



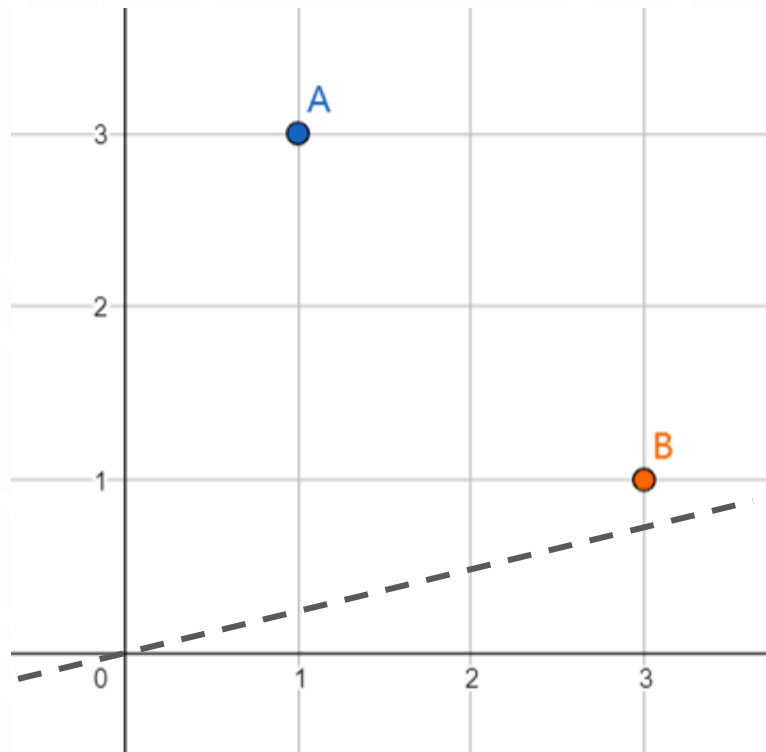
MUDSS
Data Science Society

Core Workshop 8: Artificial Neural Network

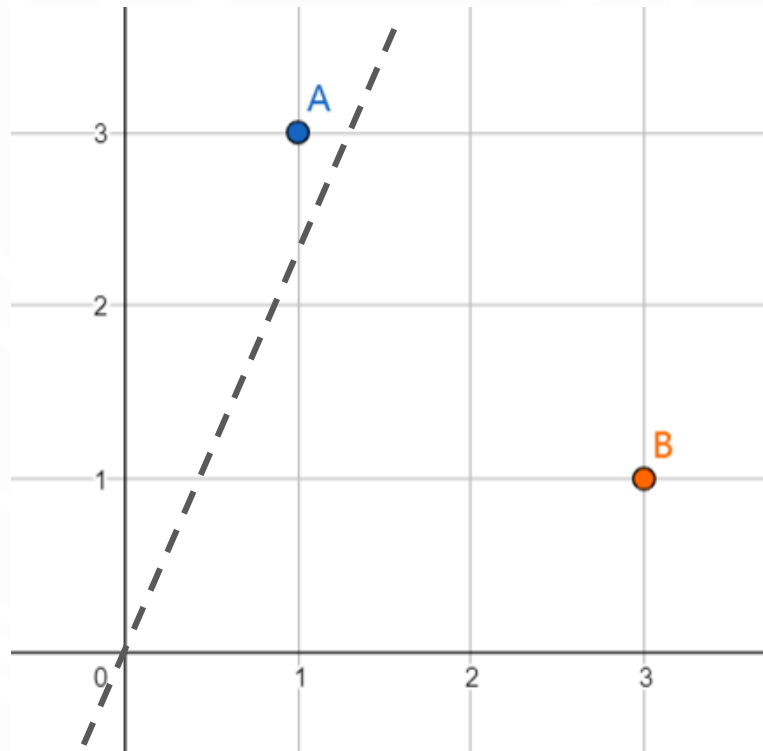
OVERVIEW

- A Simple Linear Classifier
 - One Classifier Is Not Enough
 - Neuron & Activation Function
- Feedforward Signal & Matrix Multiplication
- Backpropagation of Loss
 - Matrix Multiplication Again
 - Update Weights by Gradient Descent
- Input / Output & Initial Weights
- NumPy Code Example on MNIST

