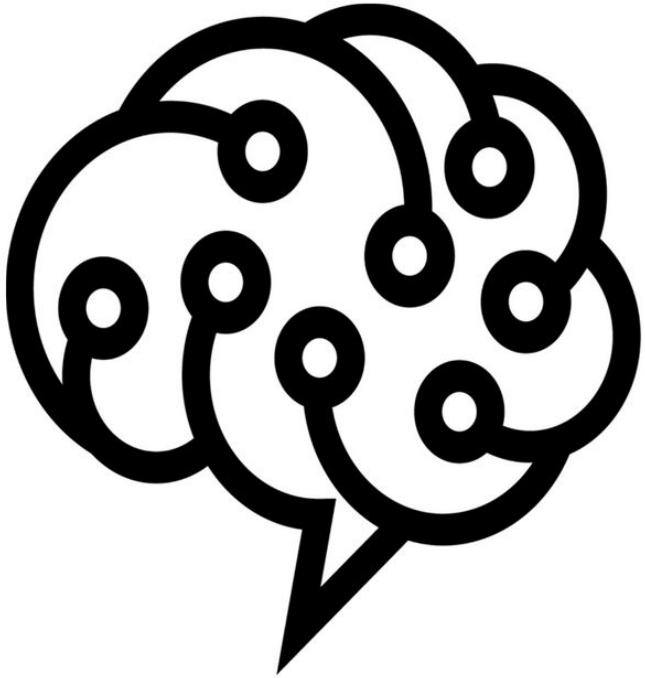
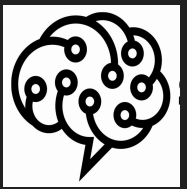


school of ai

Fundamentals of Artificial Neural Networks

3/3



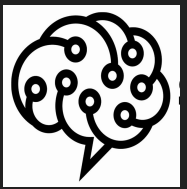
Agenda

Short recap previous sessions on **perceptrons** and **learning**

Represent functions

Learning in multi layer networks

Look back: what have we learned?

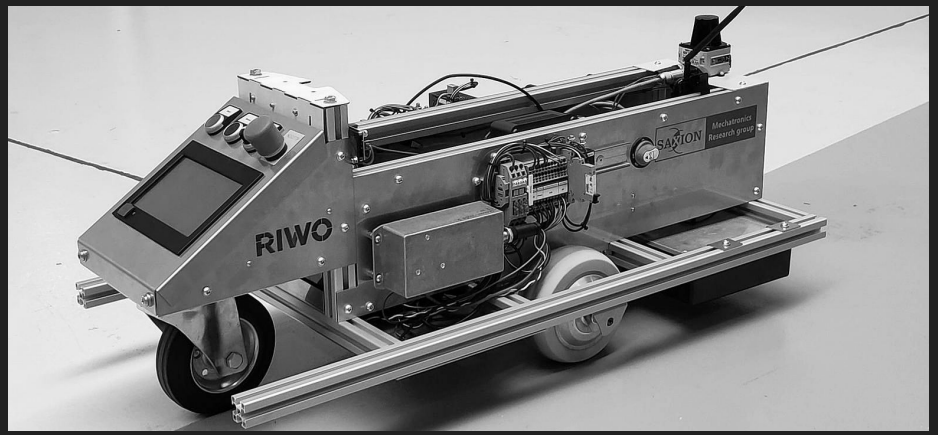


Wilco Bonestroo

Researcher at Mechatronics research group at Saxion

Software and AI for robots

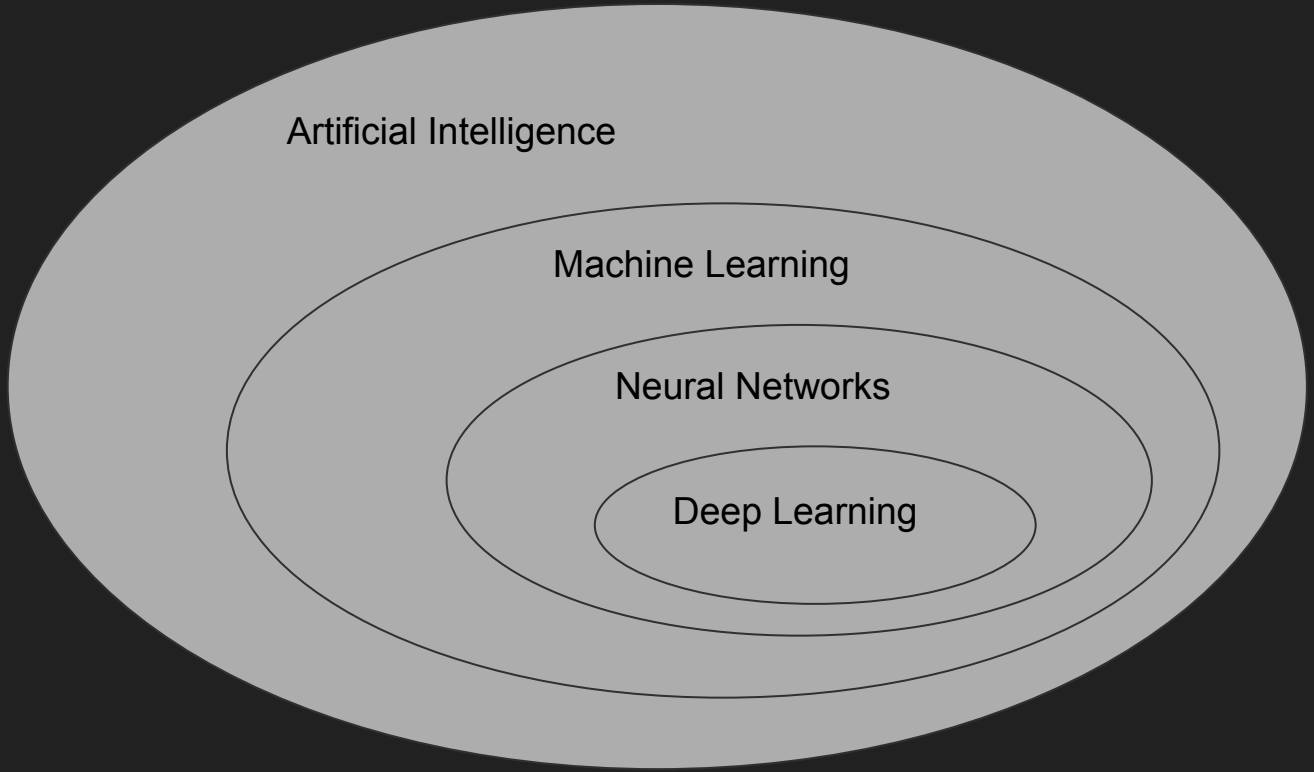
Autonomous mobile robots

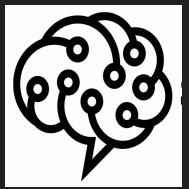


Research group

- Unmanned Robotics Systems
- Smart Industrial Systems

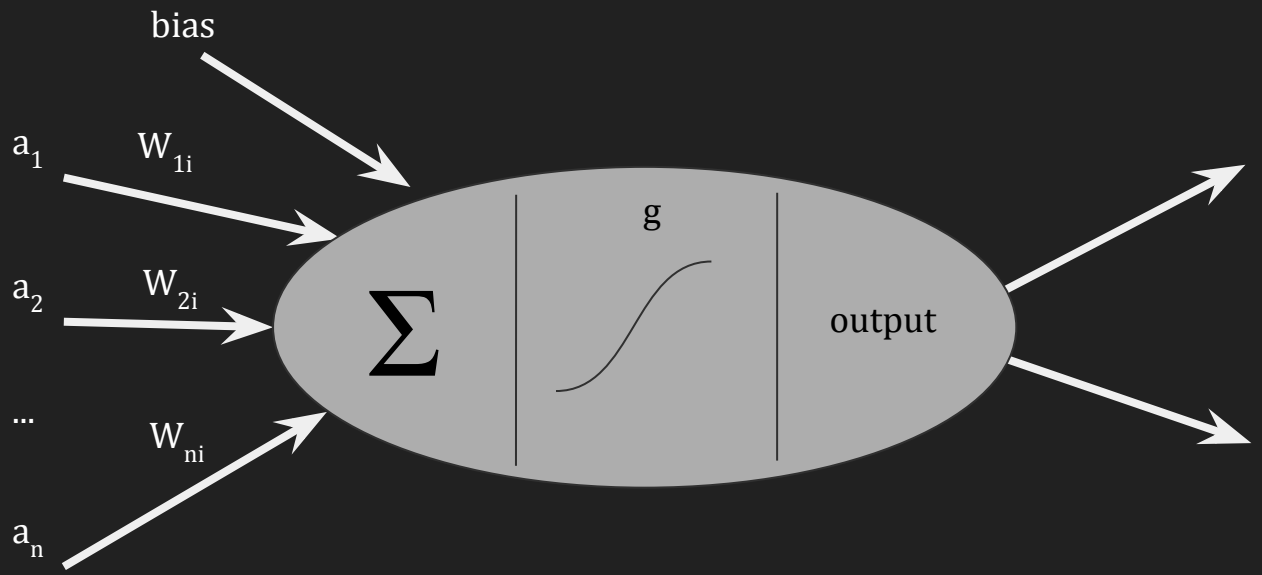
The bigger picture

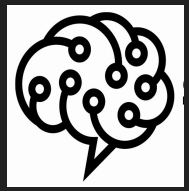




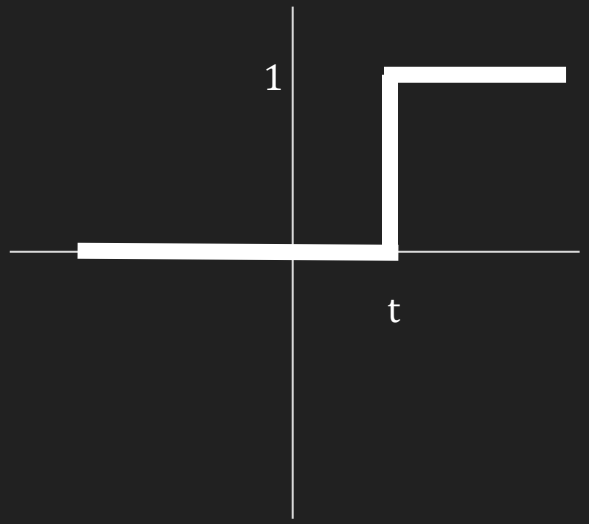
ANN concepts

- Node
- Link
- Input
- Weight
- Activation function
- Output
- Layer
- “Knowledge”

