example

$$w_{X \to Y} = w_{X \to Y} + \left(X \cdot \Delta(Y)\right)$$



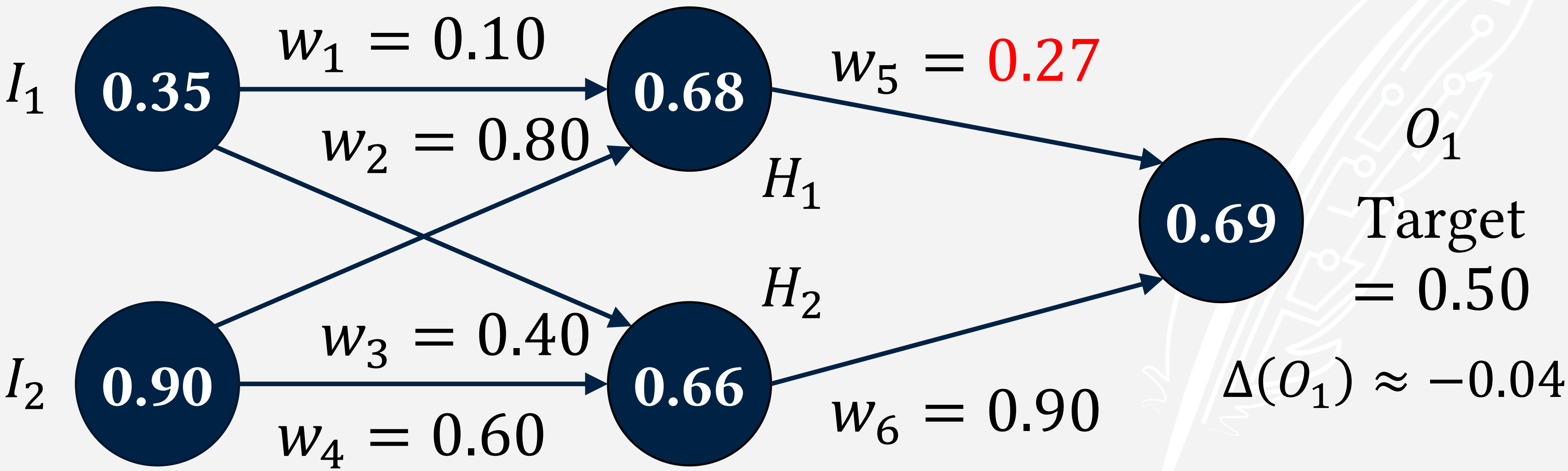
Backpropagation Example

$$w_5 = w_5 + \left(H_1 \cdot \Delta(O_1)\right)$$



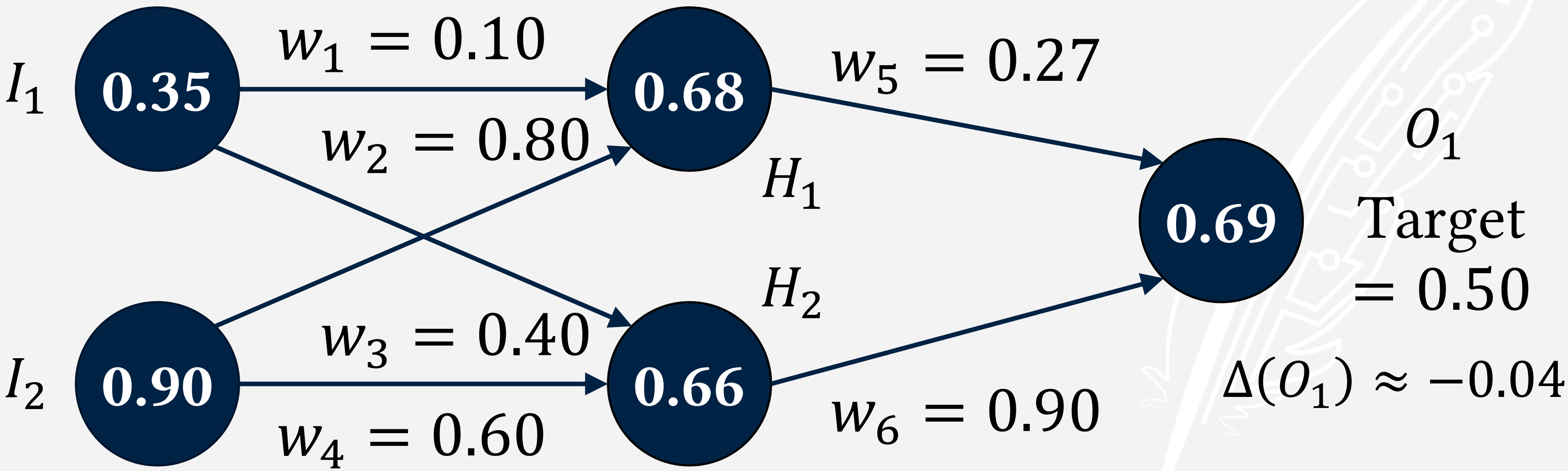
Backpropagation Example

$w_5 \approx 0.27$



