Intelligent Data Analytics

Introduction to Deep Learning

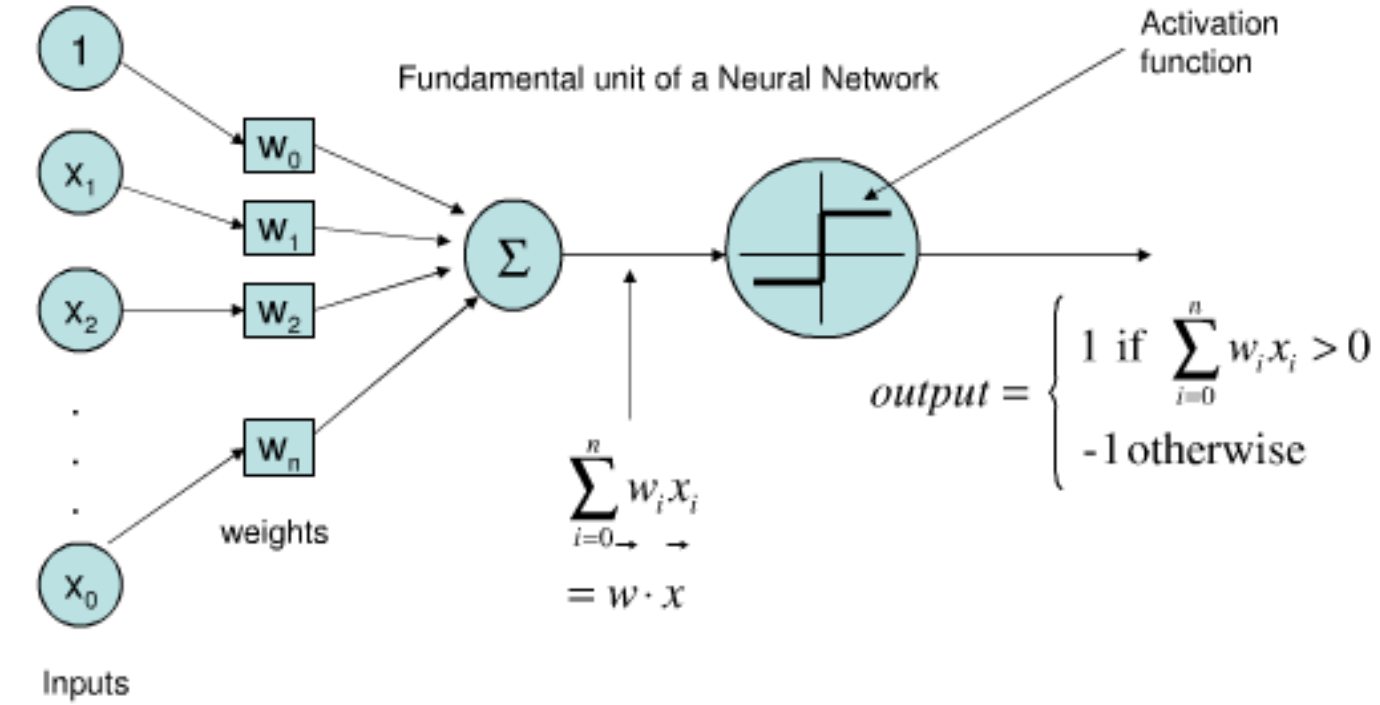
Dr. Nguyen Tien Huy

ntienhuy@fit.hcmus.edu.vn

Nội dung

1. Perceptron
2. Multi-layer neural network
3. Backpropagation
4. Deep learning with Keras + Tensorflow

Perceptron

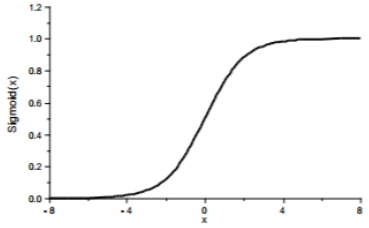

1
x_1
x_2
x_0
Inputs
w_0
w_1
w_2
w_n
weights
Σ
Fundamental unit of a Neural Network
Activation function
$\sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x}$
$output = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

Other activation functions

Sigmoid Functions These are smooth (differentiable) and monotonically increasing.

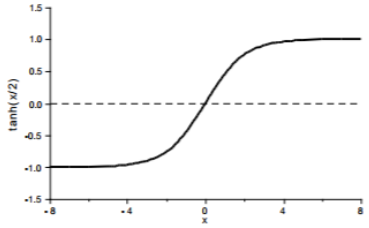
The logistic function

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



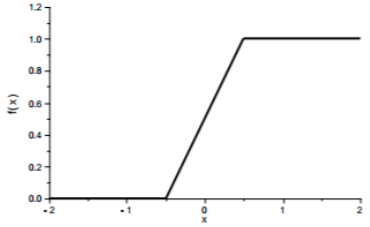
Hyperbolic tangent

$$\tanh\left(\frac{x}{2}\right) = \frac{1-e^{-x}}{1+e^{-x}}$$



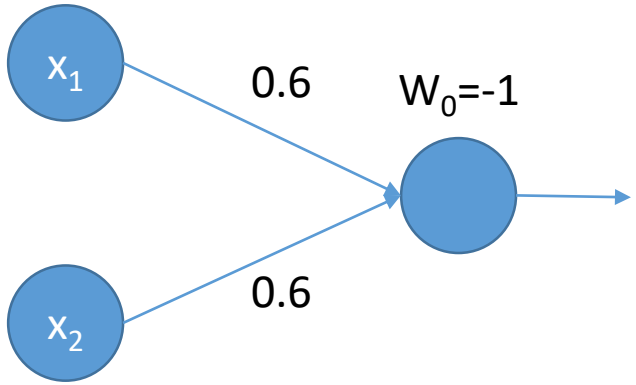
Piecewise-Linear Functions Approximations of a sigmoid functions.