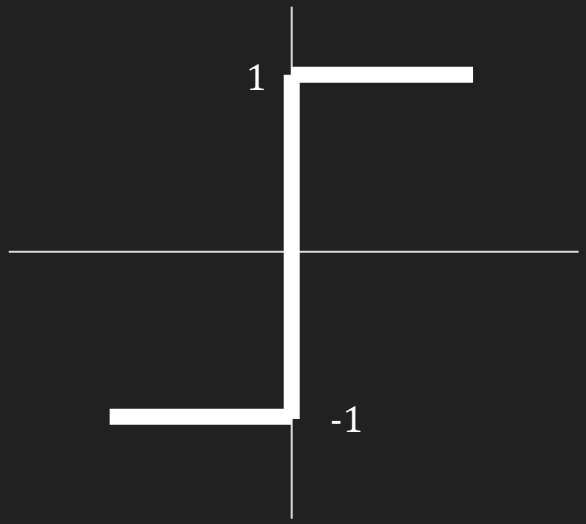




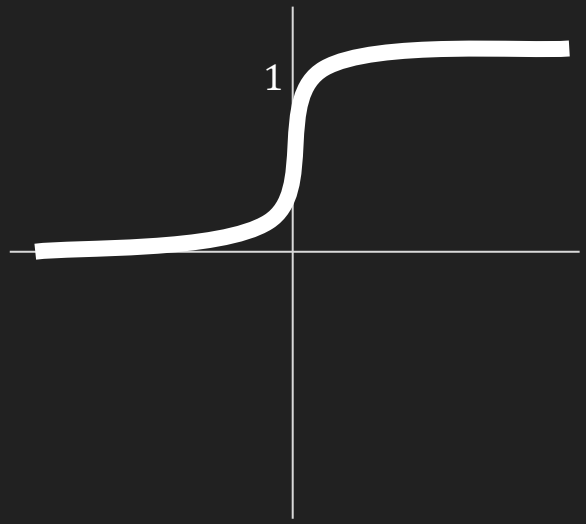
Common Activation Functions



Step function



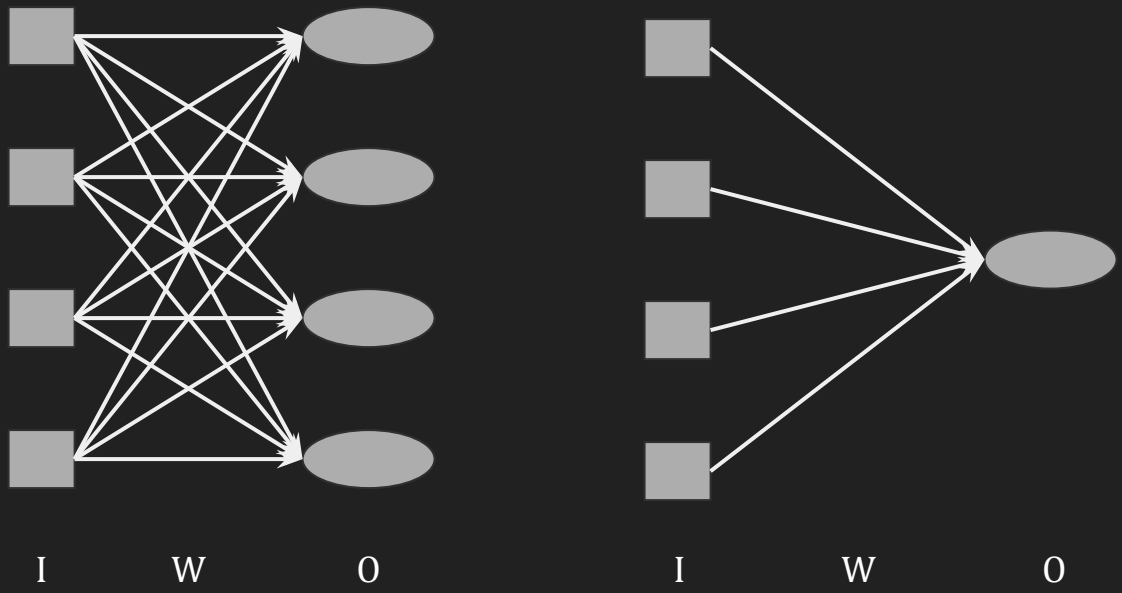
Sign function



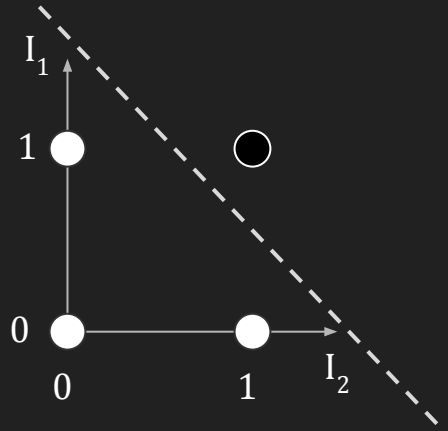
Sigmoid function

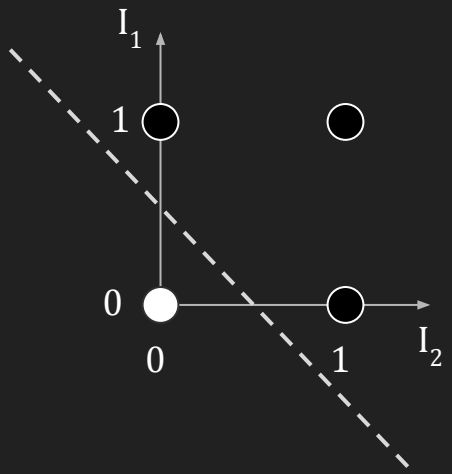
Perceptrons

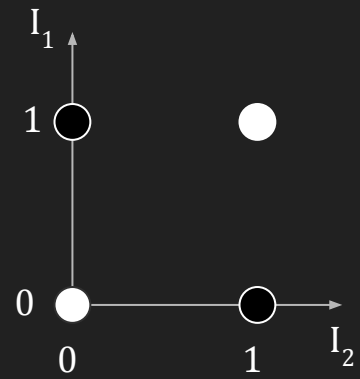
No hidden layers

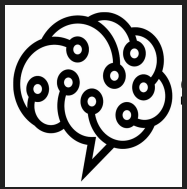


Linear Separable (two inputs / dimensions)







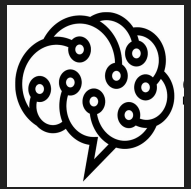


Learning in a perceptron

Find an algorithm that can “train” a network based on given examples.

Perceptron learning rule by Rosenblat (1960)

I_1	I_2	O
0	0	0
0	1	0
1	0	0
1	1	1



Generic Network Learning Algorithm

Initialise network (weights)

Repeat

    For each example

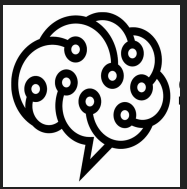
        Determine outcome of current network for example

        Compare outcome to expected value for example

        Update the network

    end

Until all examples are correctly classified



Generic Network Learning Algorithm

Initialise network (weights)

Repeat

    For each example

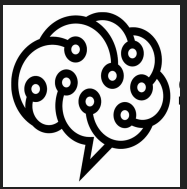
        Determine outcome of current network for example

        Compare outcome to expected value for example

        Update the network

    end

Until all examples are correctly classified

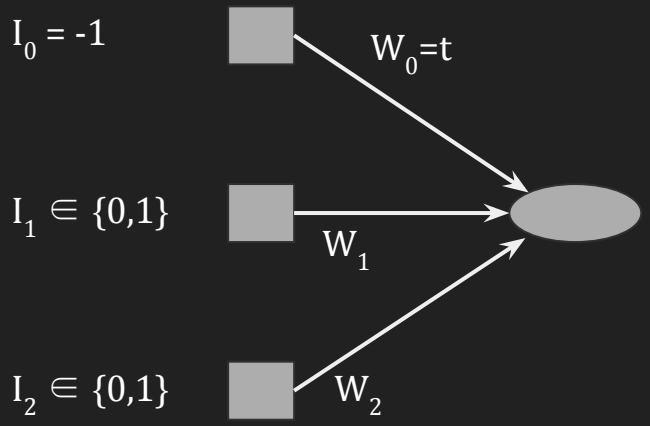


Examples (AND function)

W_0	W_1	W_2
0	0	0

I_0	I_1	I_2
-1	1	1

Output	Expected
0	1

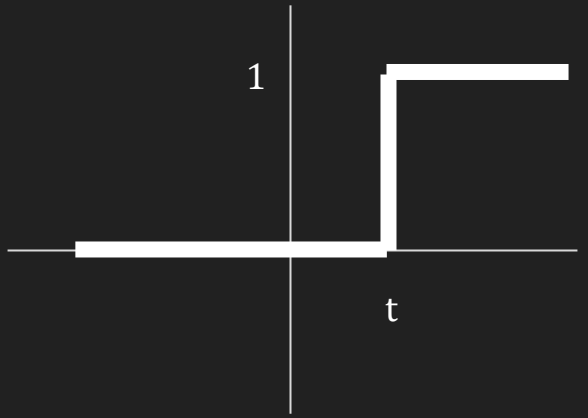


If the output should be **higher**, we should **add** the values of I_j to W_j

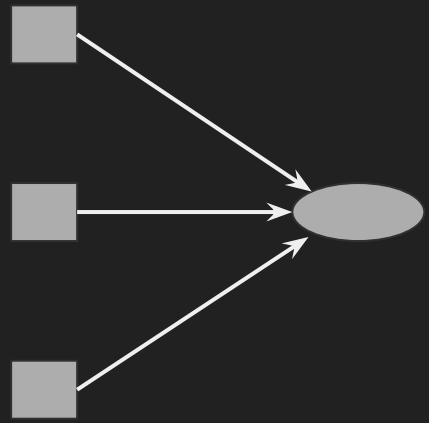
Short notation: W becomes $W + \alpha \times I$

If the output should be **lower**, we should **subtract** the values of I_j from W_j

Short notation: W becomes $W - \alpha \times I$



Step function



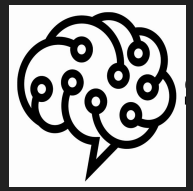
The actual output is O

The expected output is T

The error Err = T - O

Learning Rule:

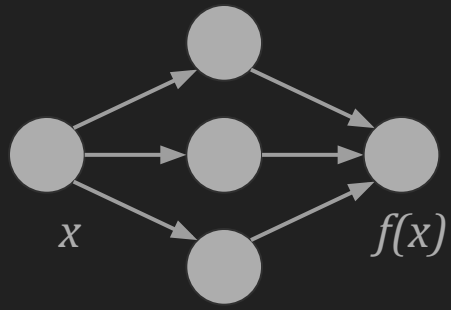
W_j becomes W_j + α x I_j x Err

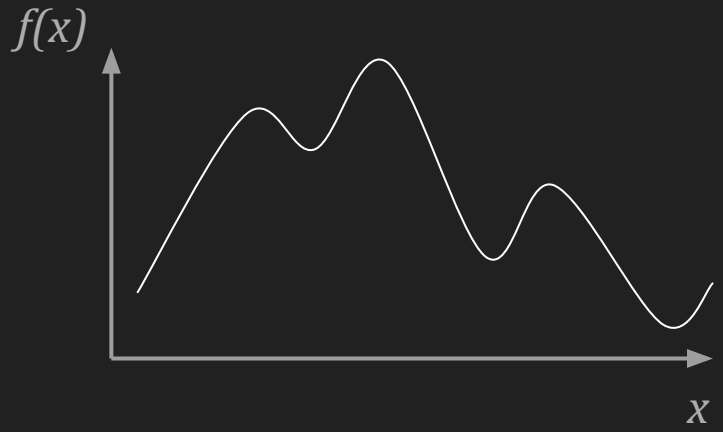


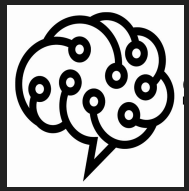
How to represent other functions?

Add layers

“With one (sufficiently large) layer of hidden units, it is possible to represent any continuous function of the inputs.”