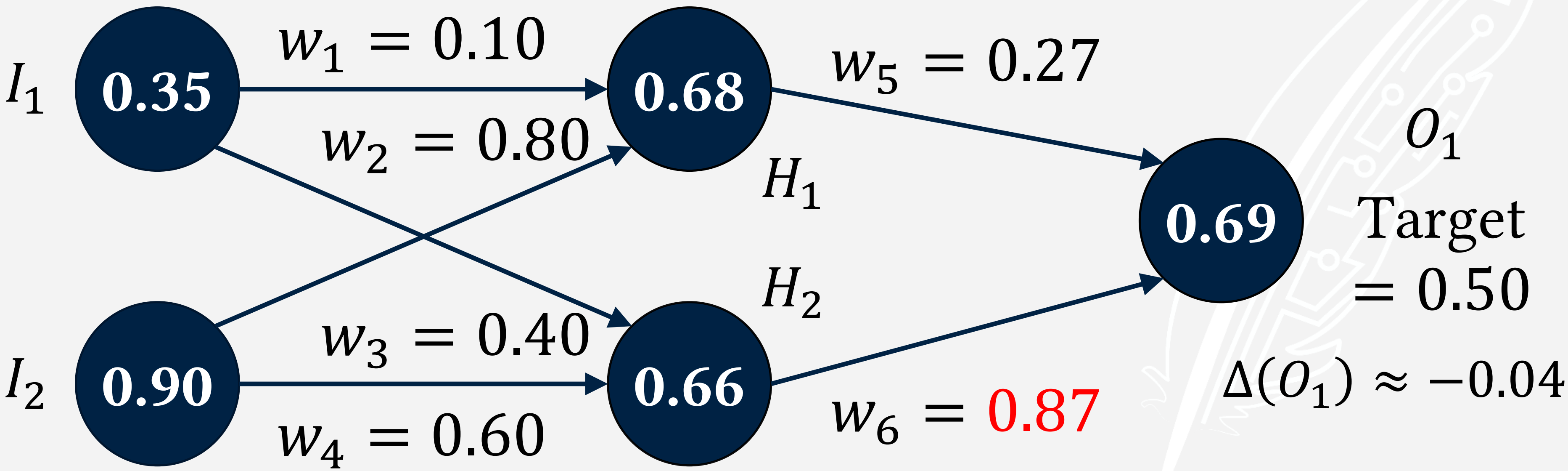
Backpropagation Example

$$w_6 = w_6 + \left(H_2 \cdot \Delta(O_1)\right)$$



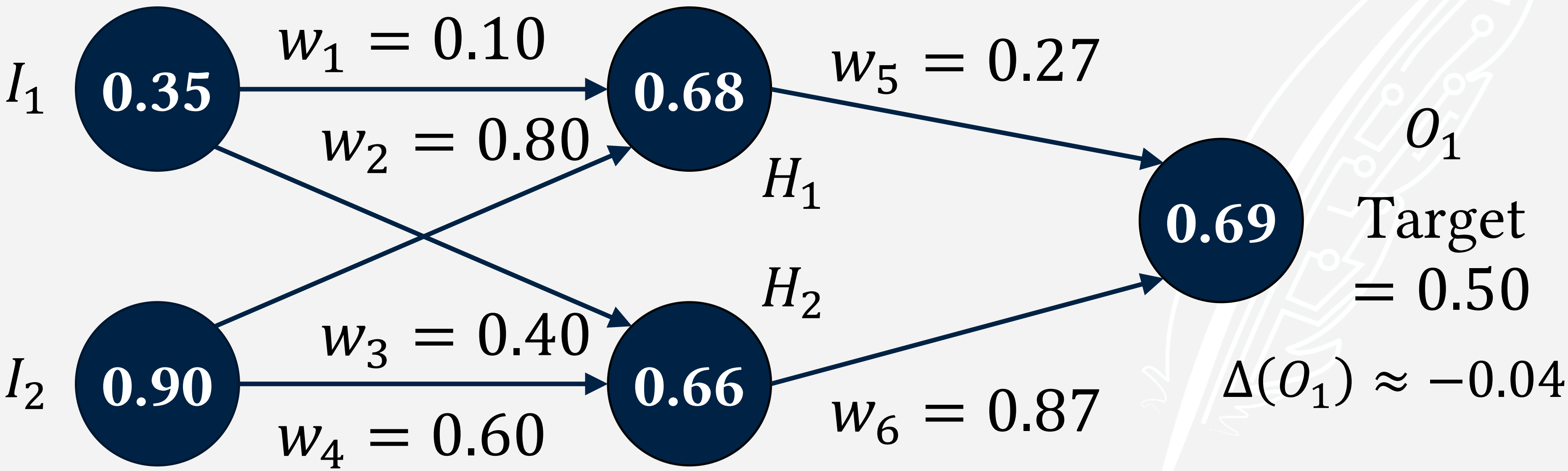
Backpropagation Example

$w_6 = 0.87$



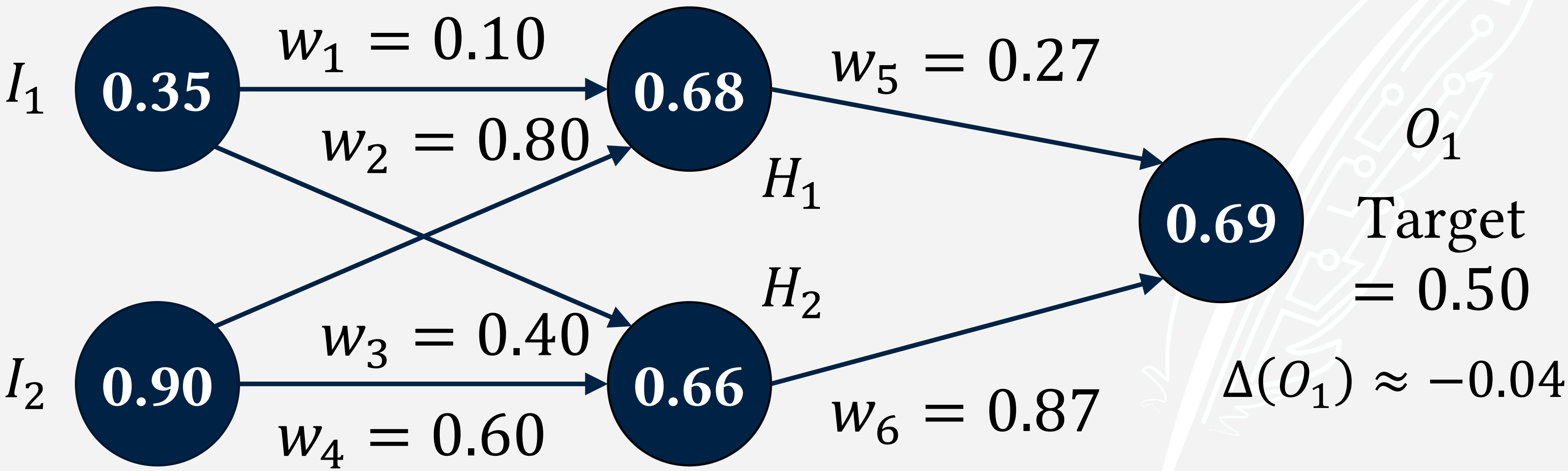
Backpropagation Example

Now we update the hidden layer.