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0.5 \\ x + 0.5 & \text{if } -0.5 \leq x \leq 0.5 \\ 0 & \text{if } x \leq 0.5 \end{cases}$$



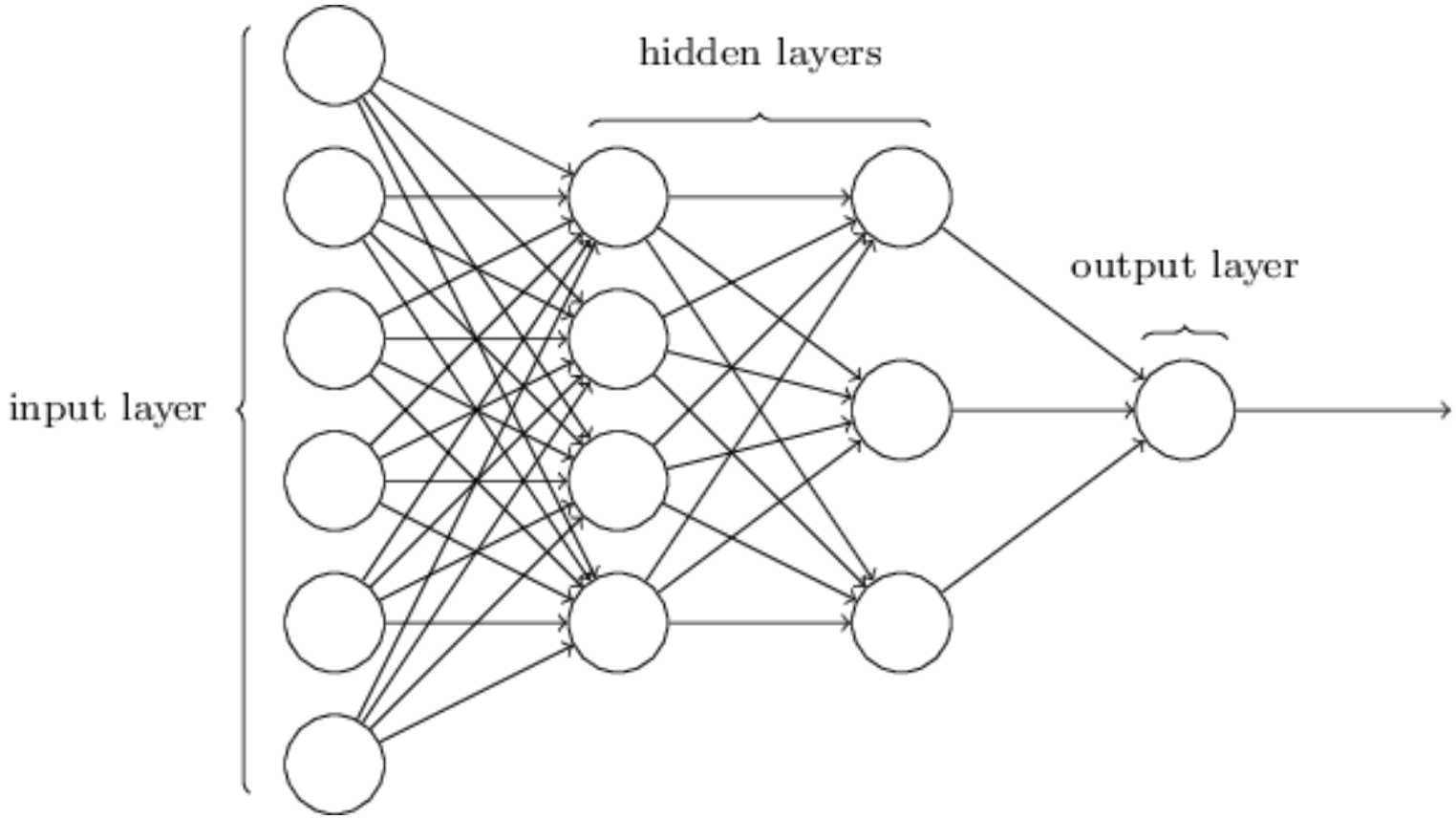
Example for AND gate

x_1	x_2	ouput
0	0	-1
0	1	-1
1	0	-1
1	1	1



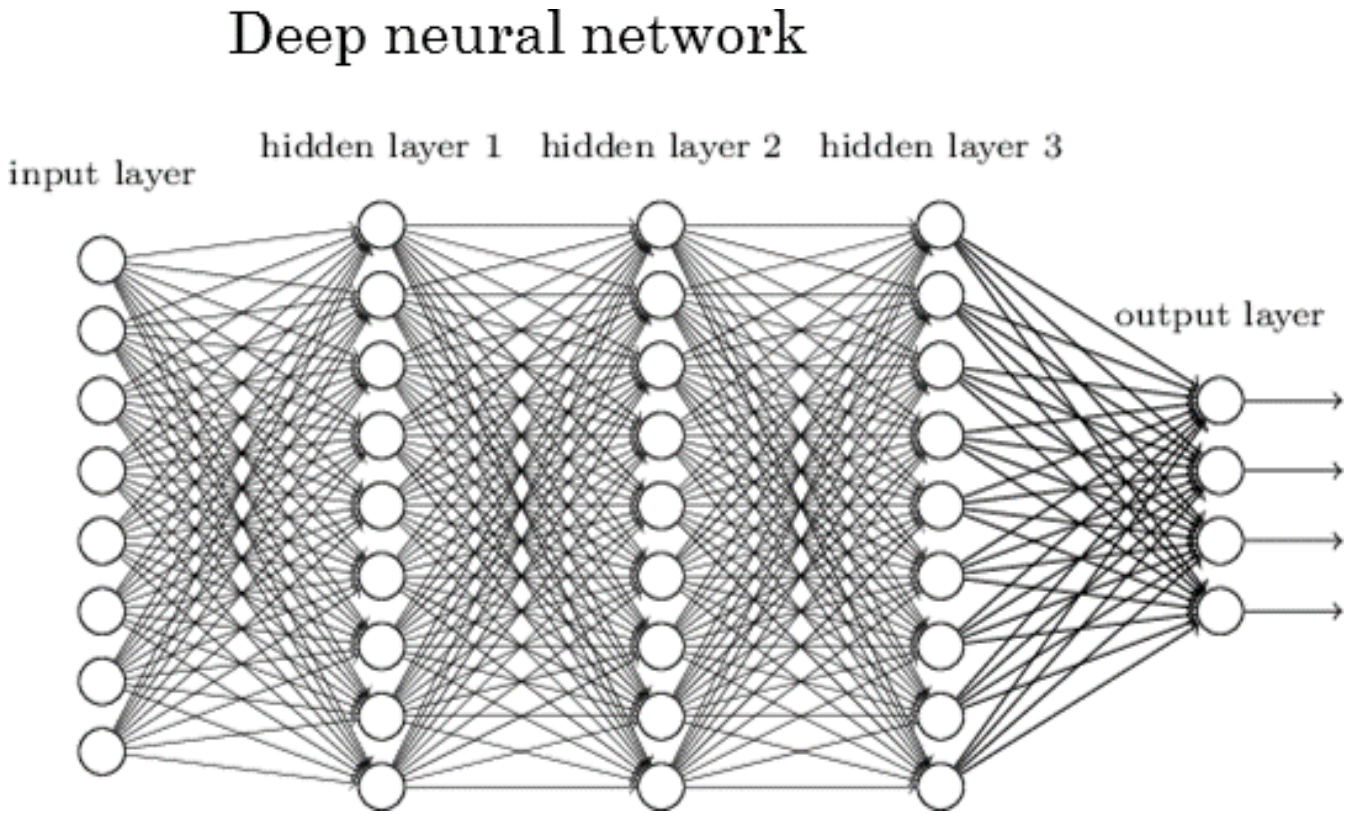
How to choose correctly the values of w_0, w_1, w_2??? => This process is called “training”

The neural network



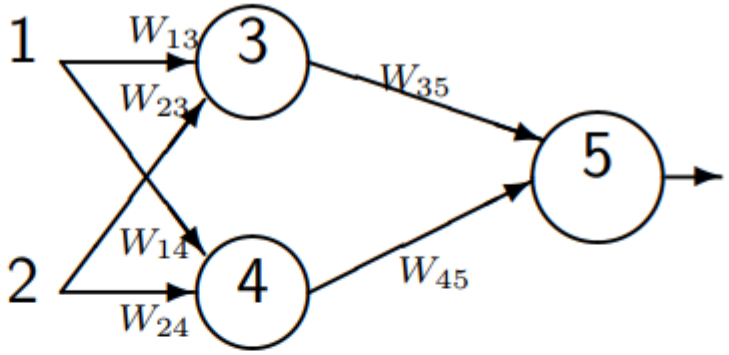
Deep learning

- A simple definition: a neural network with many hidden layers.
- AutoML



Example

Example 3.



$w_{13} = 2$ $w_{23} = -3$	$w_{35} = 2$
$w_{14} = 1$ $w_{24} = 4$	$w_{45} = -1$

$$f(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

What is the network output, if the inputs are $x_1 = 1$ and $x_2 = 0$?

Example

- Forward

Solution

1. Calculate weighted sums in the first hidden layer:

$$\begin{aligned} v_3 &= w_{13}x_1 + w_{23}x_2 = 2 \cdot 1 - 3 \cdot 0 = 2 \\ v_4 &= w_{14}x_1 + w_{24}x_2 = 1 \cdot 1 + 4 \cdot 0 = 1 \end{aligned}$$