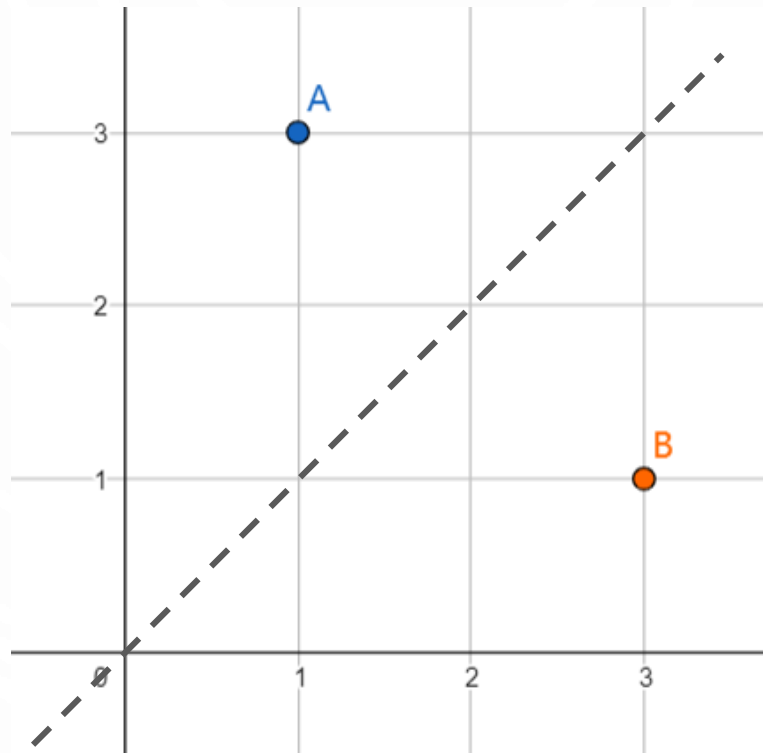
A SIMPLE LINEAR CLASSIFIER



A SIMPLE LINEAR CLASSIFIER

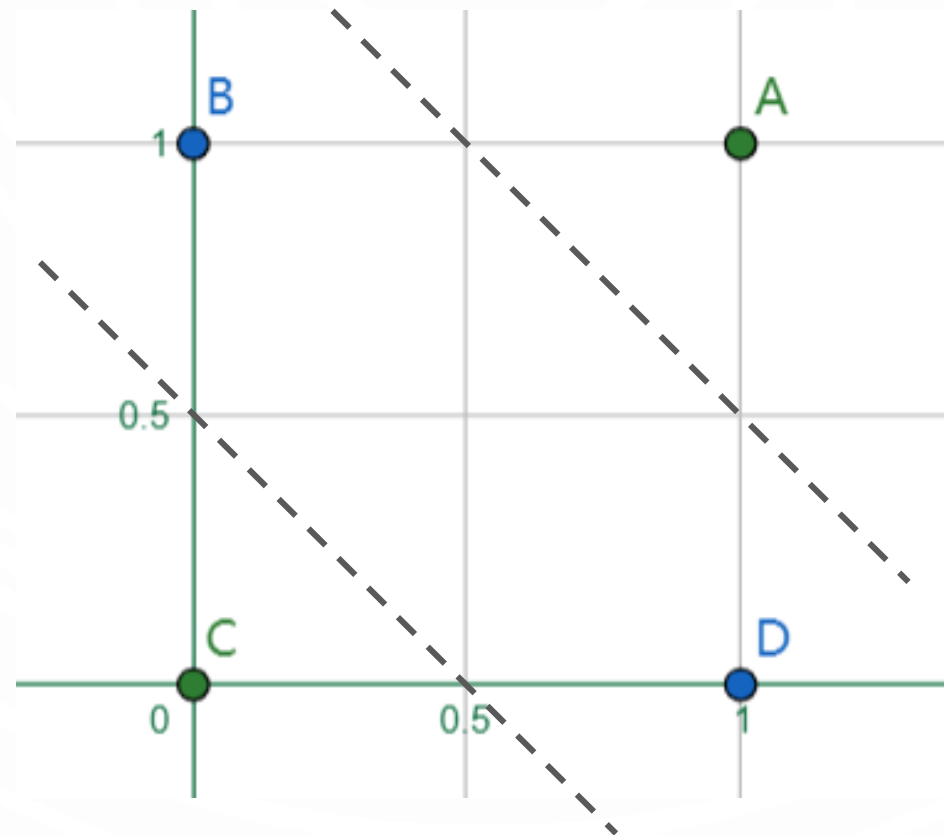


A SIMPLE LINEAR CLASSIFIER