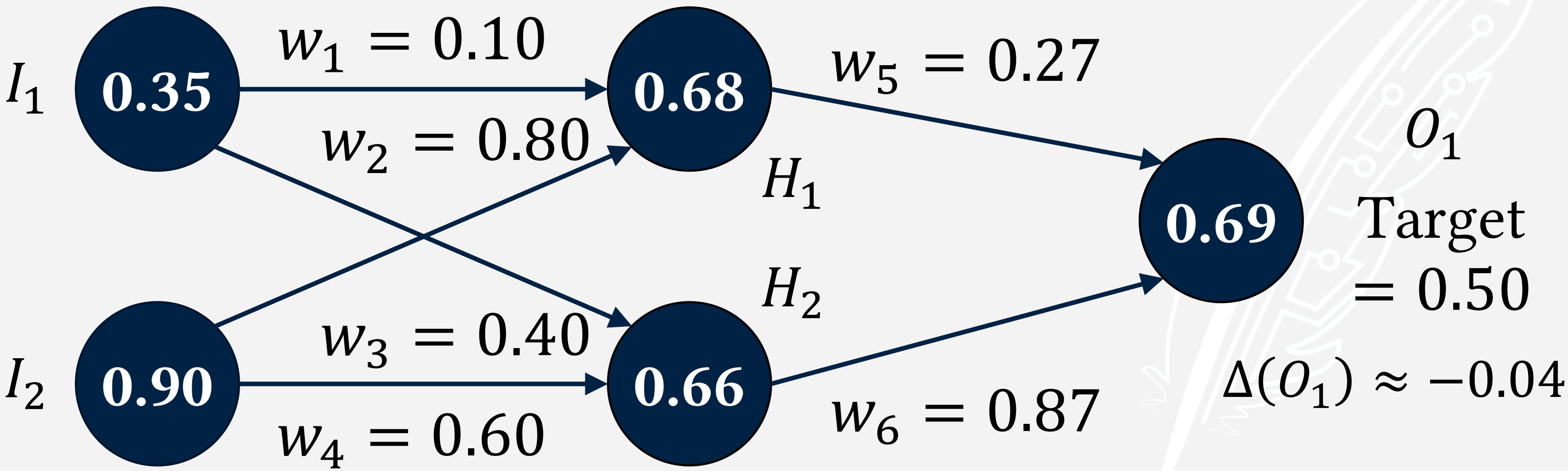
Backpropagation Example

$$\Delta(H_1) = H_1(1 - H_1)\sum_Y w_{H_1 \to Y}\Delta(Y)$$



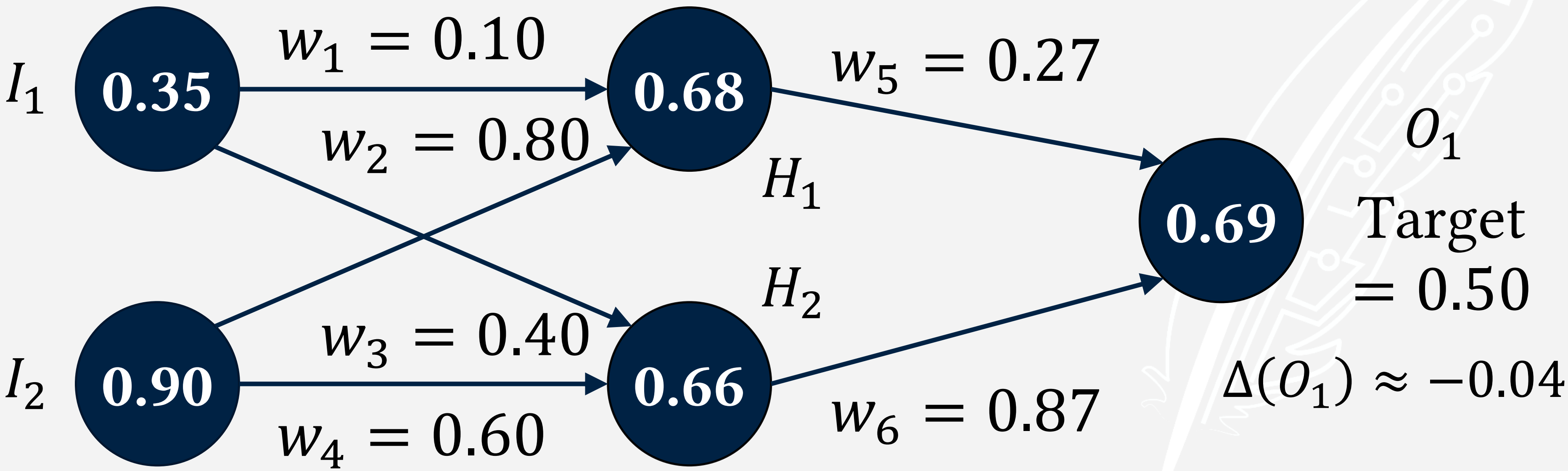
Backpropagation Example

$$\Delta(H_1) = H_1(1 - H_1)\left(w_{H_1 \to O_1} \cdot \Delta(O_1)\right)$$



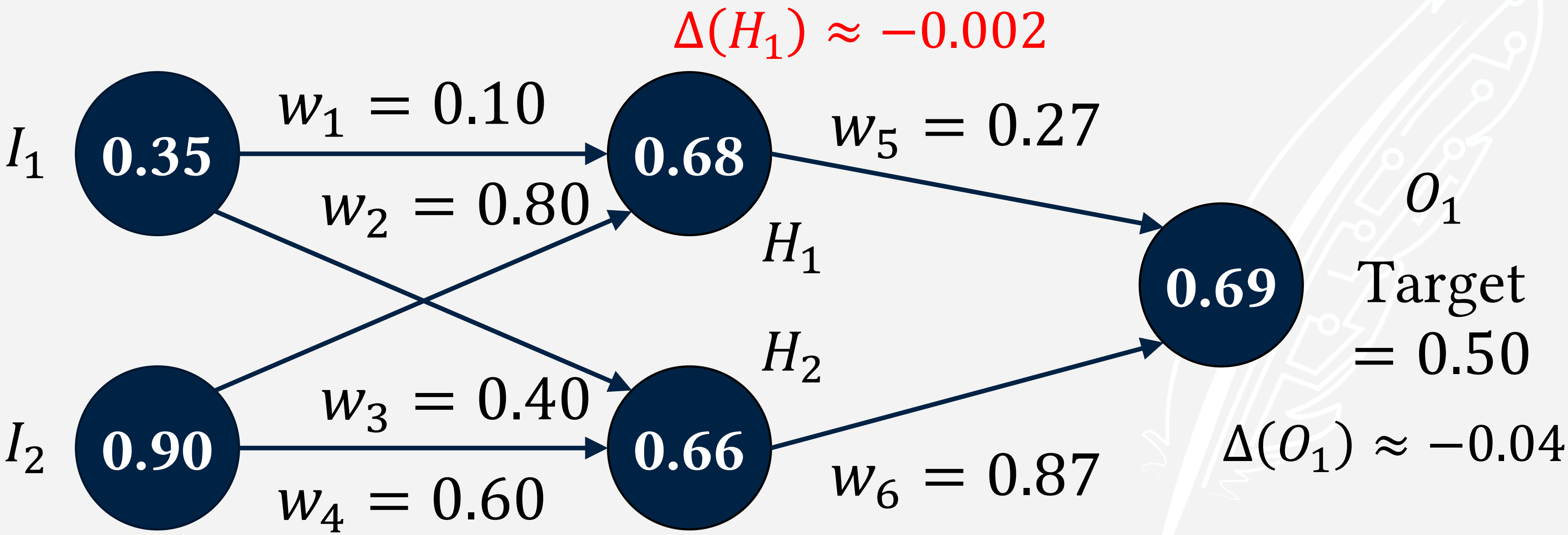
Backpropagation Example