2. Apply the activation function:

$$y_3 = f(2) = 1\ , \qquad y_4 = f(1) = 1$$

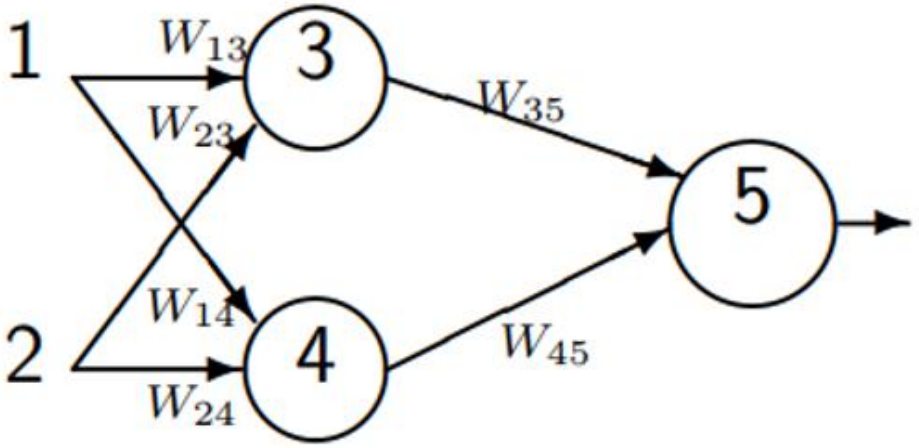
3. Calculate the weighted sum of node 5:

$$v_5 = w_{35}y_3 + w_{45}y_4 = 2 \cdot 1 - 1 \cdot 1 = 1$$

4. The output is $y_5 = f(1) = 1$

Training

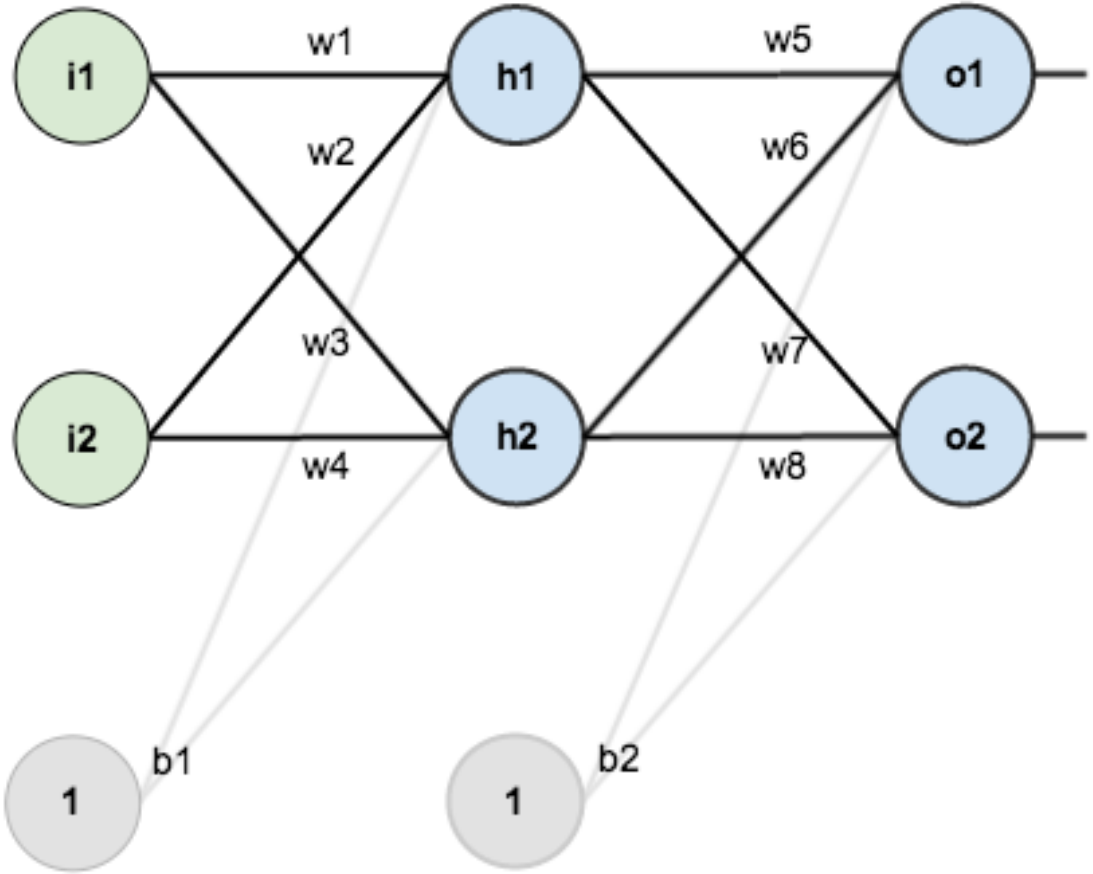
- If with $x_1=1$, $_{x2}=0$ and $y=0$, and with $x_1=0$, $x_2=0$ and $y=1$, so W?