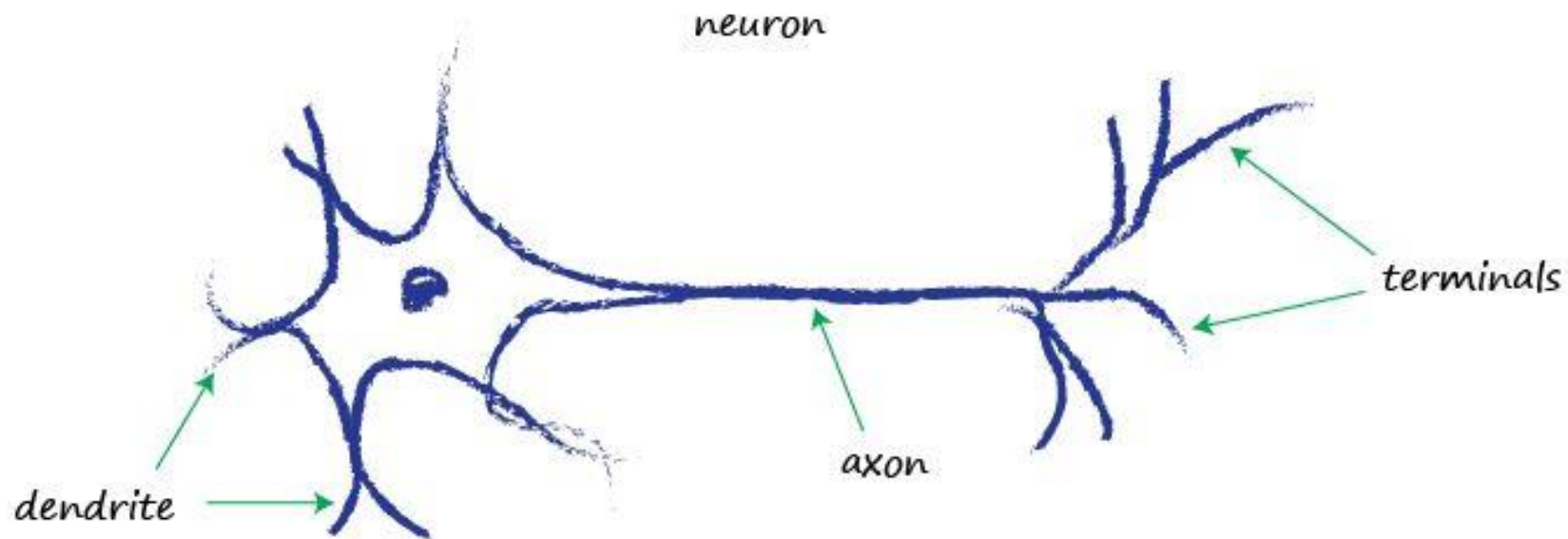


Linear Perceptron

ONE CLASSIFIER IS NOT ENOUGH



NEURONS, NATURAL COMPUTING MACHINE

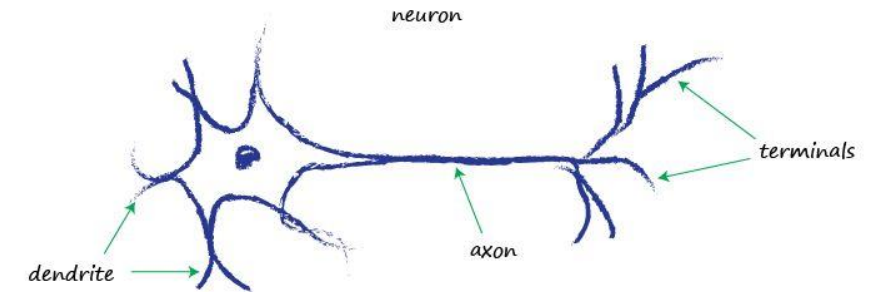