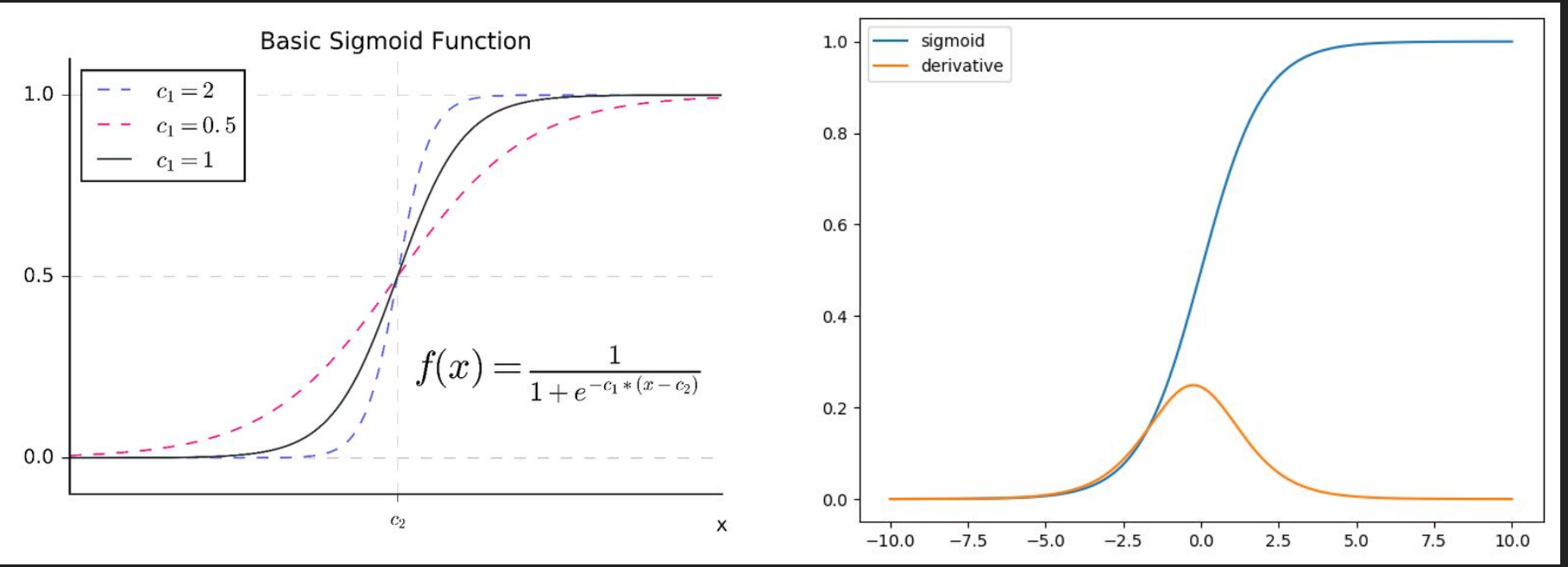




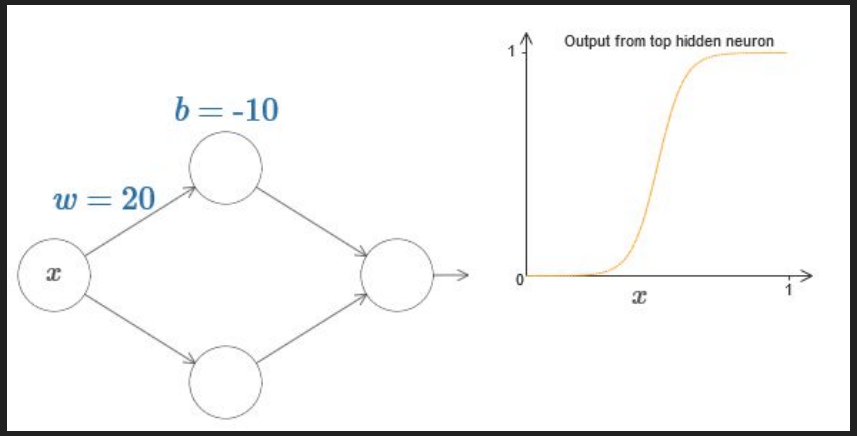


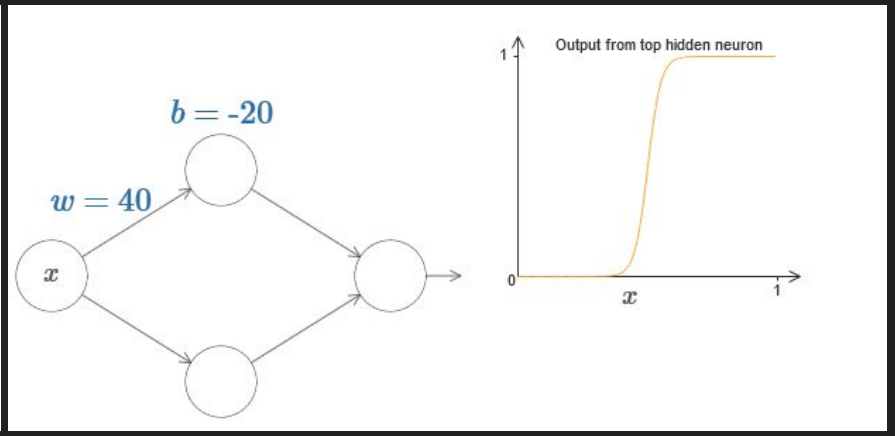
Activation function: Sigmoid

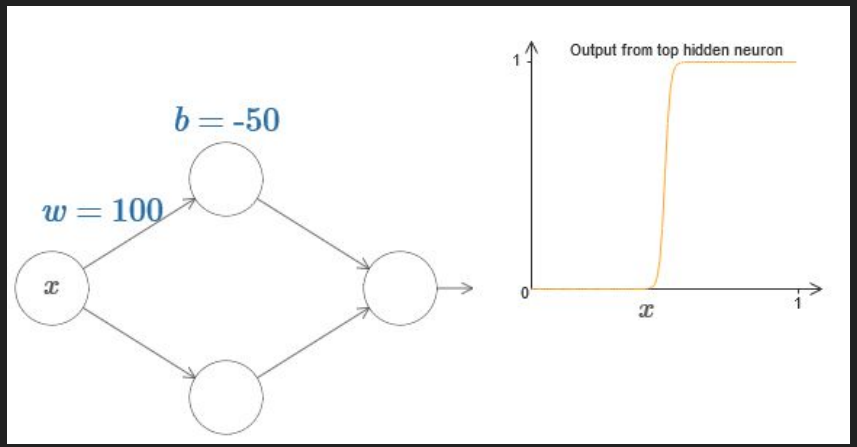
$g(z) = 1 / (1 + e^{-z})$ $z = wx + b$

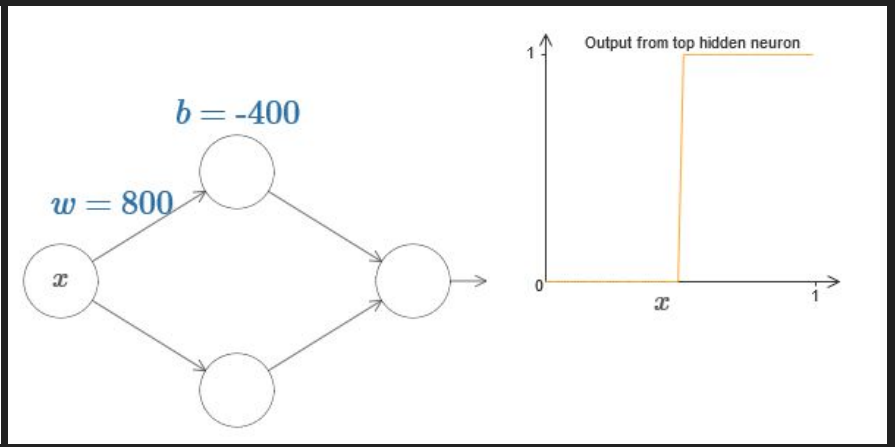


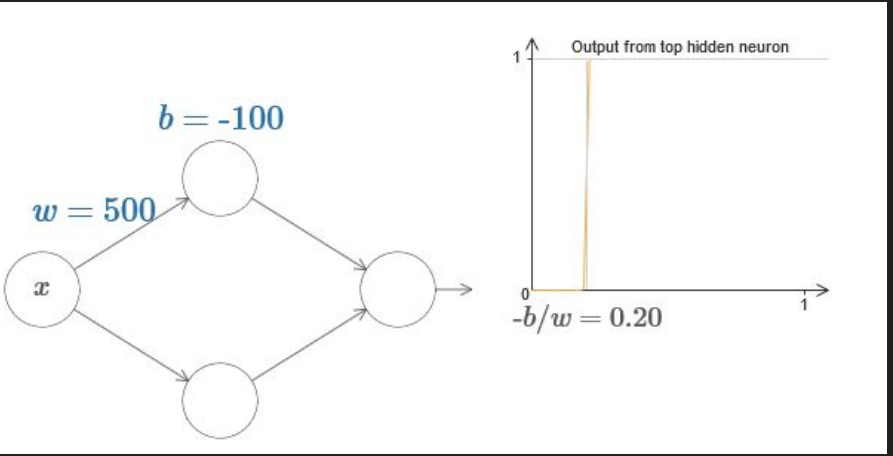
$g(z) = 1 / (1 + e^{-z})$ $z = wx + b$

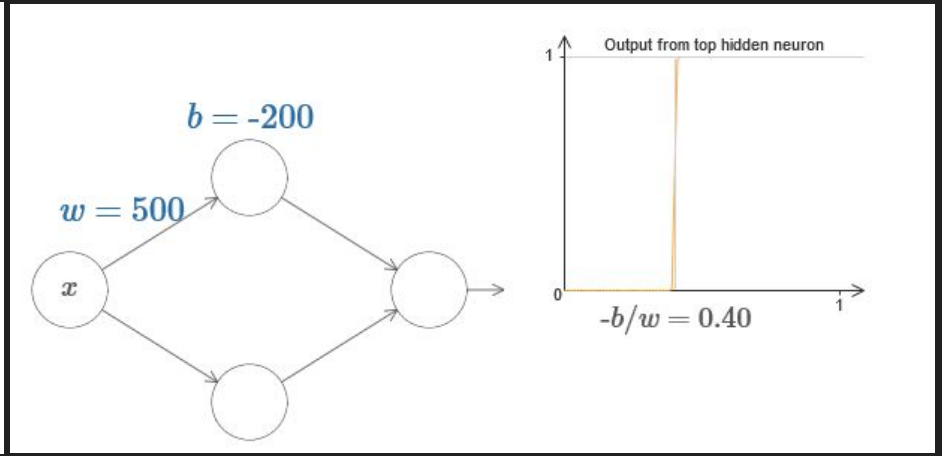

b = -10
w = 20
x
Output from top hidden neuron
1
0
x
1

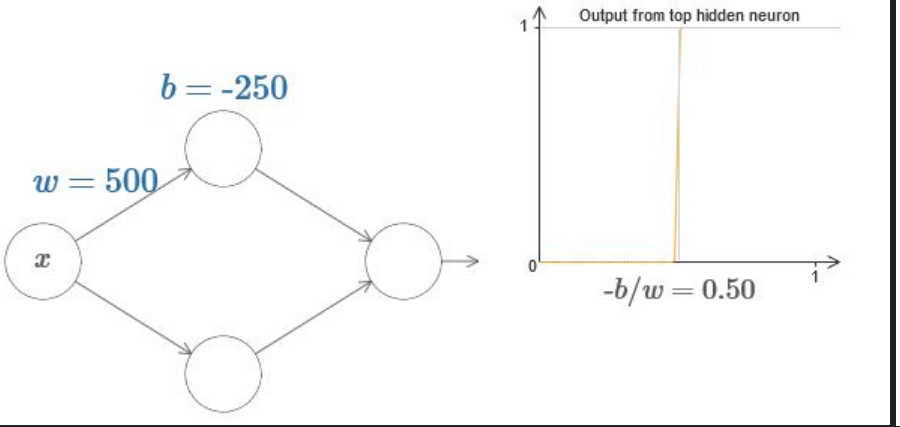

b = -20
w = 40
x
Output from top hidden neuron
1
0
x
1

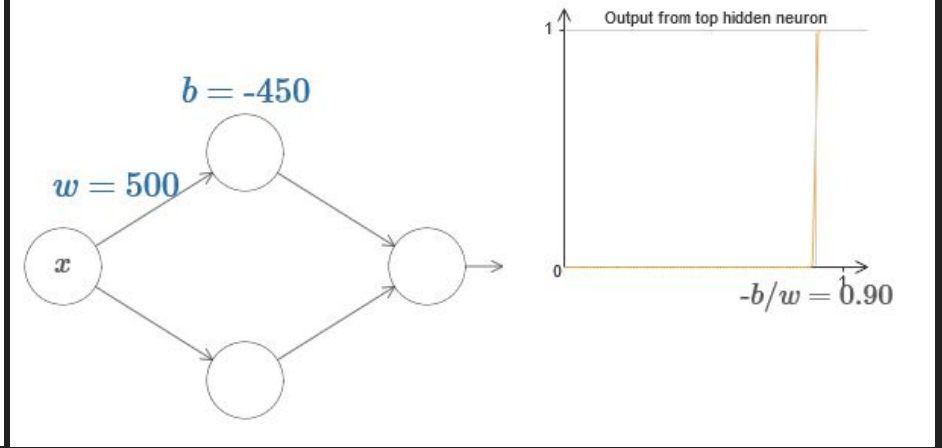

b = -50
w = 100
x
Output from top hidden neuron
1
0
x
1


b = -400
w = 800
x
Output from top hidden neuron
1
0
x
1

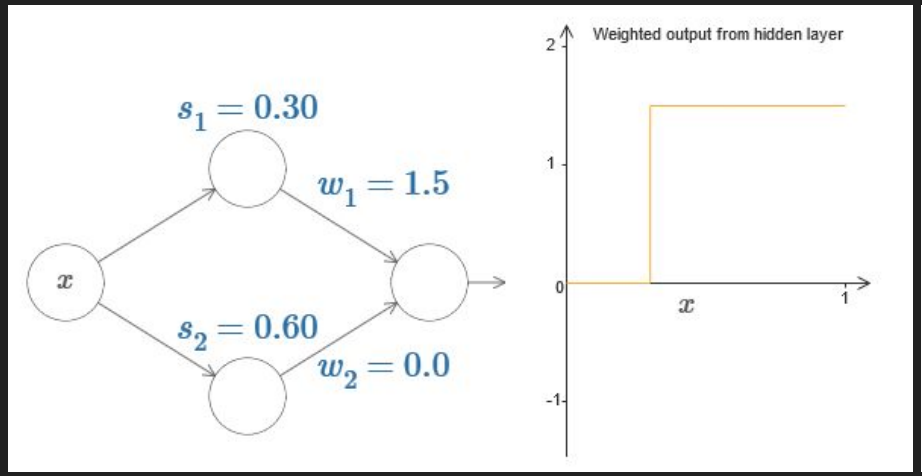


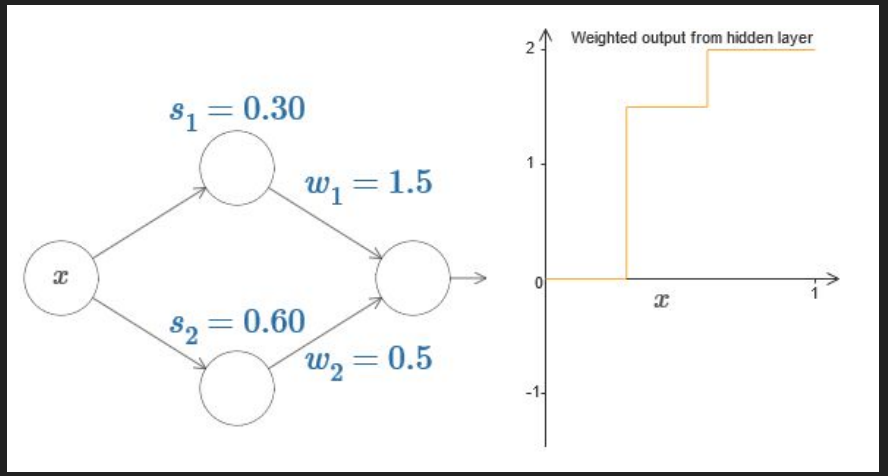


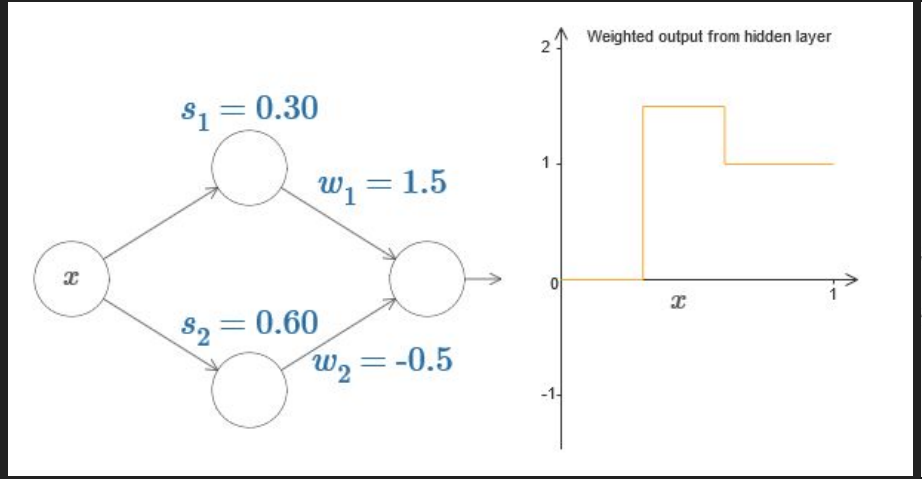


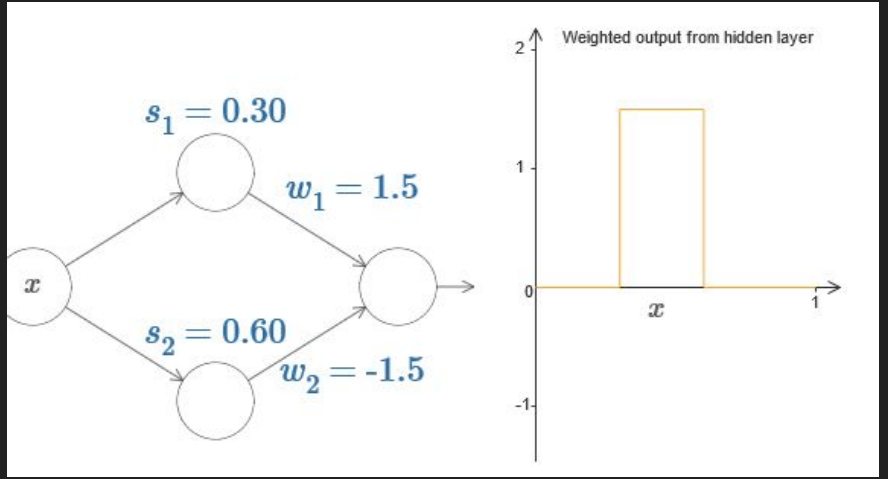


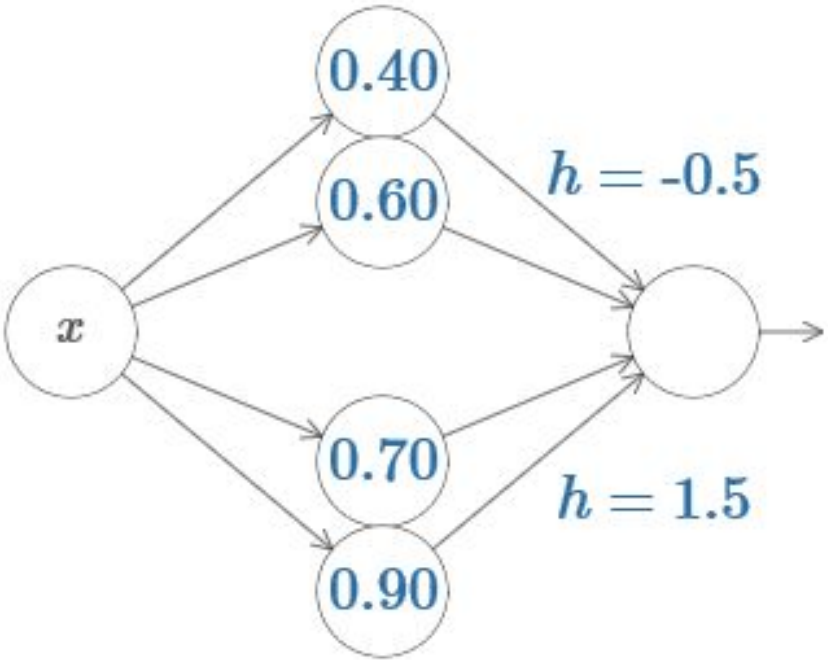