$$\Delta(H_1) = 0.68(1 - 0.68)(0.27 \cdot -0.04)$$



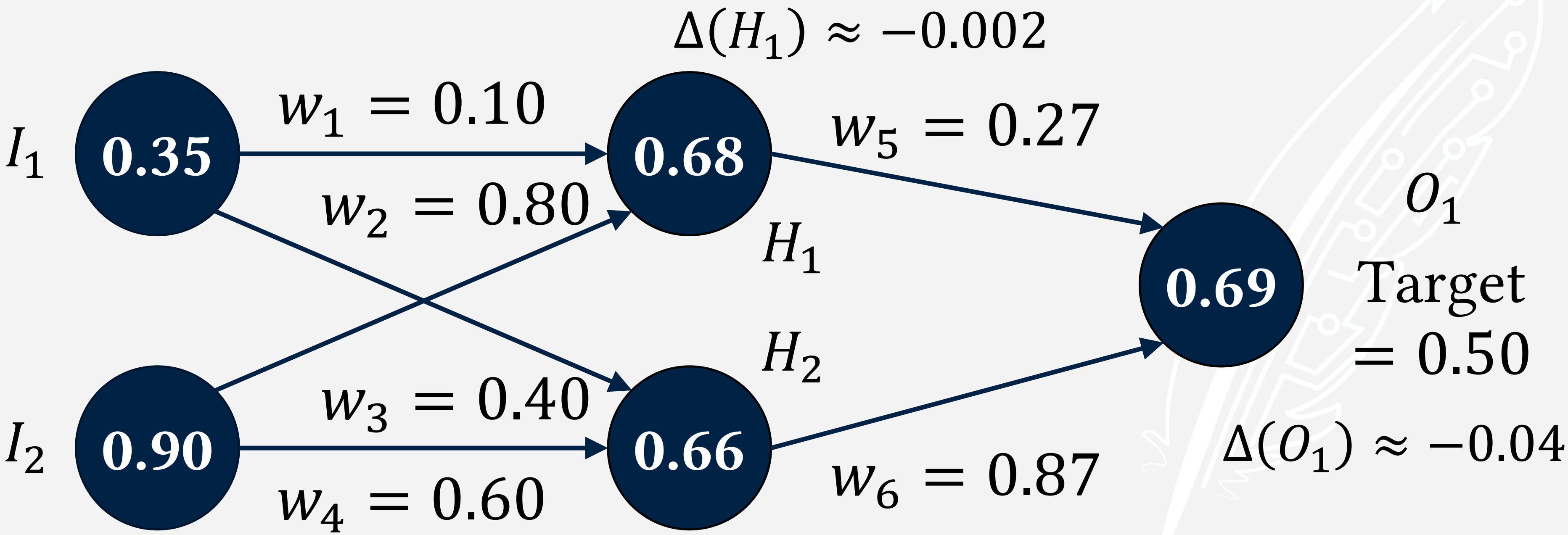
Backpropagation Example

$\Delta(H_1) \approx -0.002$



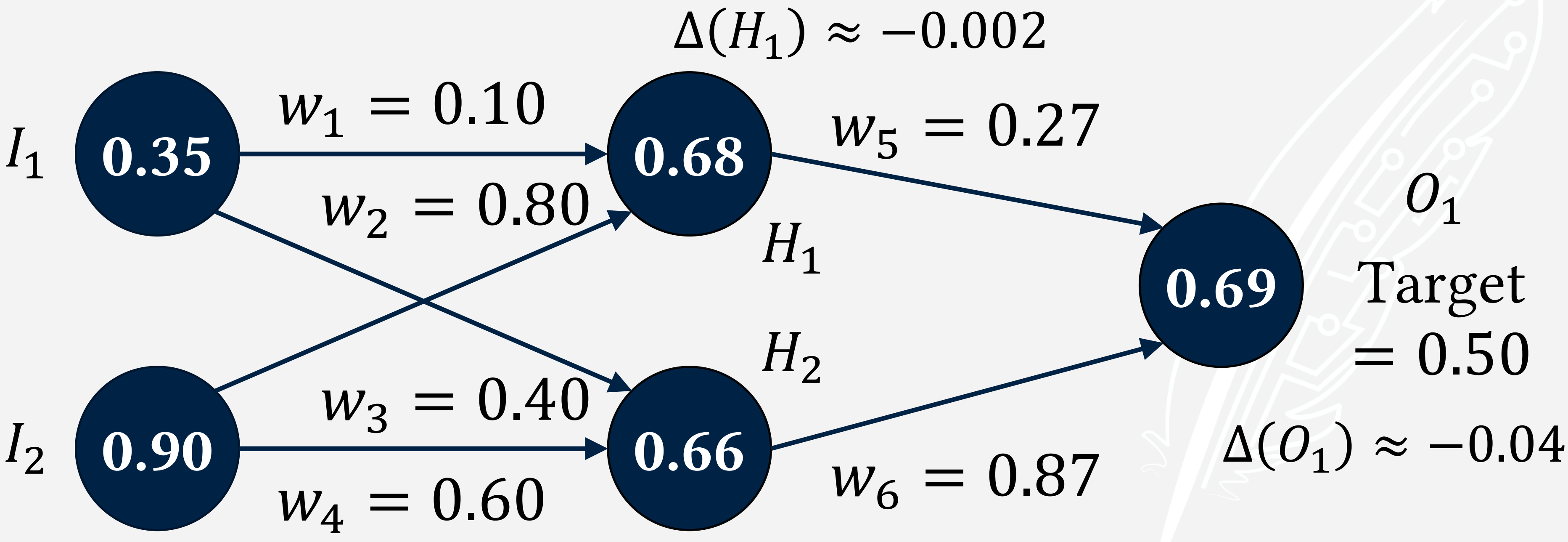
Backpropagation Example

$$\Delta(H_2) = H_2(1 - H_2)\left(w_{H_2 \to O_1} \cdot \Delta(O_1)\right)$$



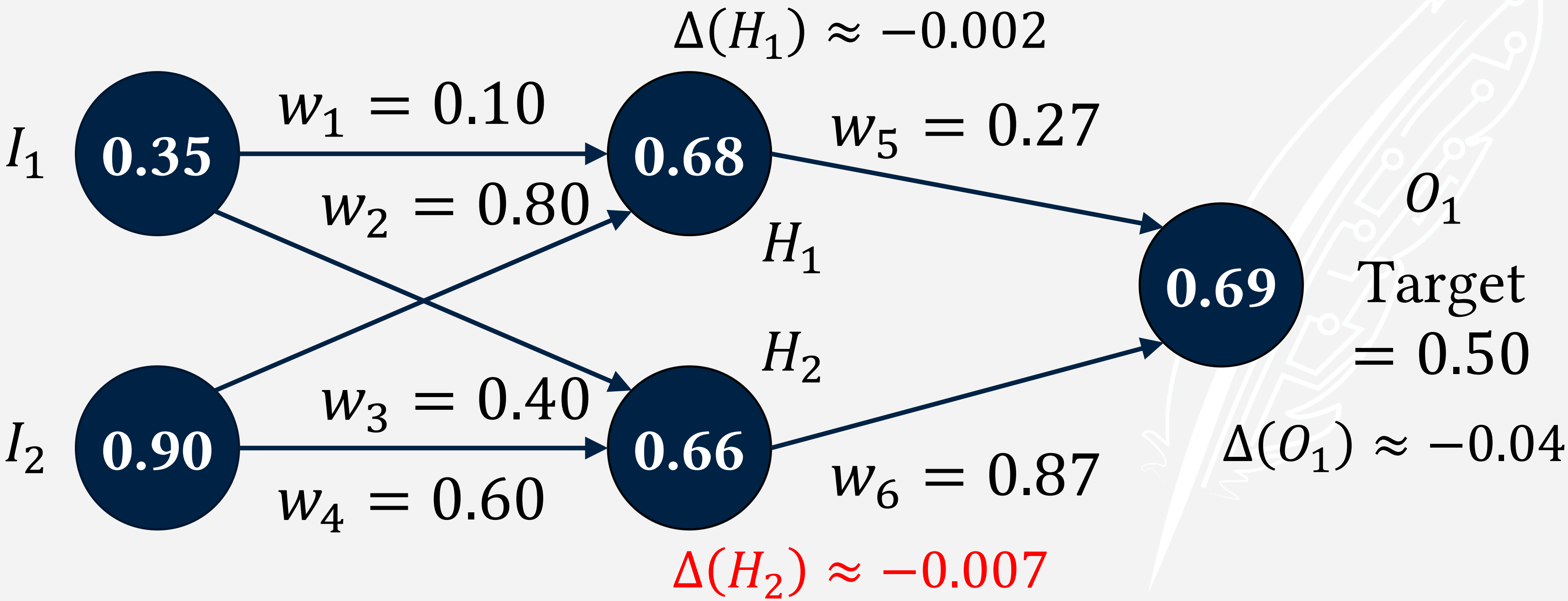