NEURONS, NATURAL COMPUTING MACHINE

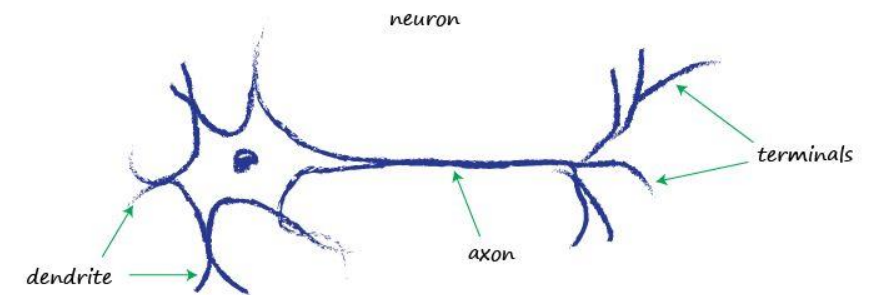
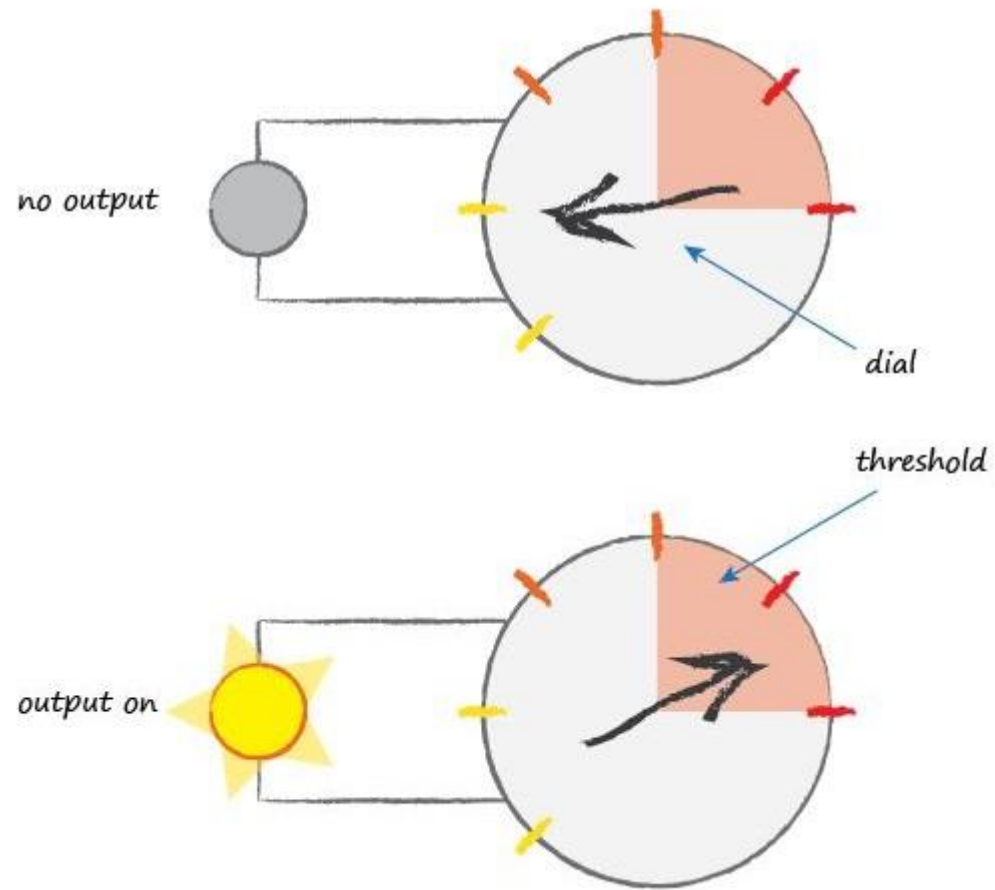
Human Brain 100 billion