$g(z) = 1 / (1 + e^{-z})$ $z = wx + b$ $s = -b/w$

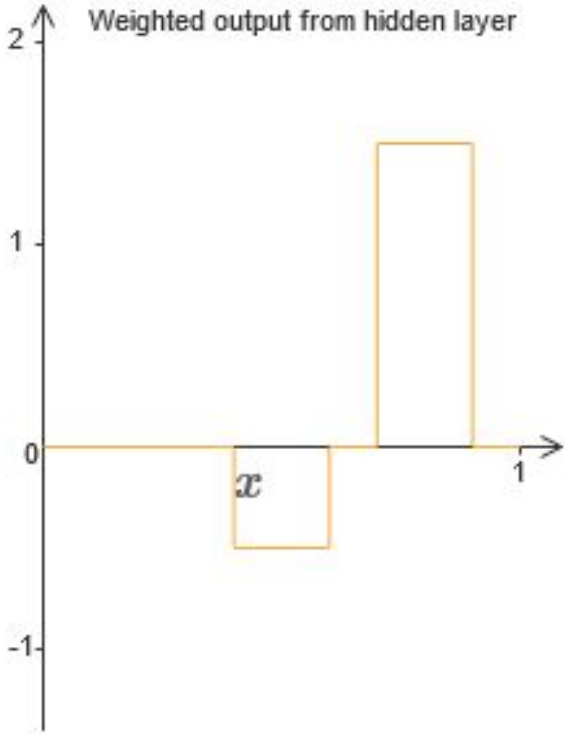

$s_1 = 0.30$
$w_1 = 1.5$
x
$s_2 = 0.60$
$w_2 = 0.0$
Weighted output from hidden layer
2
1
0
-1
x
1

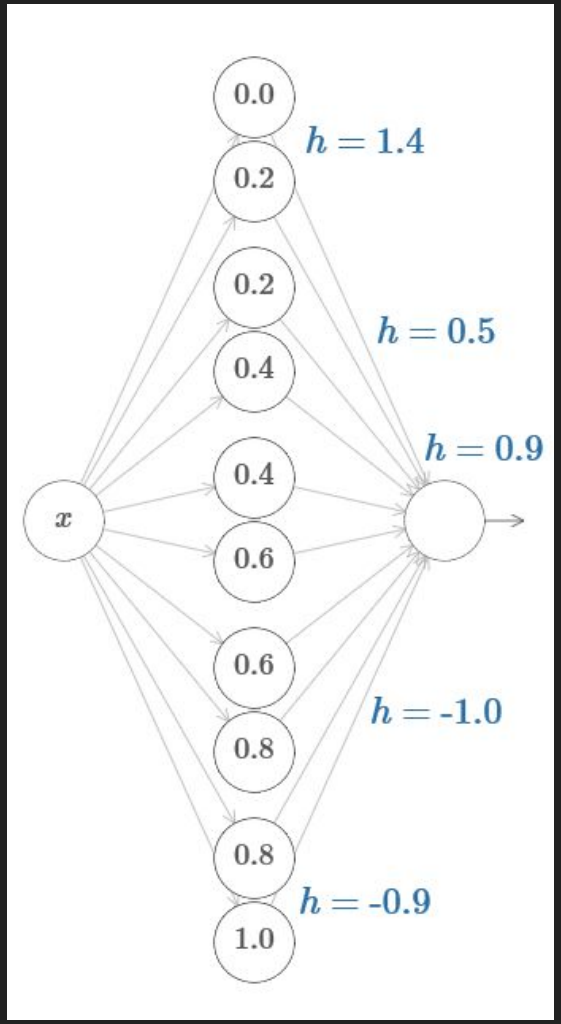

$s_1 = 0.30$
$w_1 = 1.5$
x
$s_2 = 0.60$
$w_2 = 0.5$
Weighted output from hidden layer
2
1
0
-1
x
1

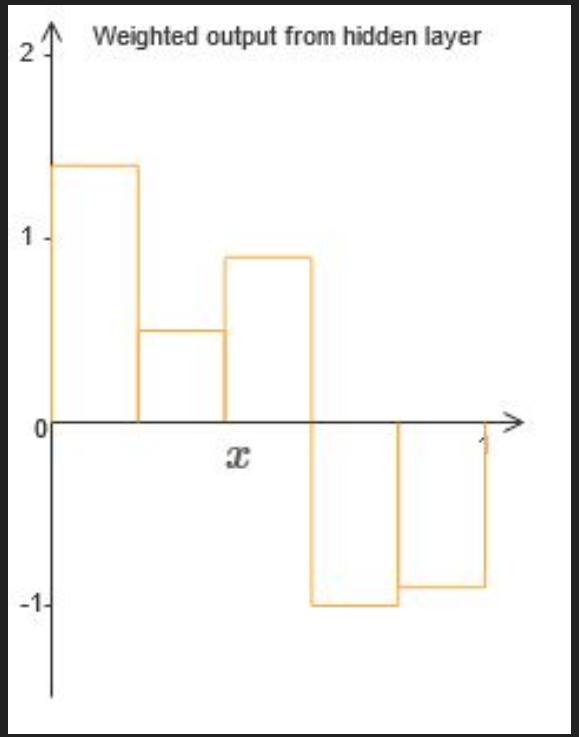

$s_1 = 0.30$
$w_1 = 1.5$
x
$s_2 = 0.60$
$w_2 = -0.5$
Weighted output from hidden layer
2
1
0
-1
x
1


$s_1 = 0.30$
$w_1 = 1.5$
x
$s_2 = 0.60$
$w_2 = -1.5$
Weighted output from hidden layer
2
1
0
-1
x
1


0.40
0.60
0.70
0.90
x
$h = -0.5$
$h = 1.5$


Weighted output from hidden layer
2
1
0
-1
x
1

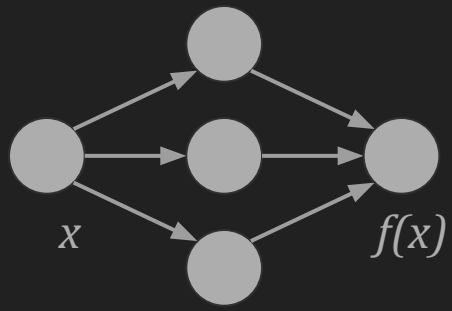

0.0
0.2
0.2
0.4
0.4
0.6
0.6
0.8
0.8
1.0
x
h = 1.4
h = 0.5
h = 0.9
h = -1.0
h = -0.9

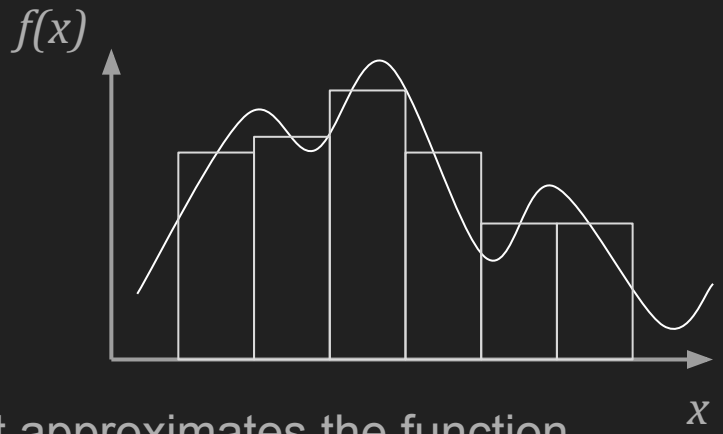

Weighted output from hidden layer
2
1
0
-1
x

How to represent other functions?