Backpropagation Example

$$\Delta(H_2) = 0.66(1 - 0.66)(0.87 \cdot -0.04)$$



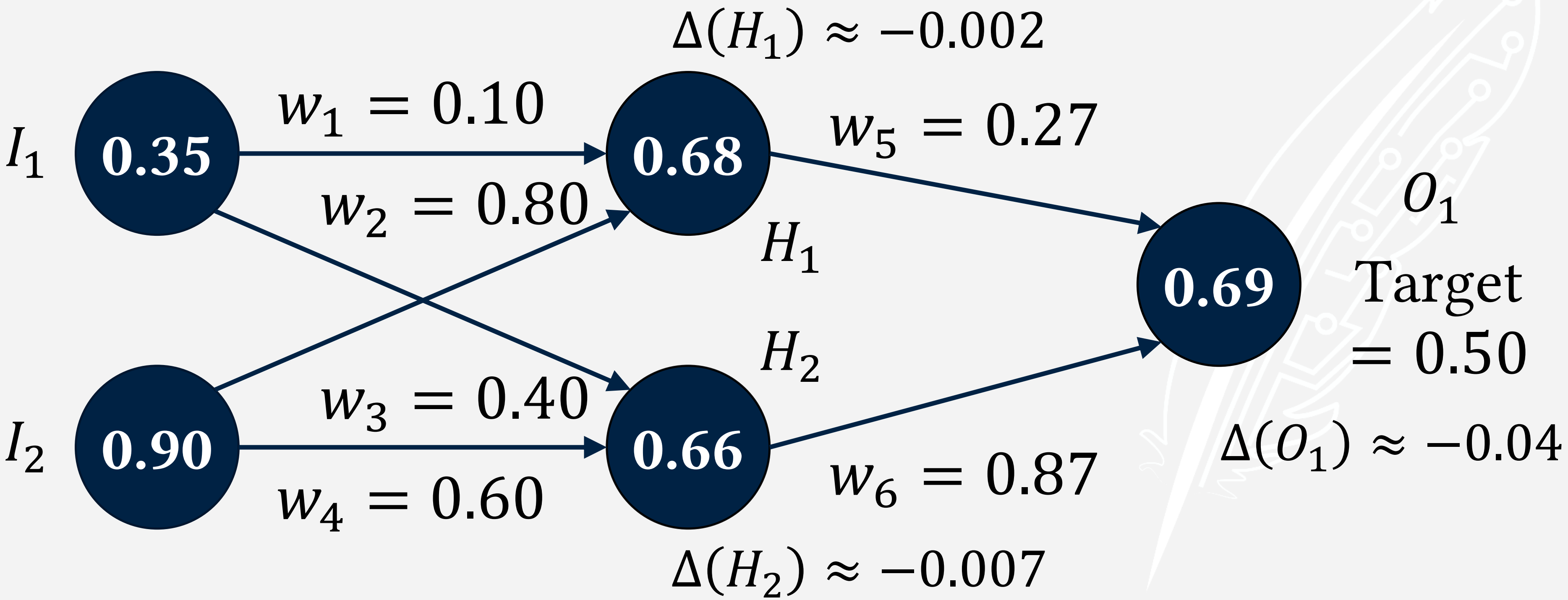
Backpropagation Example

$\Delta(H_2) \approx -0.007$



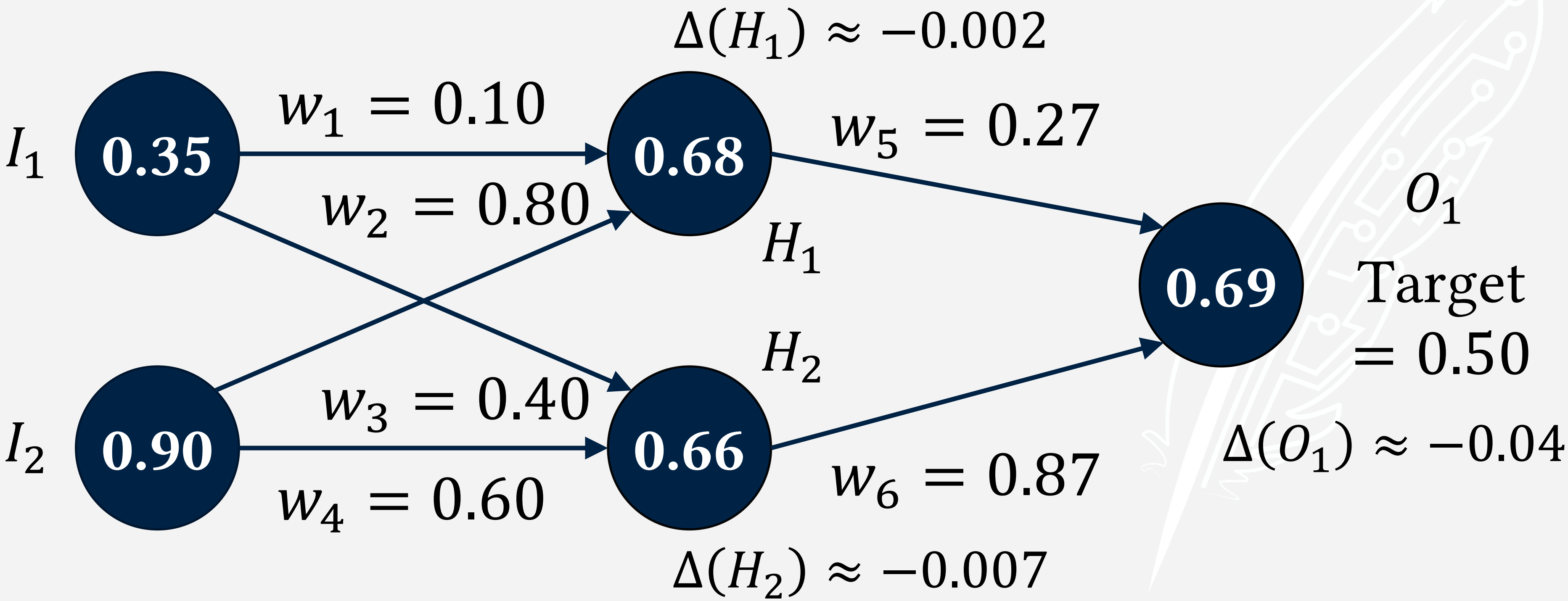
Backpropagation Example

Now we need to update the edges leading from the previous layer to this hidden layer.