Backpropagation

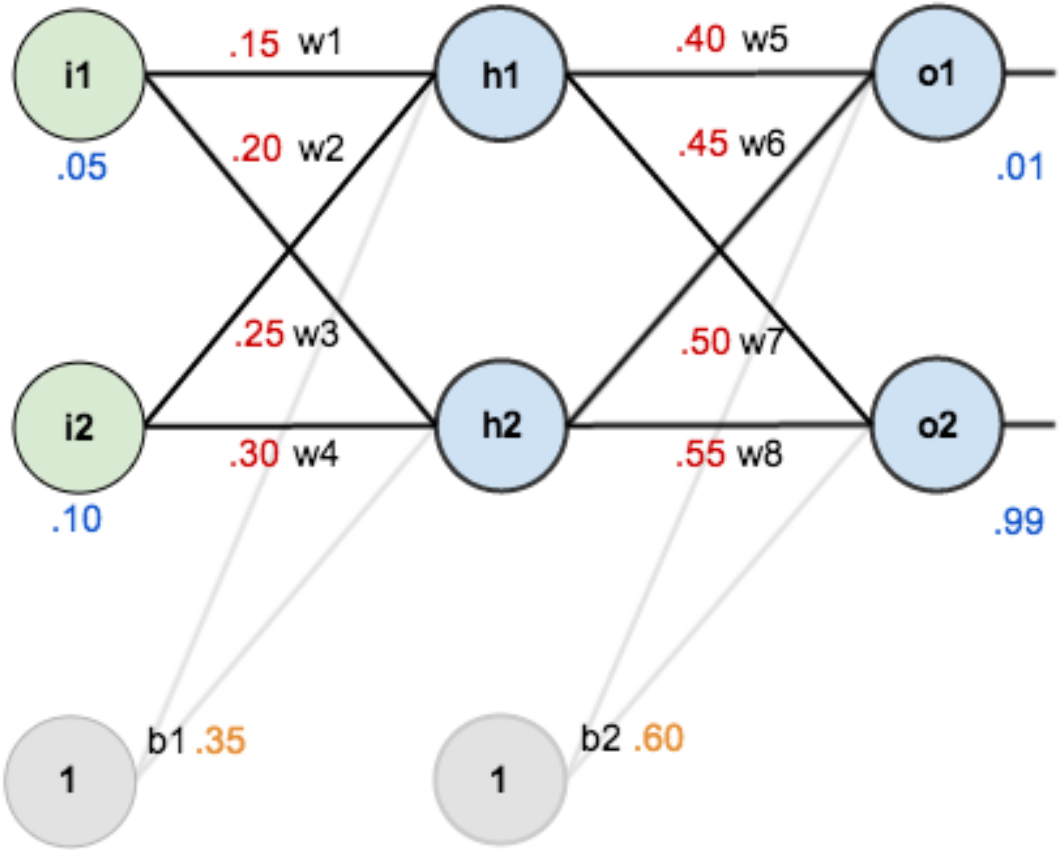
- Given inputs 0.05 and 0.10, we want the neural network to output 0.01 and 0.99.

- Activation function is sigm $\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$



Backpropagation

Initiate the weights and biases



Backpropagation -

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

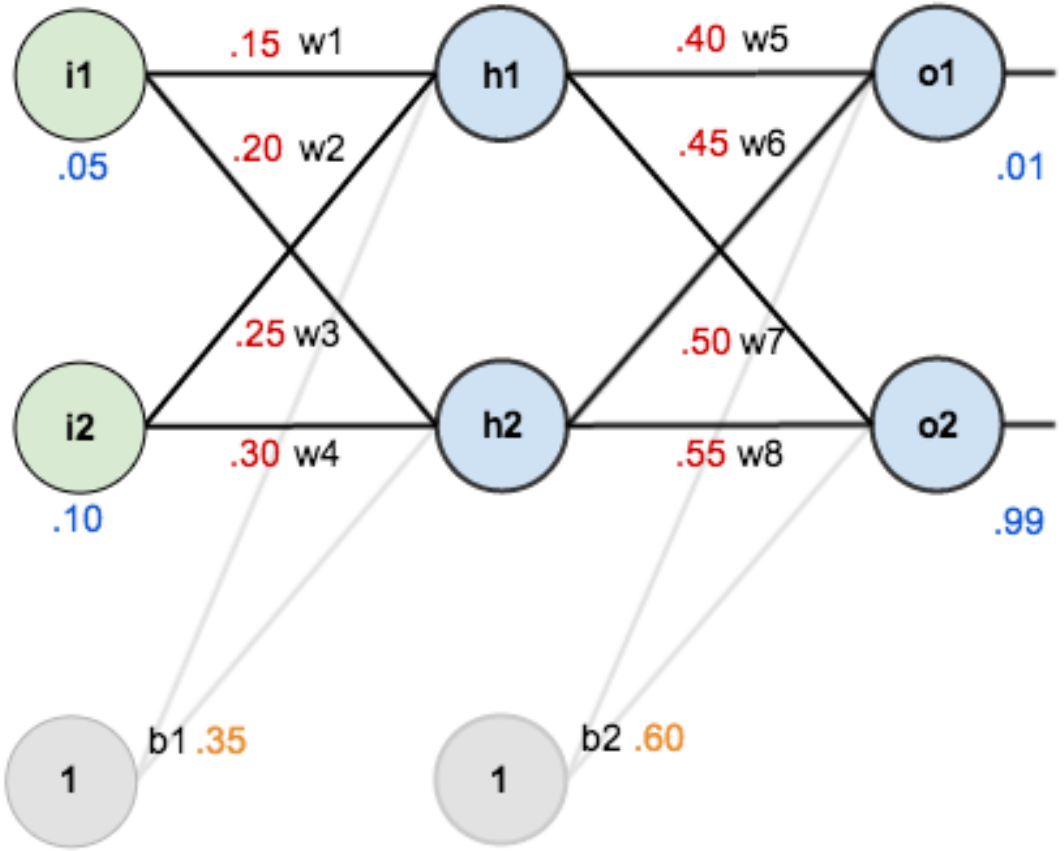
$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

We then squash it using the logistic function to get the output of h_1:

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

Carrying out the same process for h_2 we get:

$$out_{h2} = 0.596884378$$



Backpropagation

Here's the output for o_1:

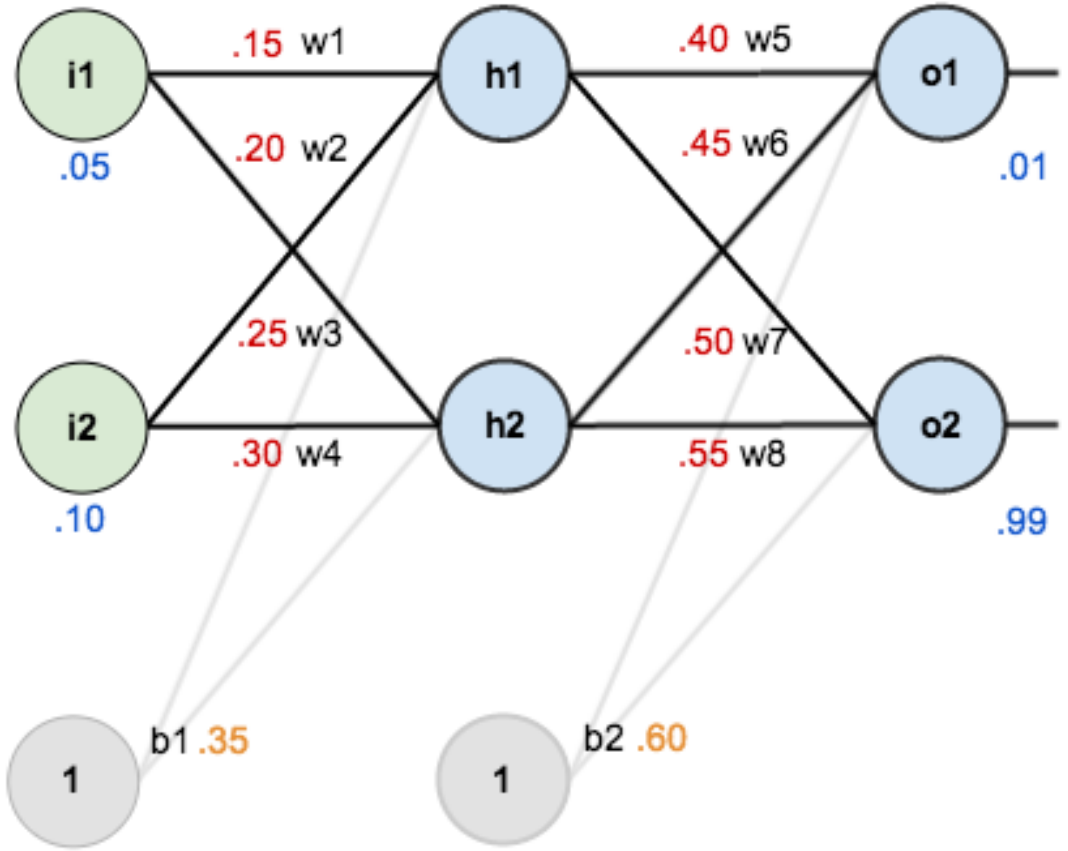
$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

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

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

And carrying out the same process for o_2 we get:

$$out_{o2} = 0.772928465$$



Backpropagation – Error, loss function, obtimization

Calculate the error:

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

For example, the target output for o_1 is 0.01 but the neural network output 0.75136507, therefore its error is:

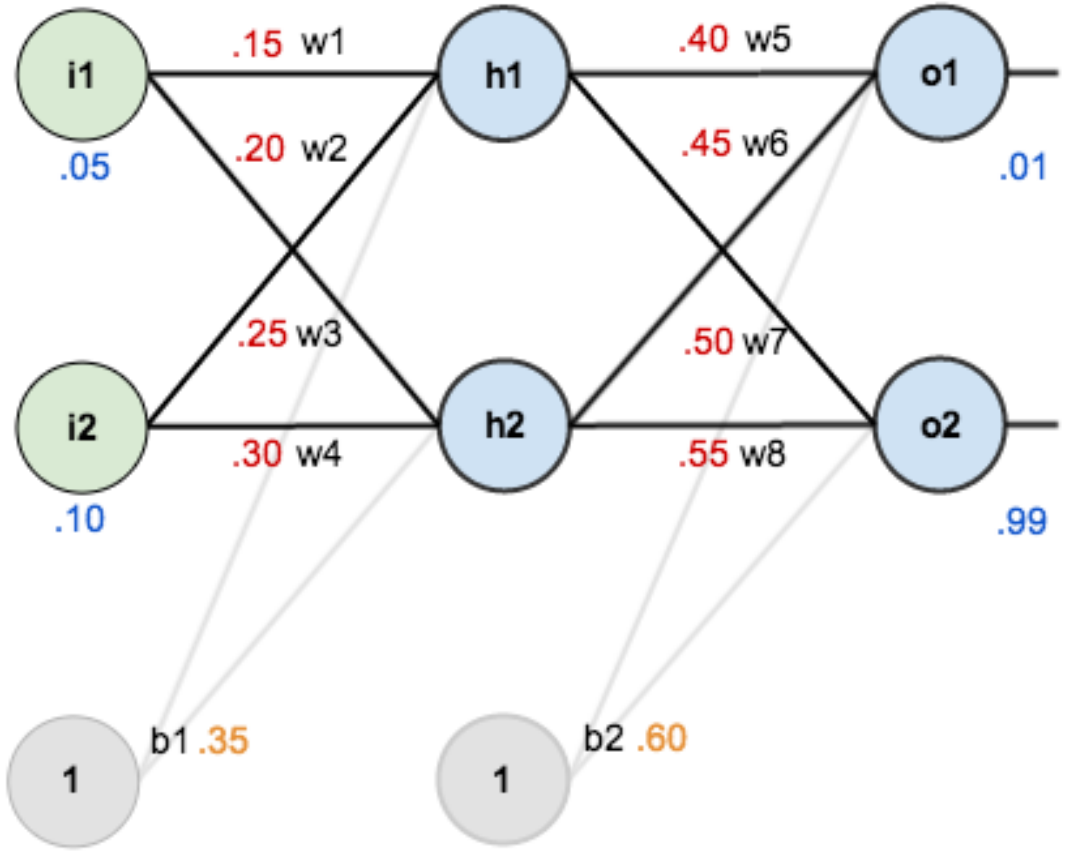
$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