Add layers

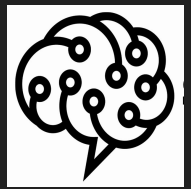
“With one (sufficiently large) layer of hidden units, it is possible to represent any continuous function of the inputs.”





It approximates the function

Learning in multi layer network



Generic Network Learning Algorithm

Initialise network (weights)

Repeat

    For each example

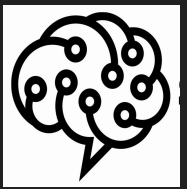
        Determine outcome of current network for example

        Compare outcome to expected value for example

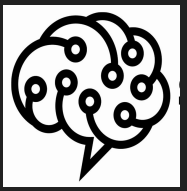
        Update the network

    end

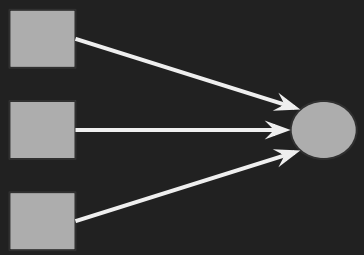
Until all examples are correctly classified

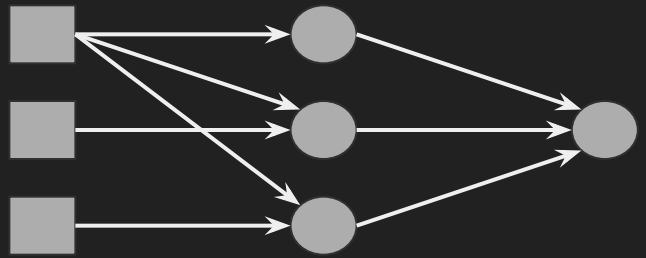


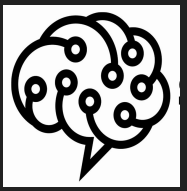
Updating the weights



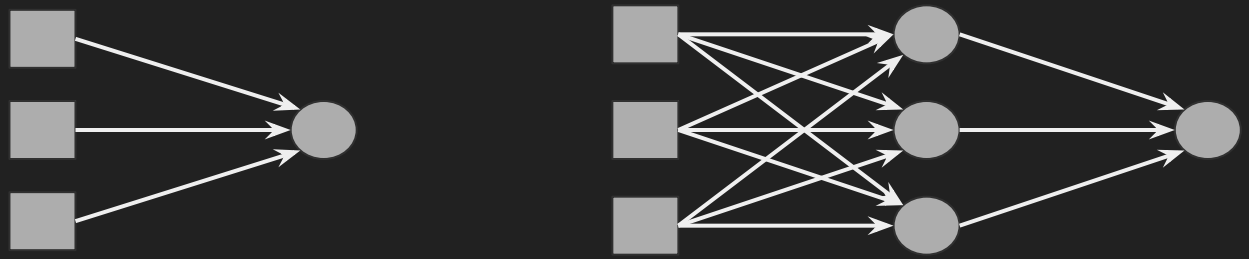
Updating the weights

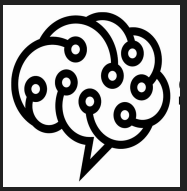




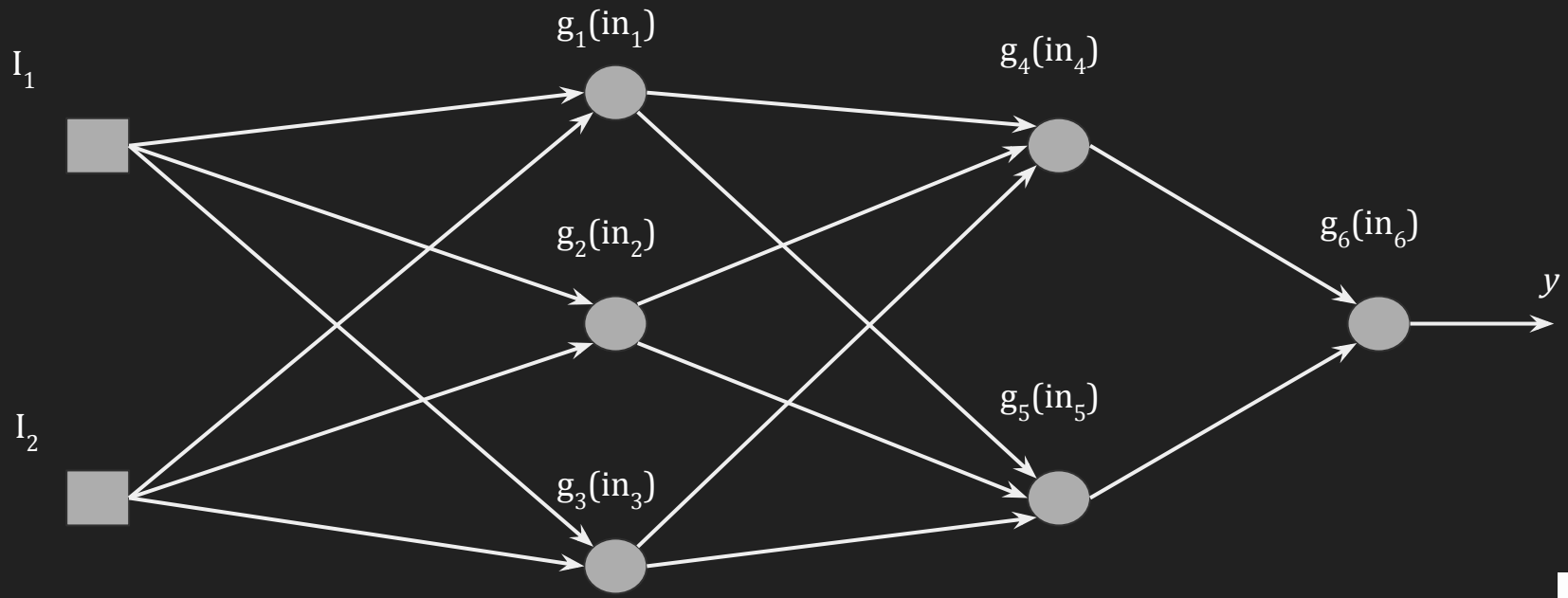


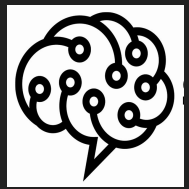
Updating the weights



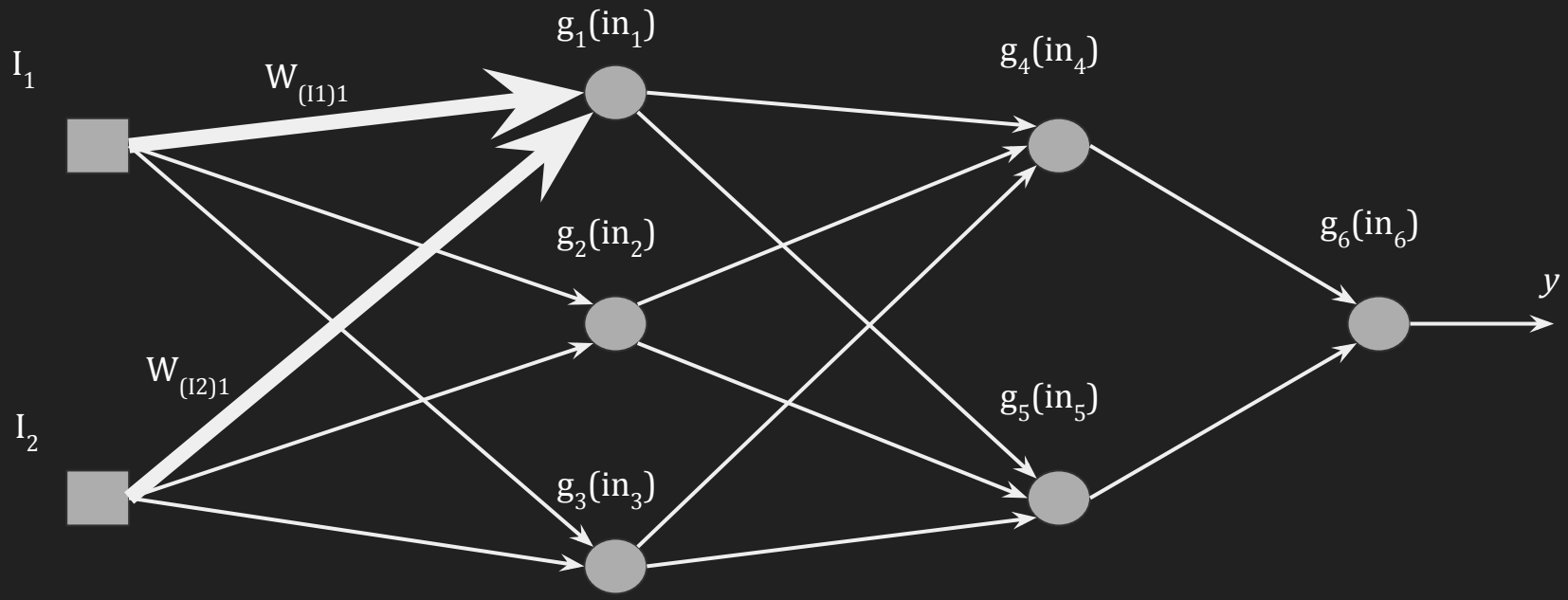


Back Propagation: give an example



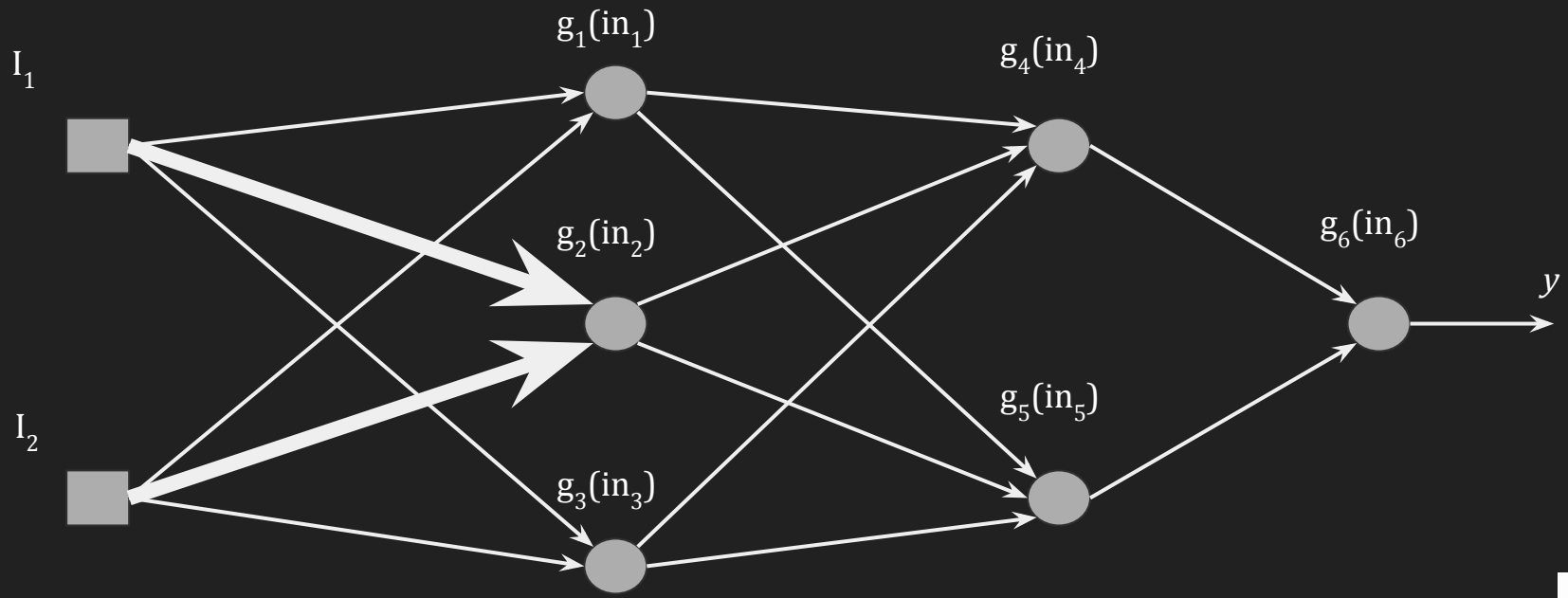


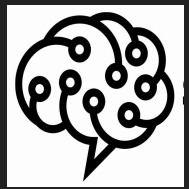
Back Propagation



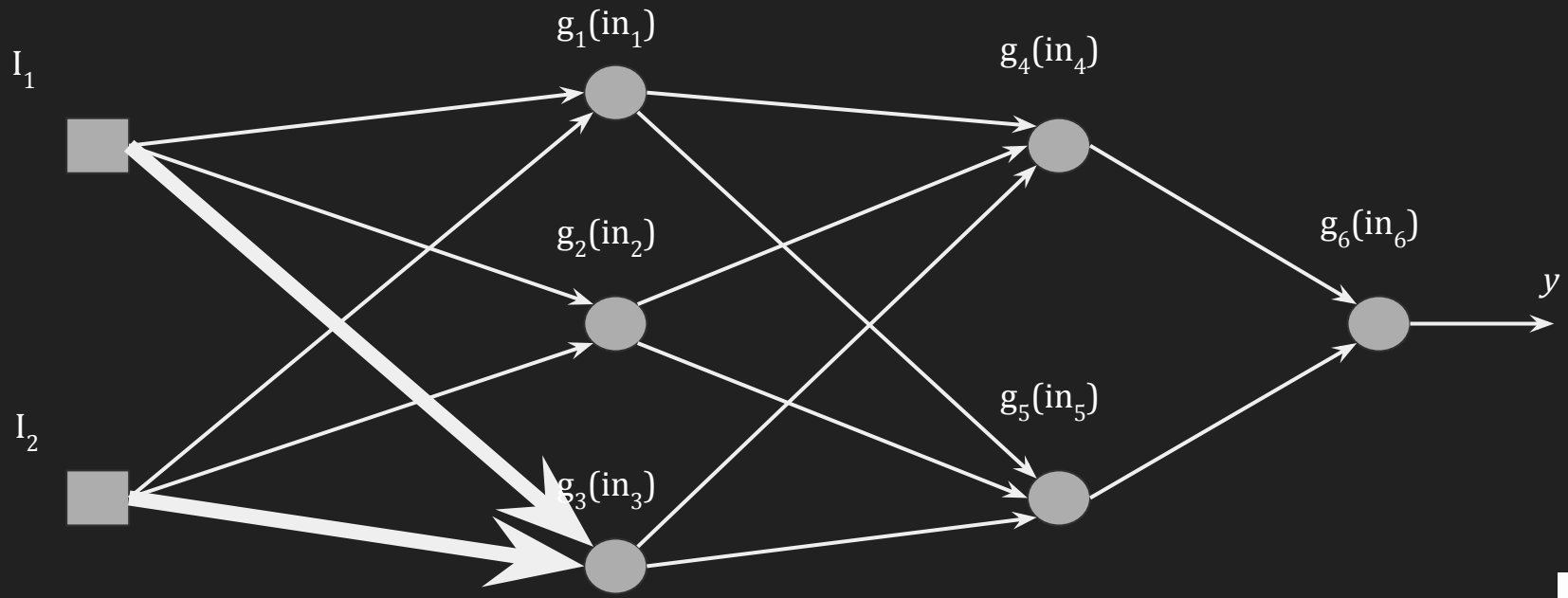
$$y_1 = g_1(\text{in}_1) = g_1(I_1 * W_{(I1)1} + I_2 * W_{(I2)1})$$