Fruit Fly 100,000

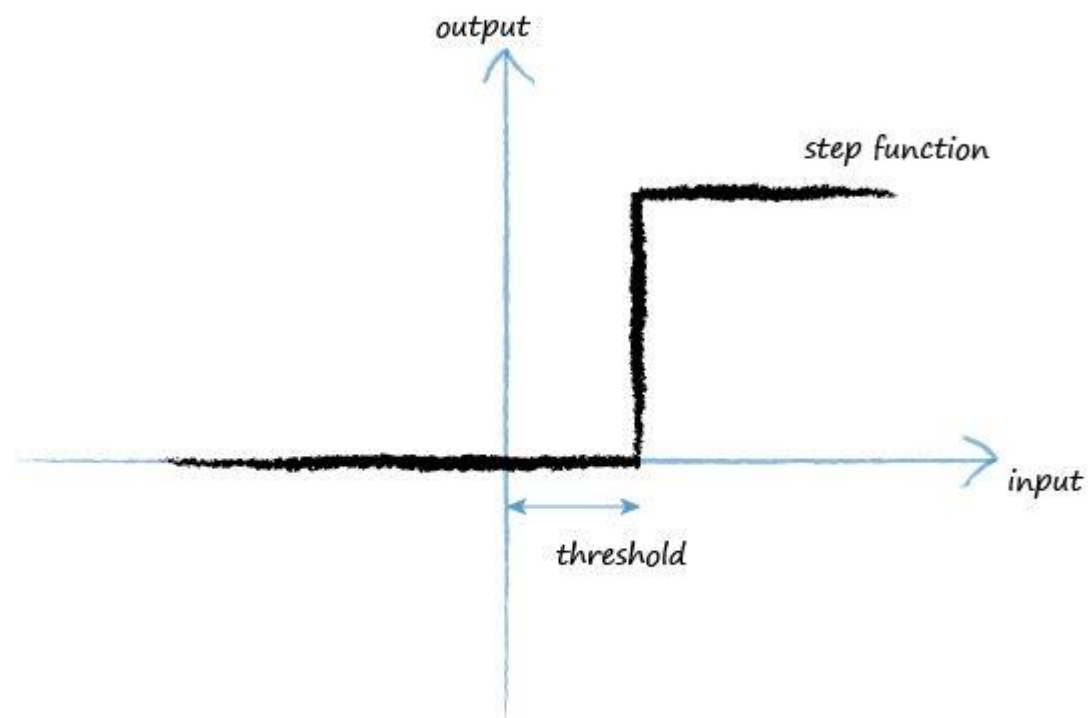
Nematode Worm 302



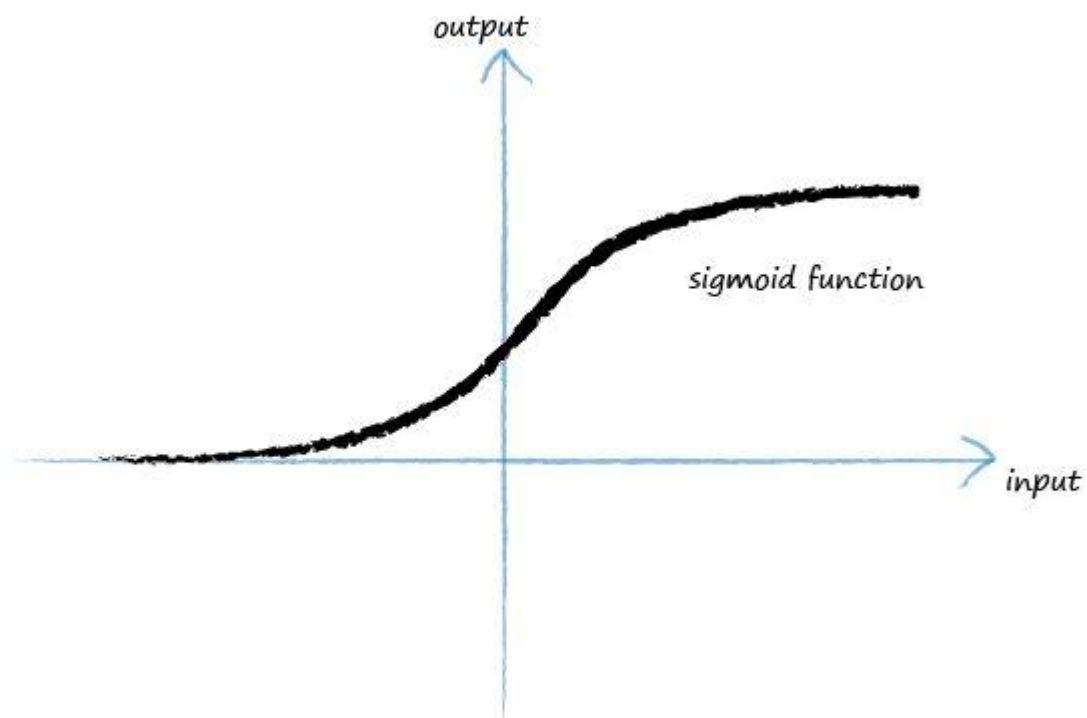