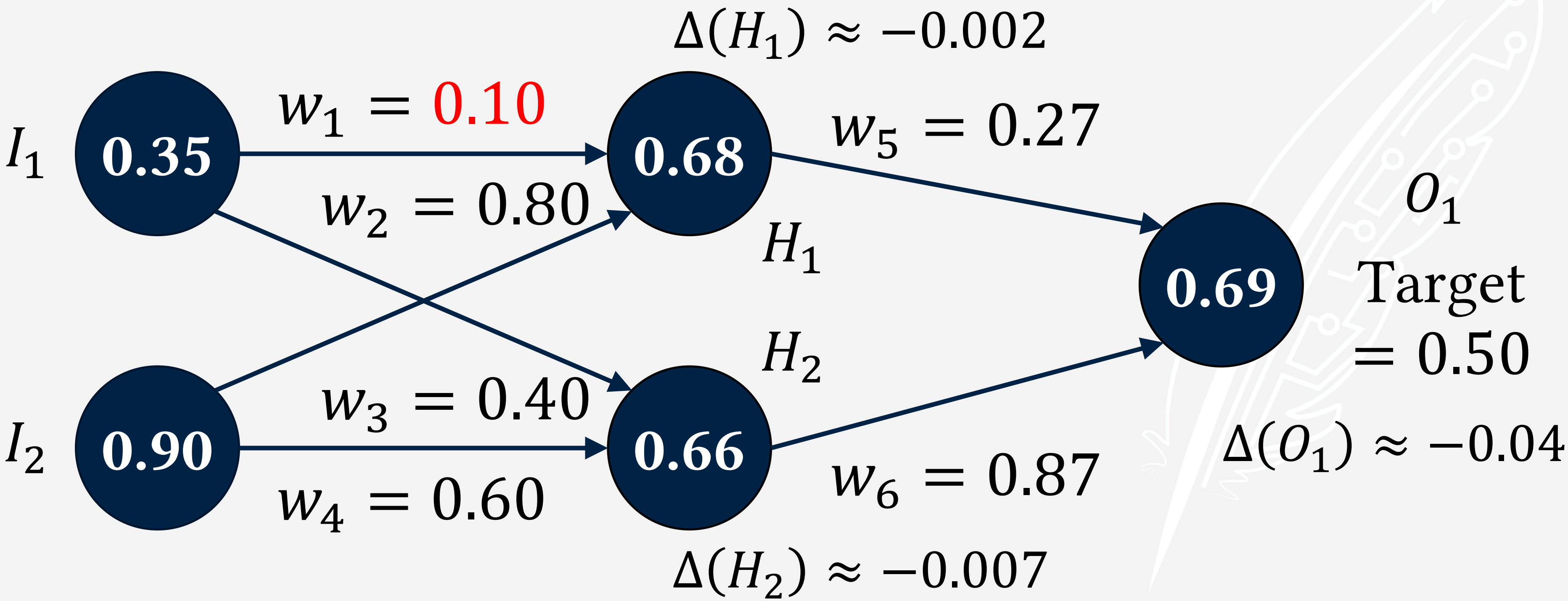
Backpropagation Example

$$w_1 = w_1 + \left(I_1 \cdot \Delta(H_1)\right)$$



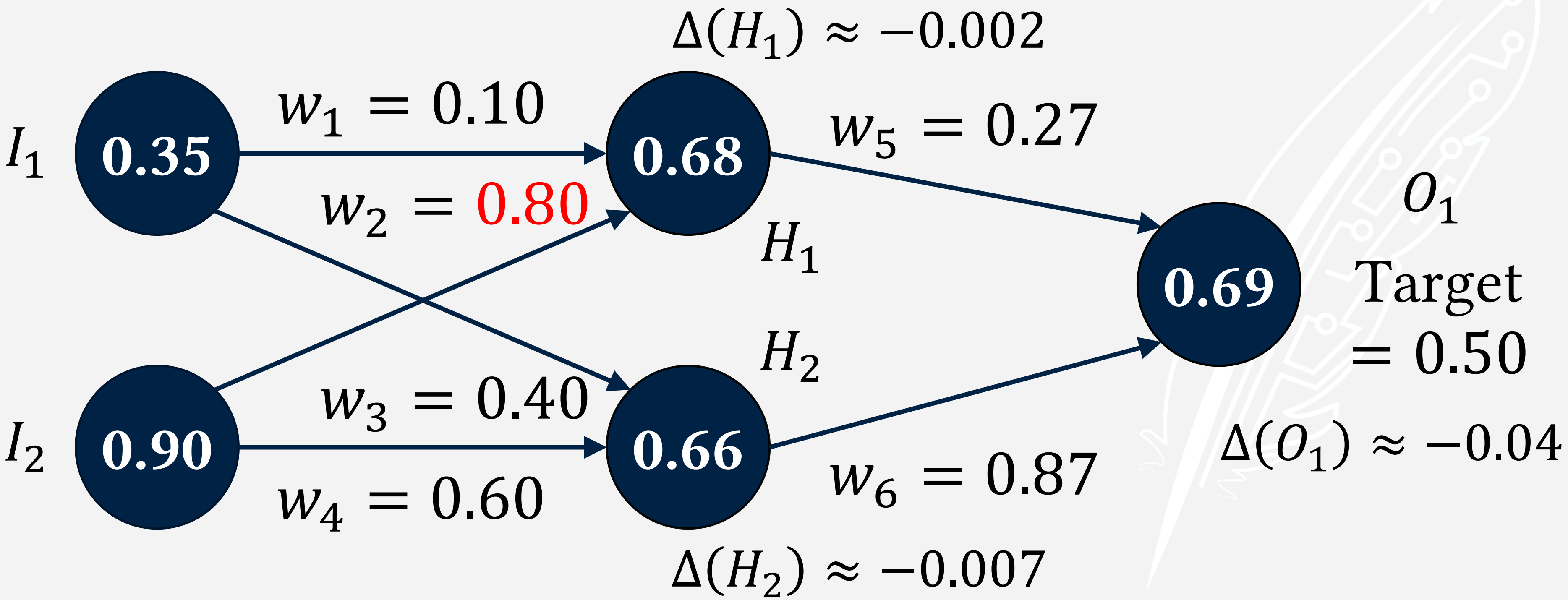
Backpropagation Example

$w_1 \approx 0.10$



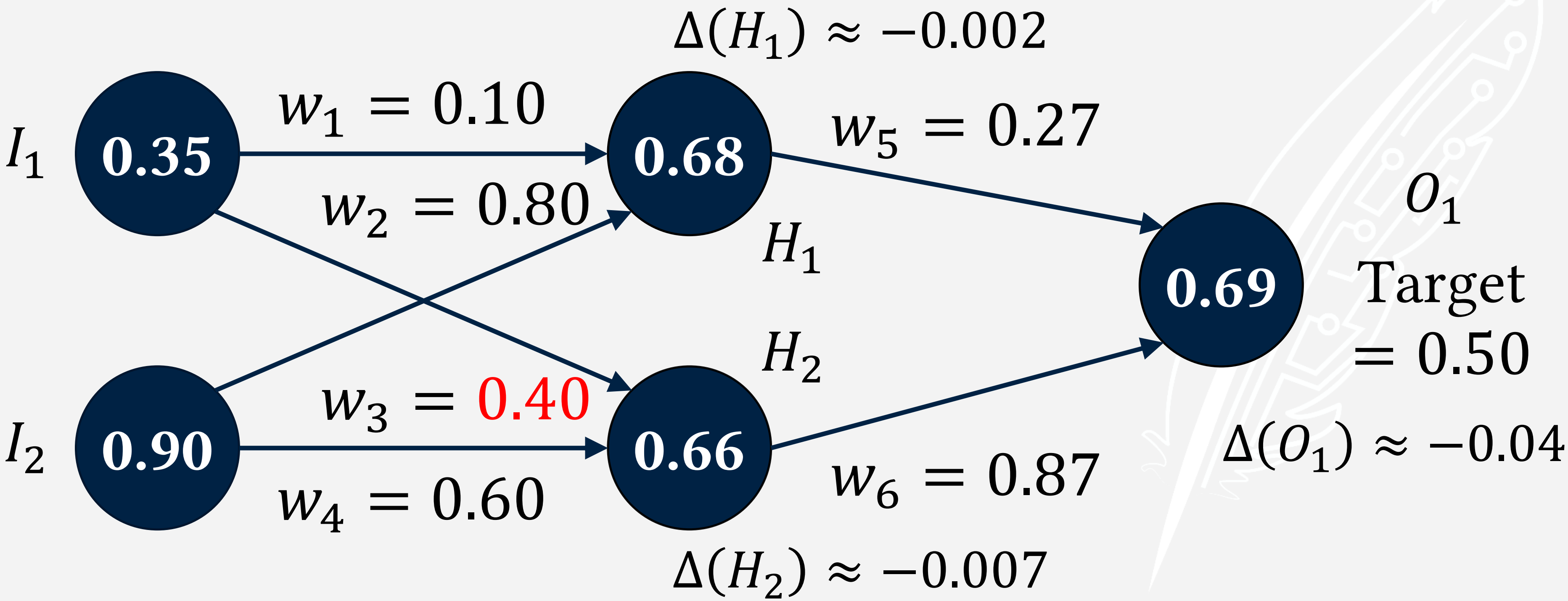
Backpropagation Example

$w_2 \approx 0.80$



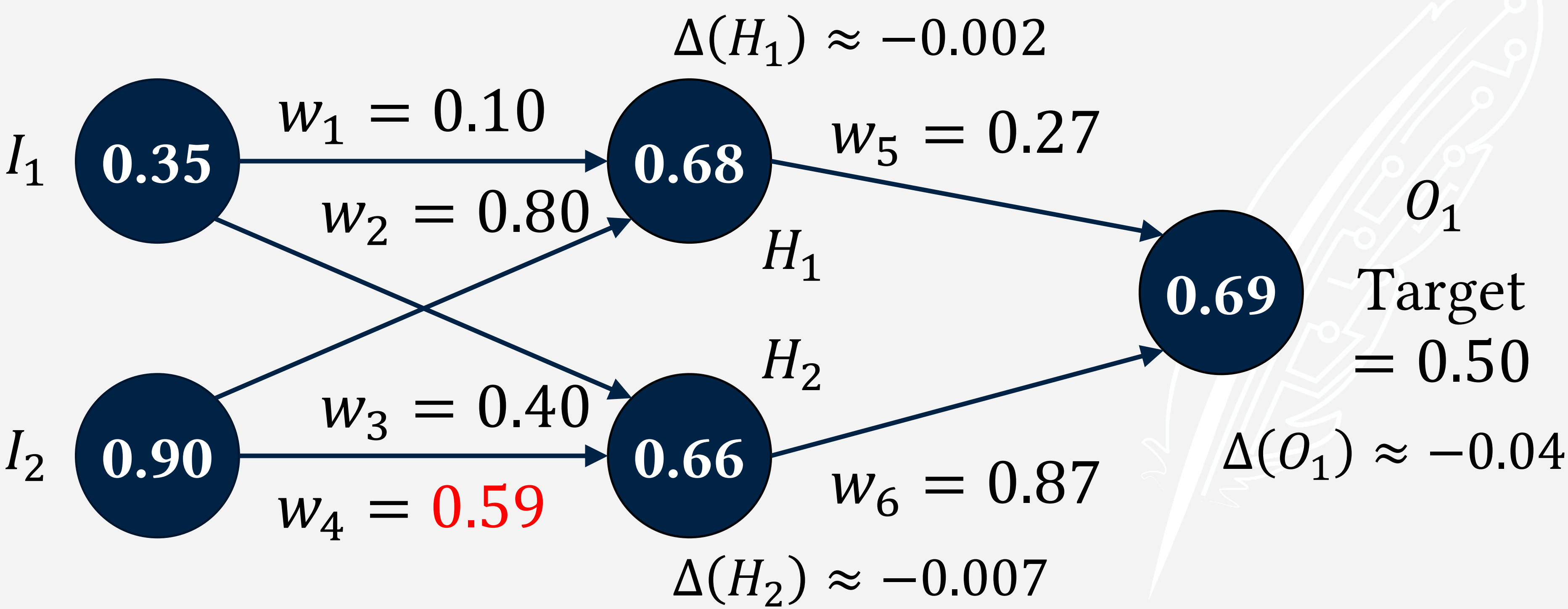
Backpropagation Example