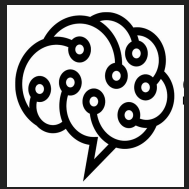
Back Propagation



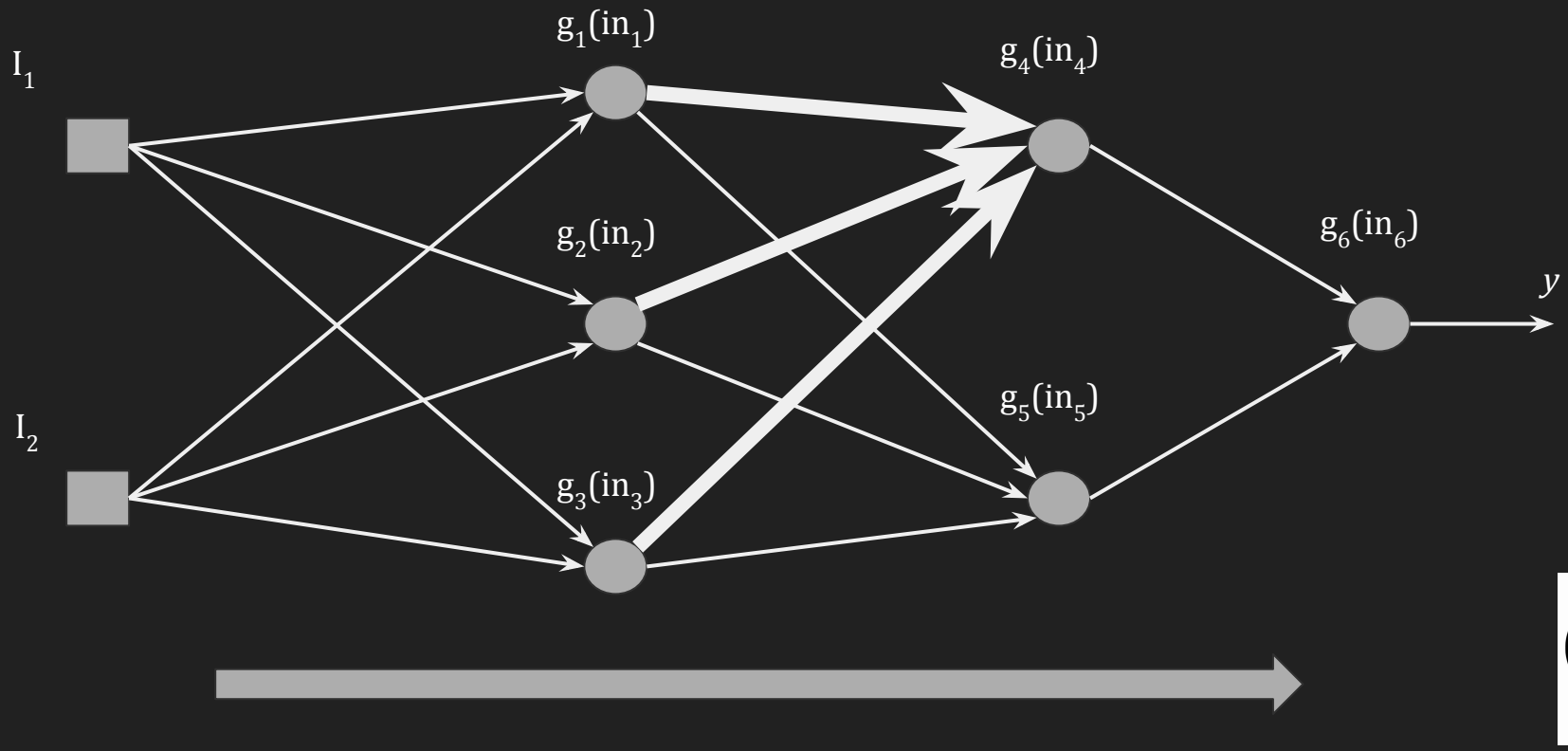


Back Propagation

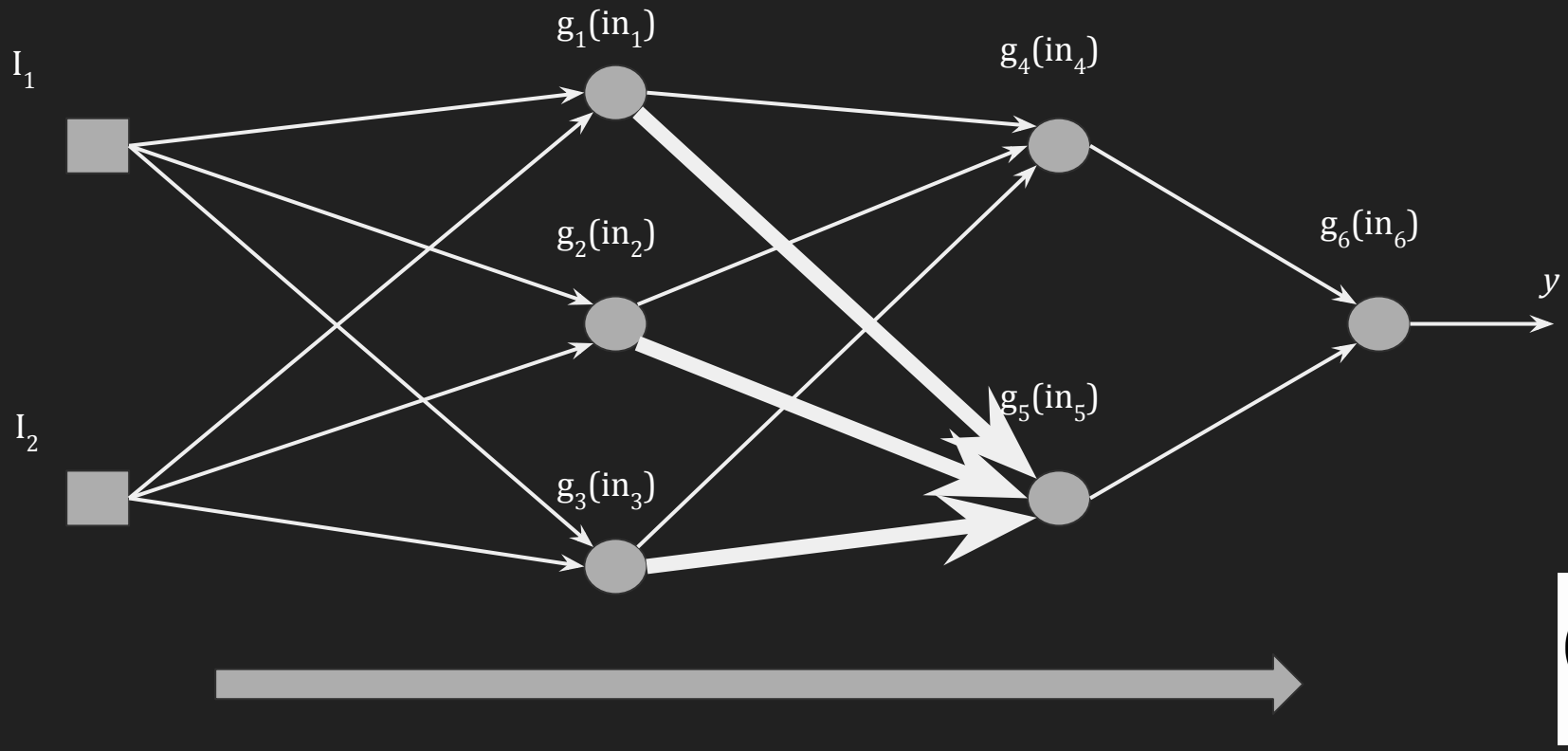




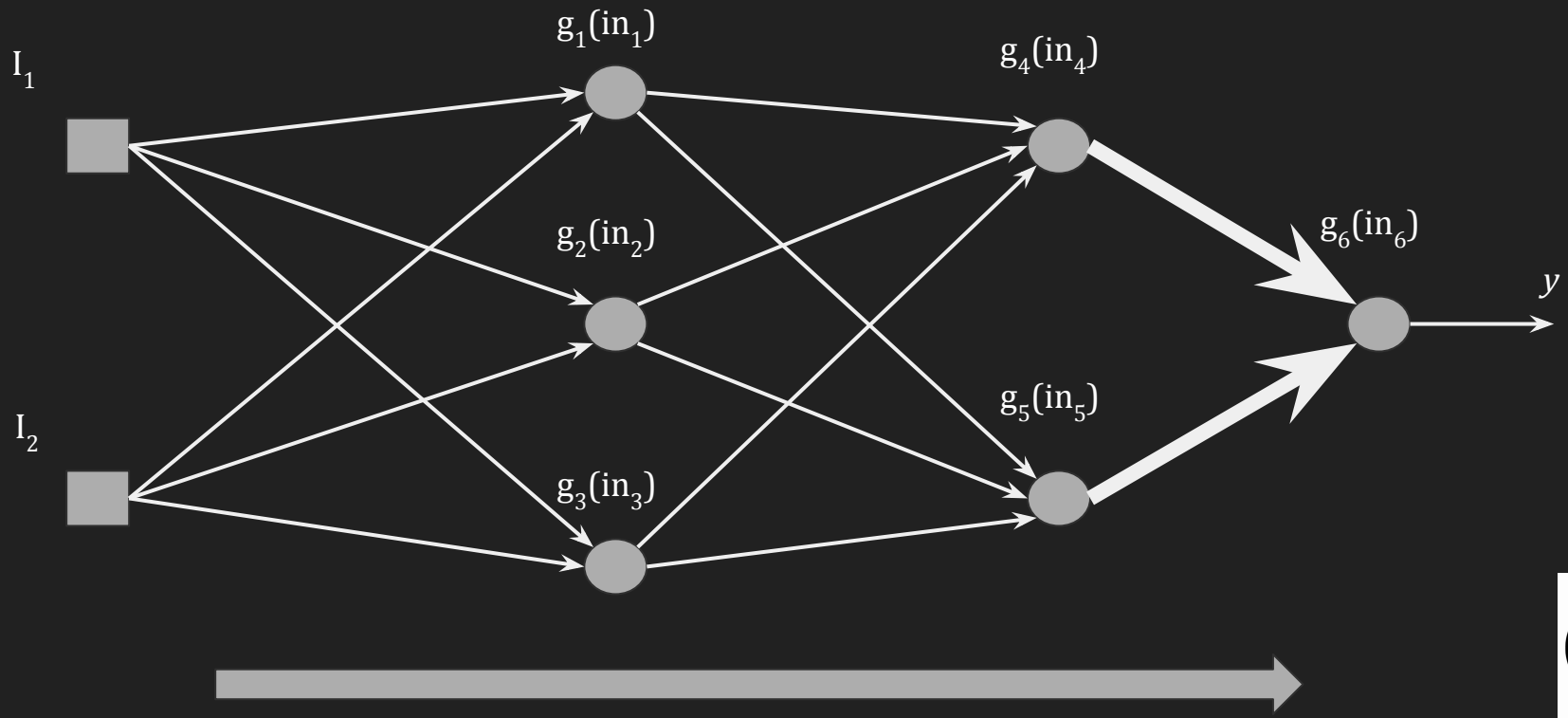
Back Propagation