$$E_{o2} = 0.023560026$$

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

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



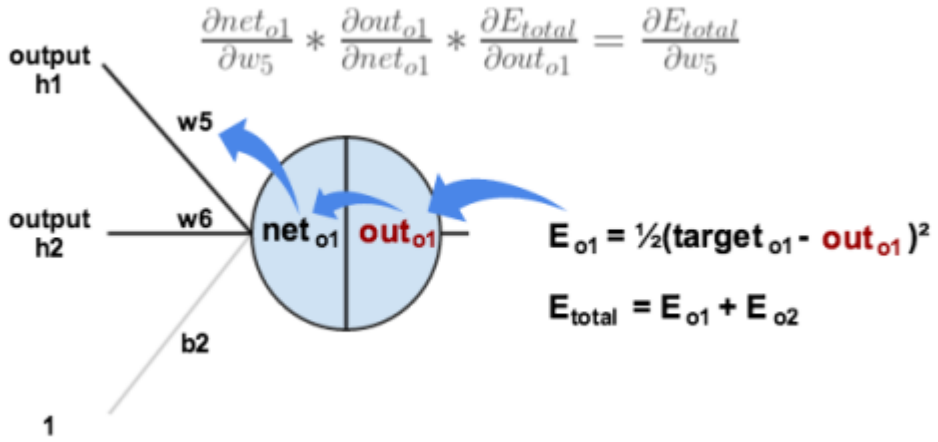
Backpropagation

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:

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



Backpropagation

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

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

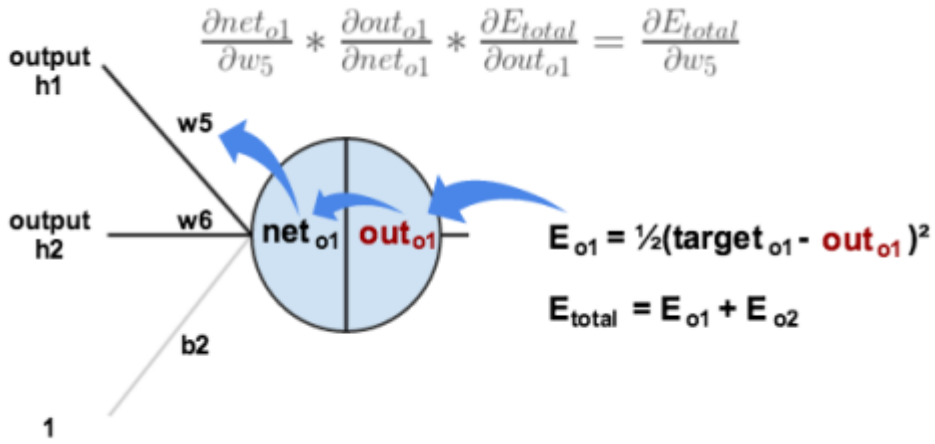
$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

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

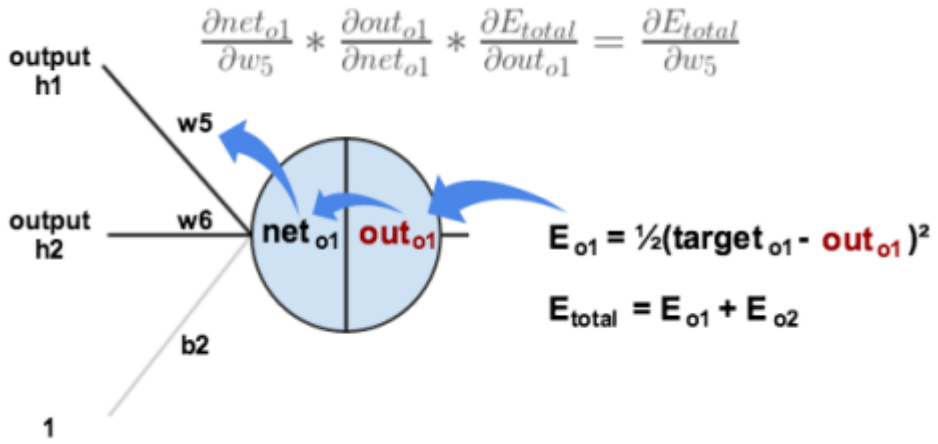
$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