$w_3 \approx 0.40$



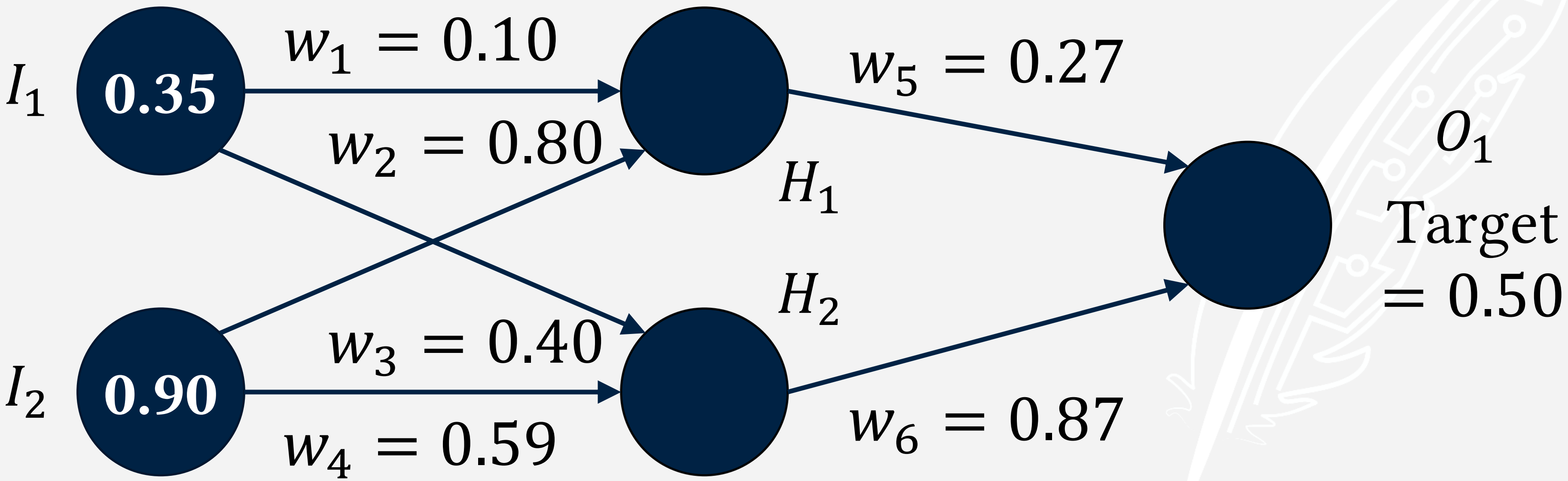
Backpropagation Example

$w_4 \approx 0.59$



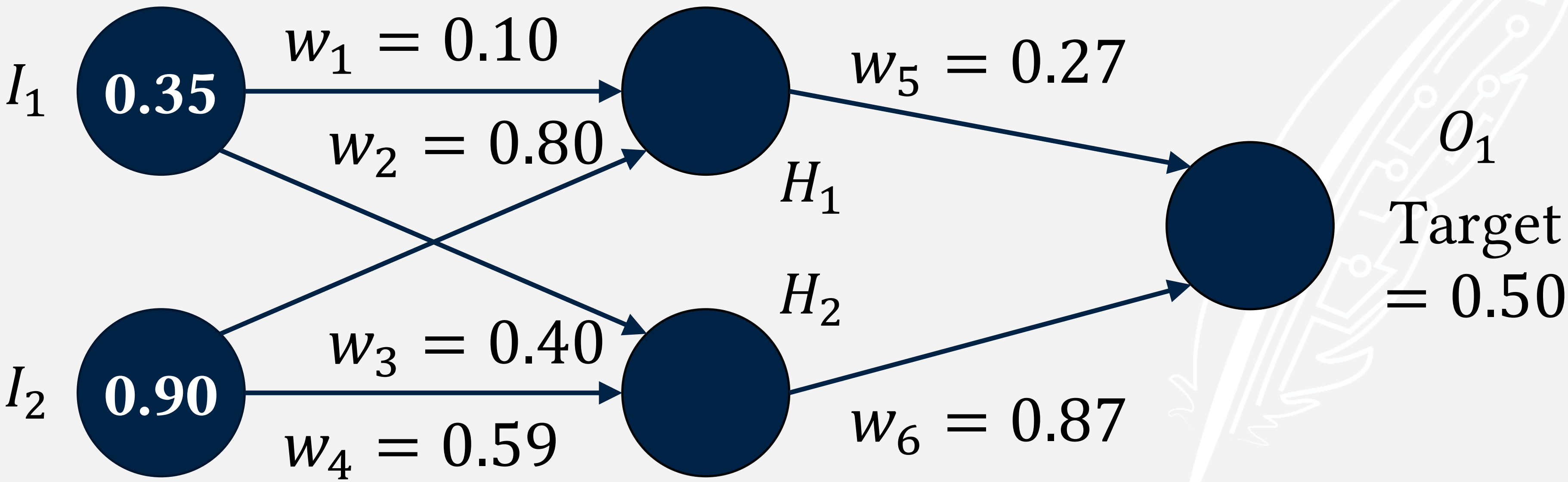
Backpropagation Example

This iteration of back propagation has finished.



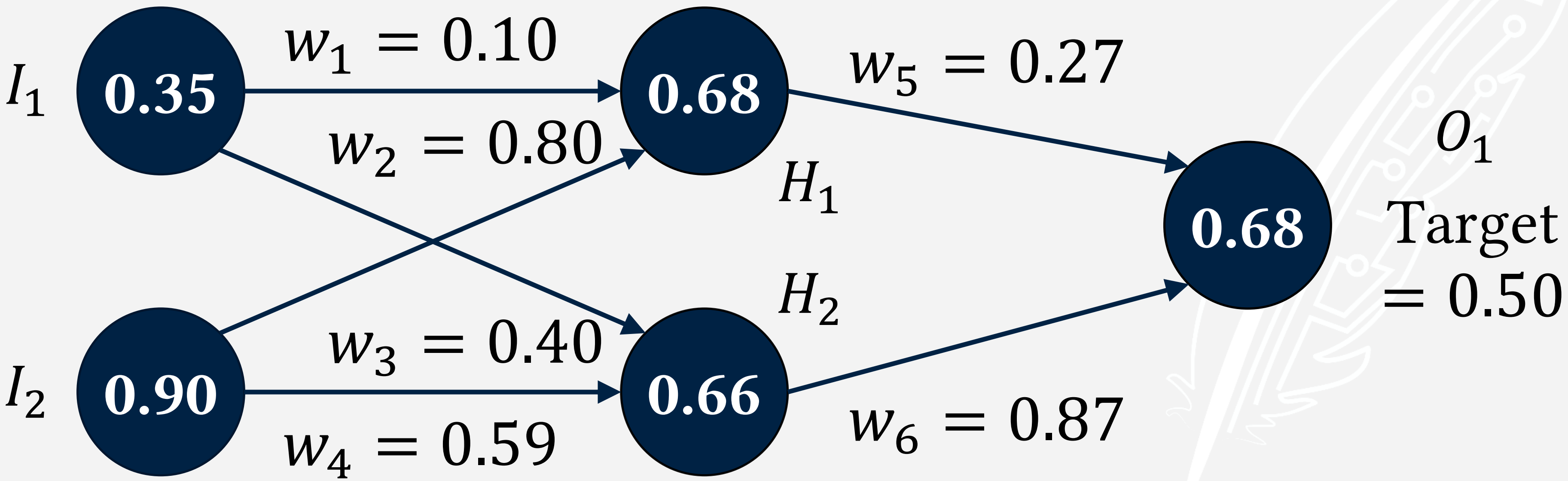
Backpropagation Example

Now we feed values forward again to see what the new output will be.