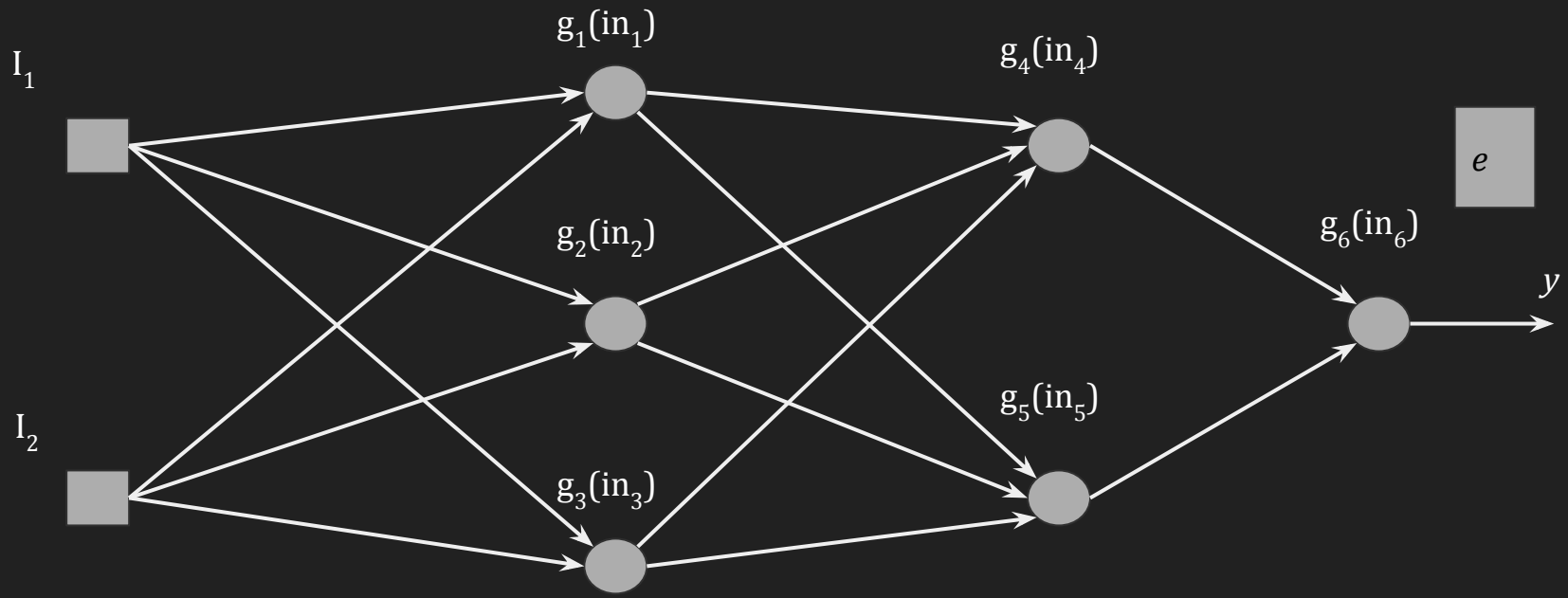
Back Propagation



Back Propagation



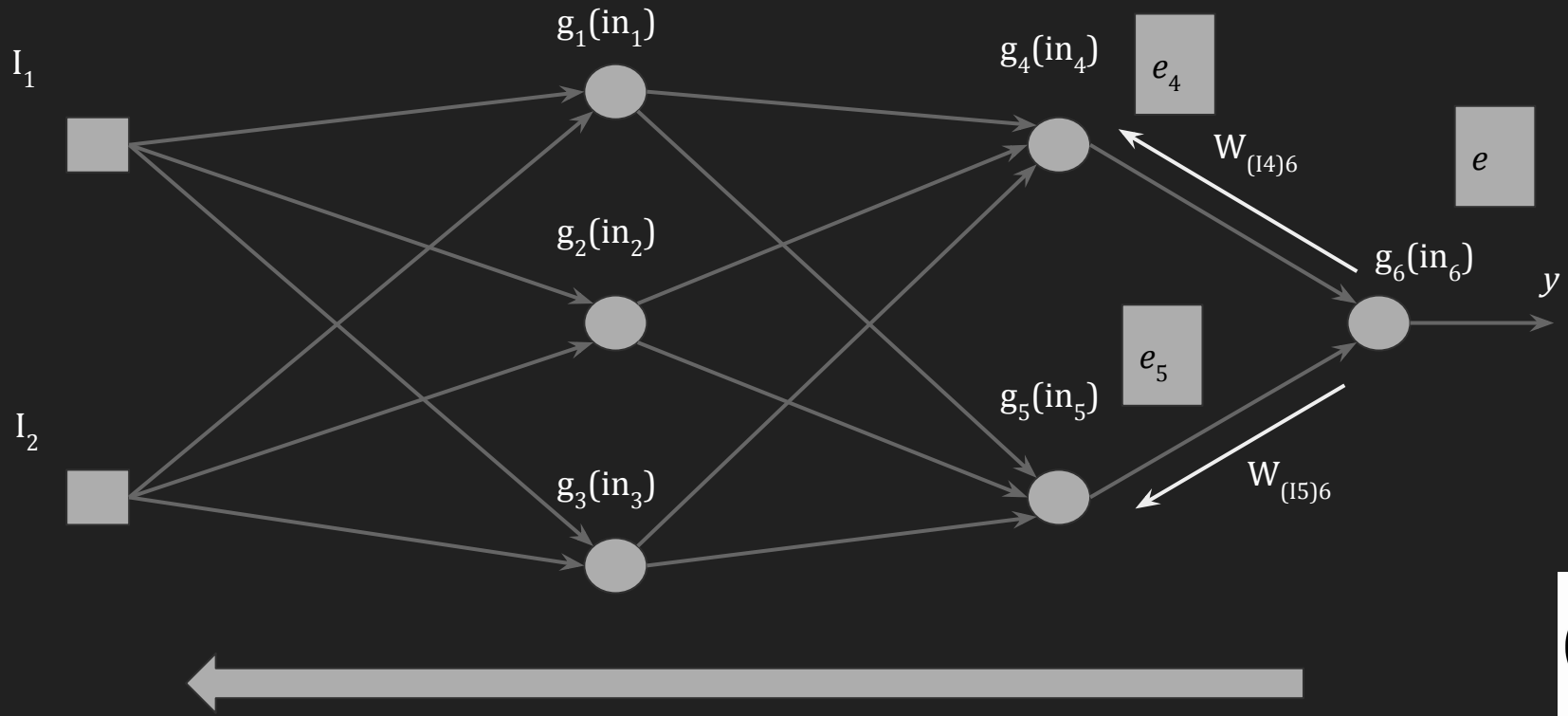
Back Propagation



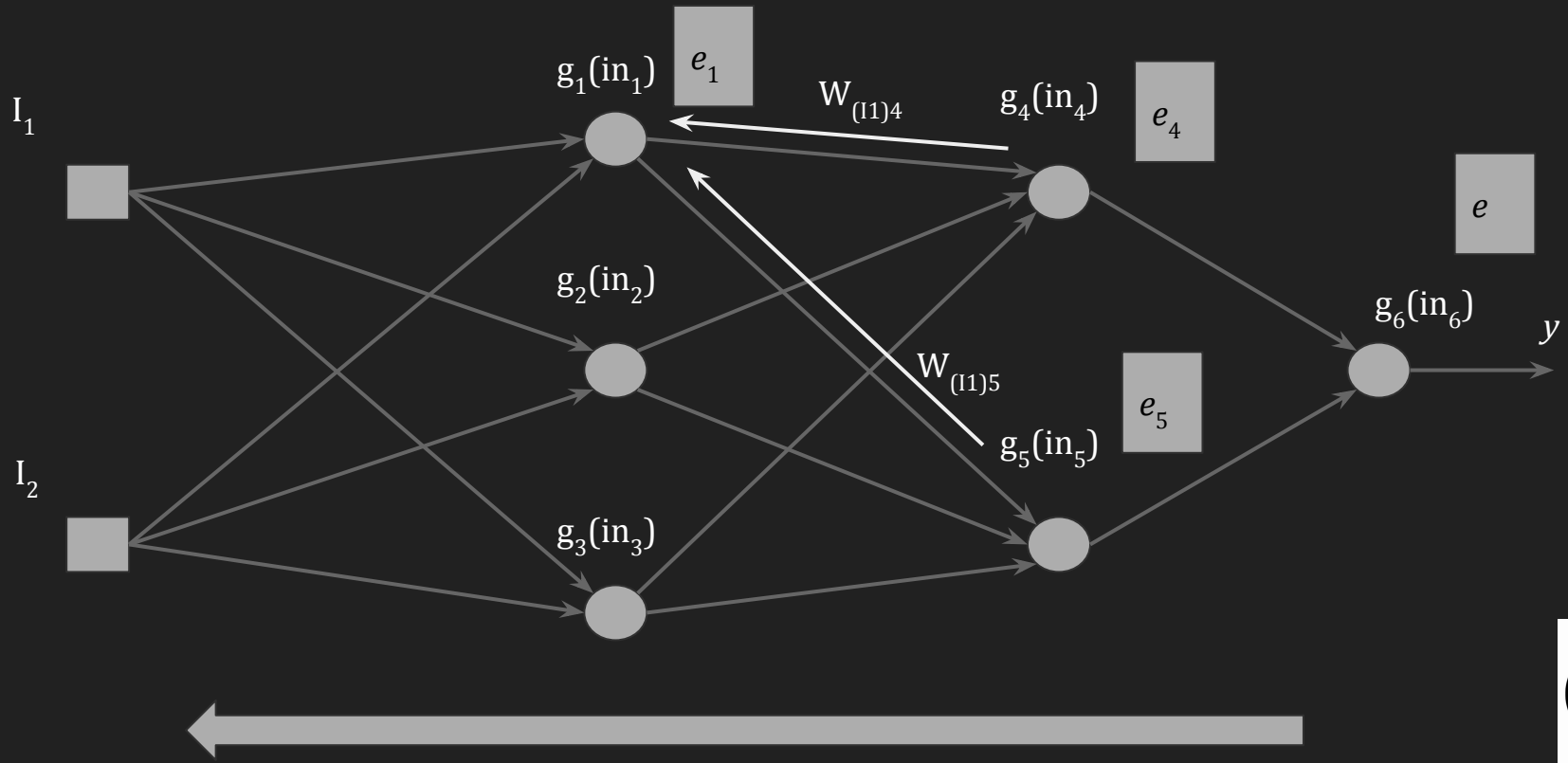
This was feed forward / forward propagation. We can now determine error e