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

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



Backpropagation



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, eta, which we'll set to 0.5):

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

Backpropagation

You'll often see this calculation combined in the form of the delta rule:

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

Alternatively, we have $\frac{\partial E_{total}}{\partial out_{o1}}$ and $\frac{\partial out_{o1}}{\partial net_{o1}}$ which can be written as $\frac{\partial E_{total}}{\partial net_{o1}}$, aka δ_{o1} (the Greek letter delta) aka the *node delta*. We can use this to rewrite the calculation above:

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

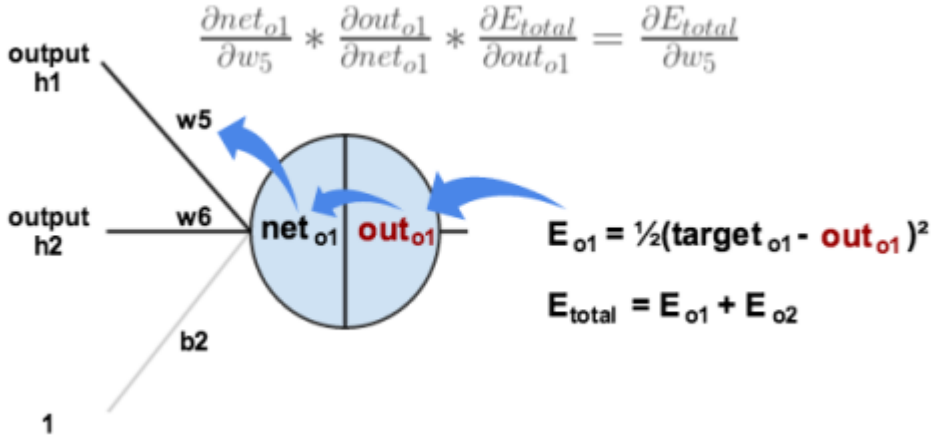
$$\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

Therefore:

$$\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} out_{h1}$$

Some sources extract the negative sign from δ so it would be written as:

$$\frac{\partial E_{total}}{\partial w_5} = -\delta_{o1} out_{h1}$$

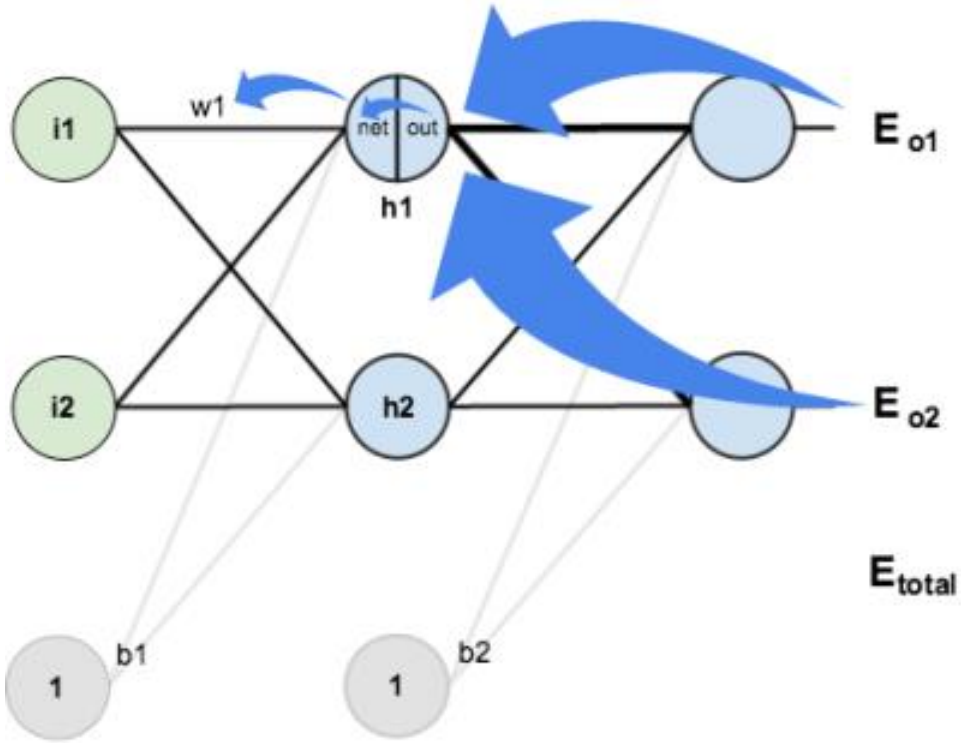


Backpropagation

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

$$\downarrow$$

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



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

Backpropagation

- W1 = 0.149780716
- W2 = 0.19956143
- W3 = 0.24975114
- W4 = 0.29950229
- W5 = 0.35891648
- W6 = 0.408666186
- W7 = 0.51130127
- W8 = 0.561370121

=> Repeat this training process (epoch) until the error get small enough.

Practice with Dense (Fully-connected)

- Pytorch: https://colab.research.google.com/drive/1m0A2oglbfQNQXo94xzgcUusKALf7eFpt?usp=sharing

- Keras: https://colab.research.google.com/drive/1h-mNrQDmdUy1IDwY_QaO4uFKqd3EGrgI?usp=sharing

Summary

- Deep learning: a network of perceptrons

- Training a deep learning model via Backpropagation

Thanks for your attention!