Backpropagation Example

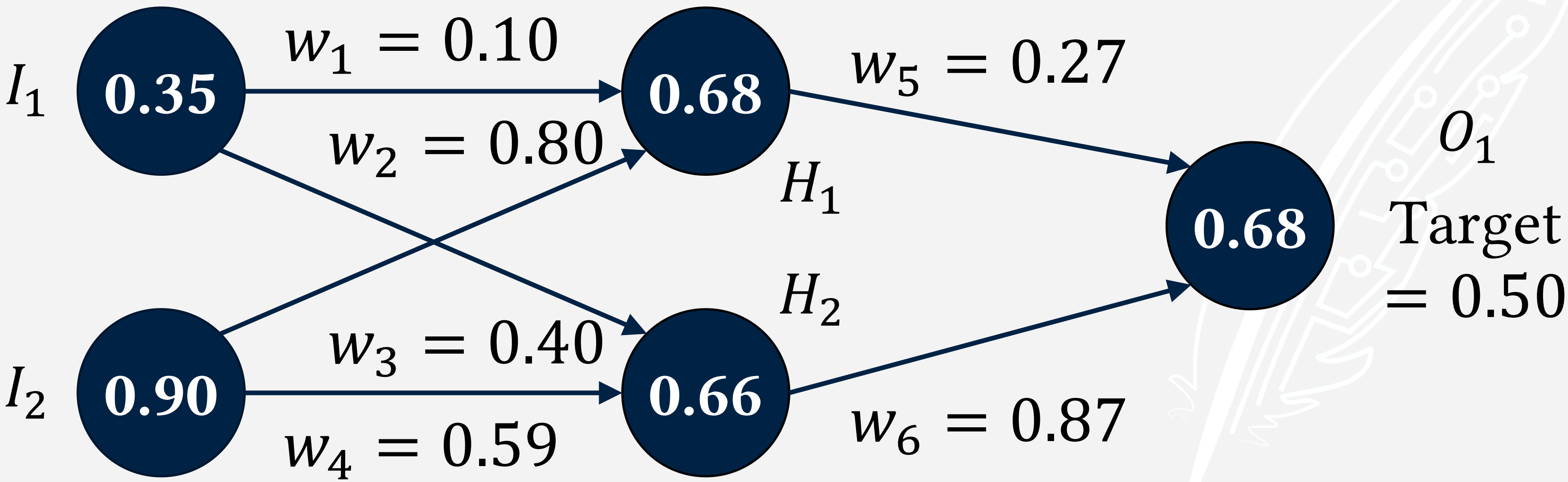
Now we feed values forward again to see what the new output will be.



Backpropagation Example

The previous error was $0.50 - 0.69 = -0.19$.

The new error is $0.50 - 0.68 = -0.18$.



Backpropagation Example

Error has been reduced! We then repeat back propagation until we have reduced it as much as possible.