Back Propagation

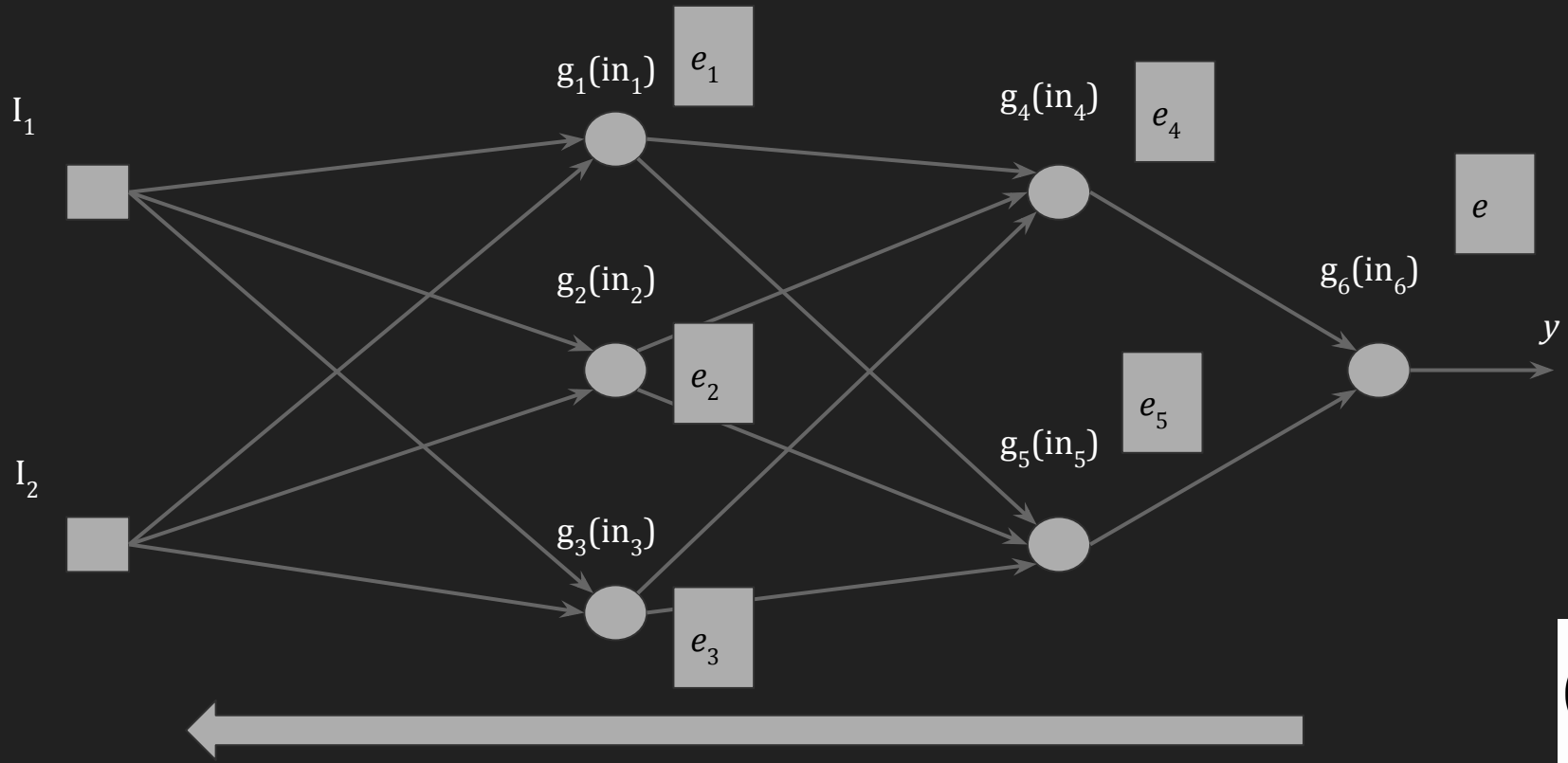
Distribute according to current weights

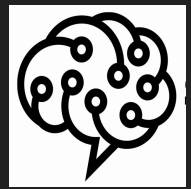


Back Propagation



Back Propagation

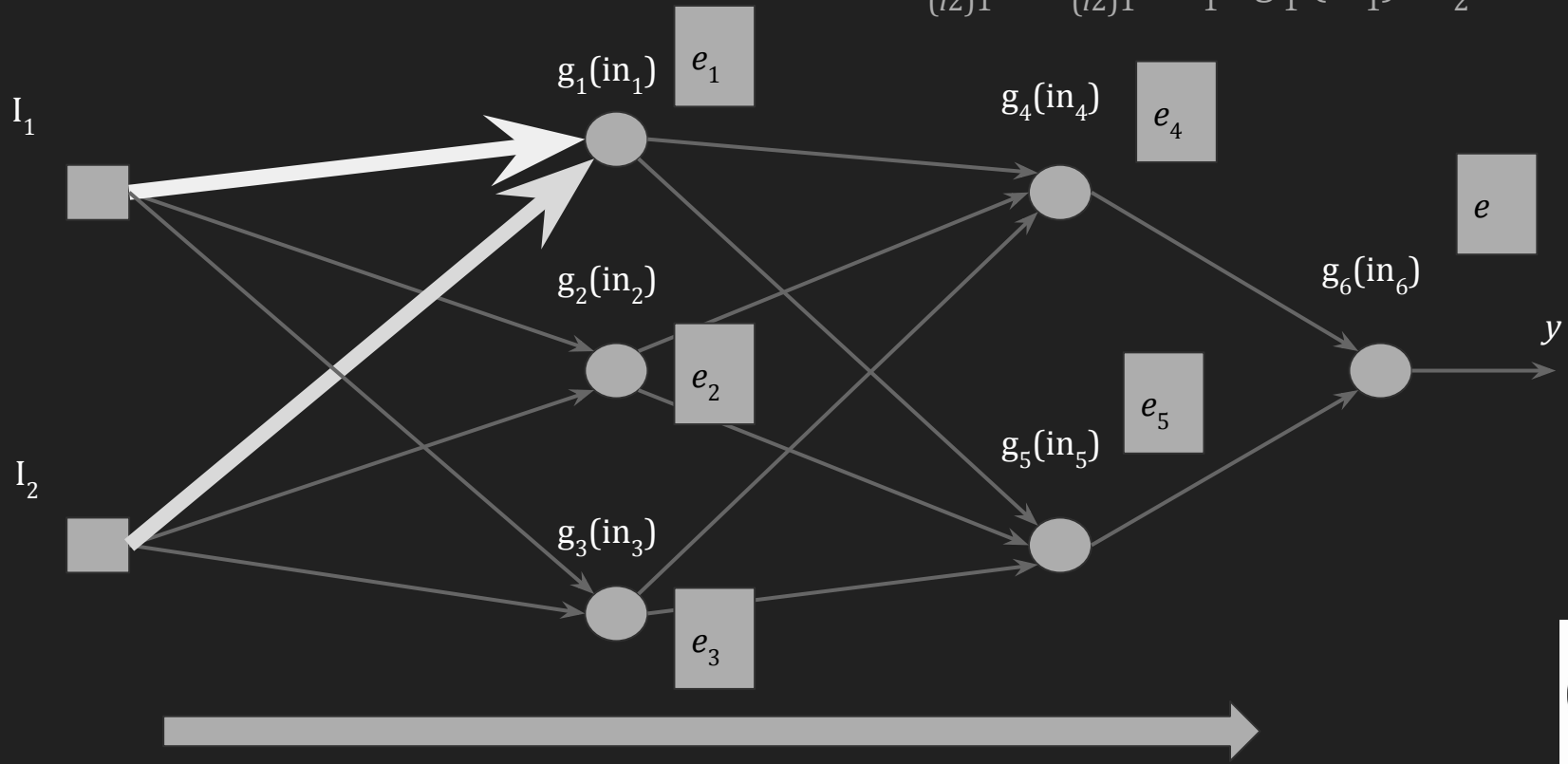


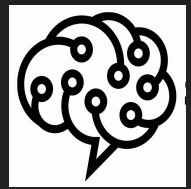


Back Propagation

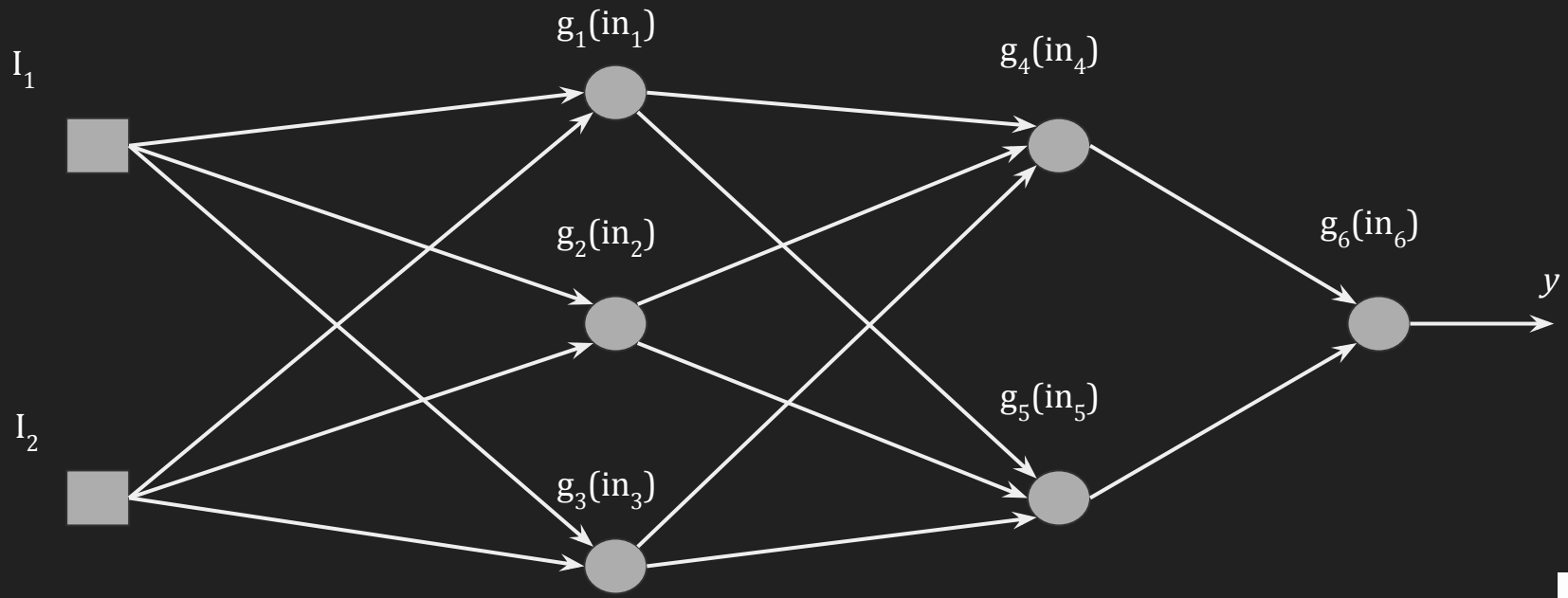
$$w'_{(i1)1} = w_{(i1)1} + e_1 * g_1'(\text{in}_1) * I_1$$

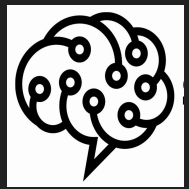
$$w'_{(i2)1} = w_{(i2)1} + e_1 * g_1'(\text{in}_1) * I_2$$





Back Propagation: iterate through all examples



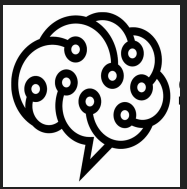


What do we know?

Neural Networks with one hidden layer can approximate functions

Two types of learning

- Learning in perceptrons
- Back Propagation in multi layered networks



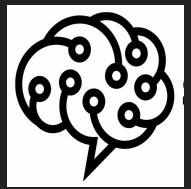
Resources

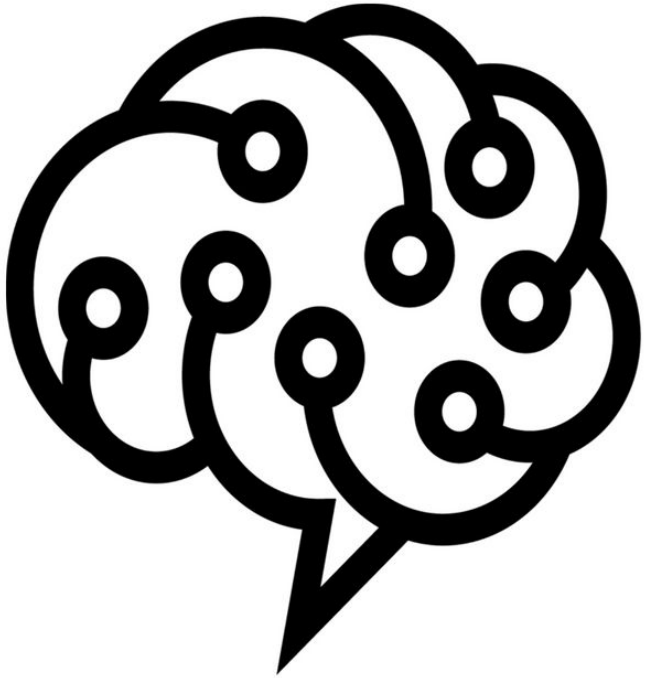
Book: Artificial Intelligence: A Modern Approach

Auteur: Stuart Russell

First part: neuralnetworksanddeeplearning.com

Second part: https://www.ibu.edu.ba/assets/userfiles/it/2012/eee-ANN-5.pdf





school of ai