NEURONS, NATURAL COMPUTING MACHINE



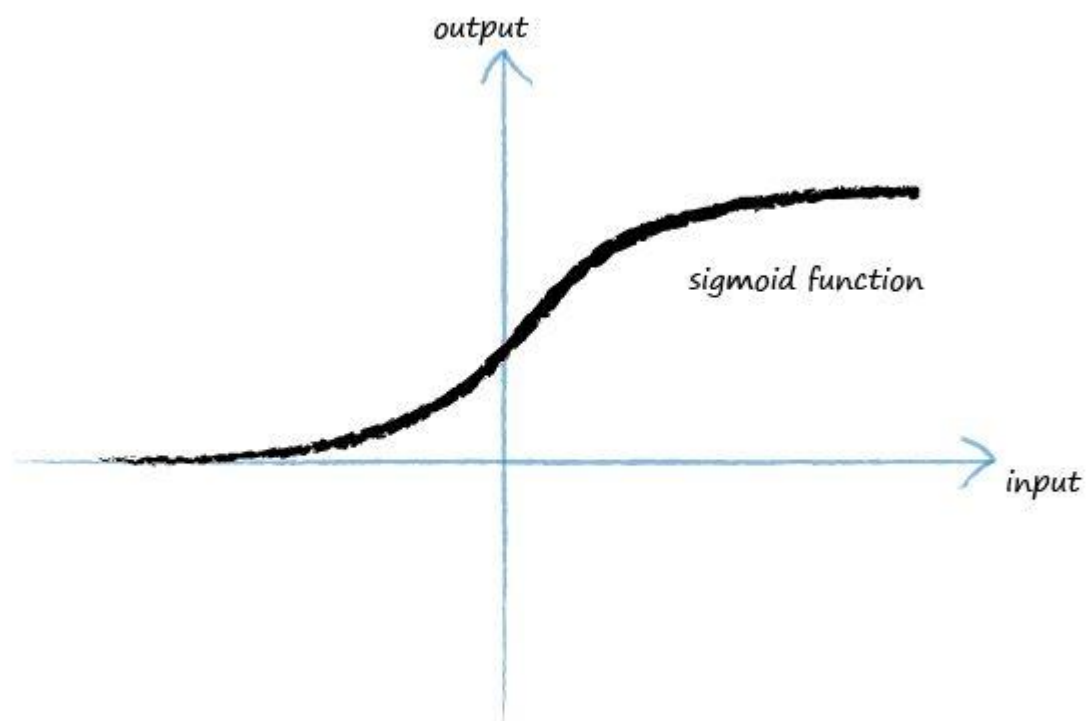
ACTIVATION FUNCTION



ACTIVATION FUNCTION

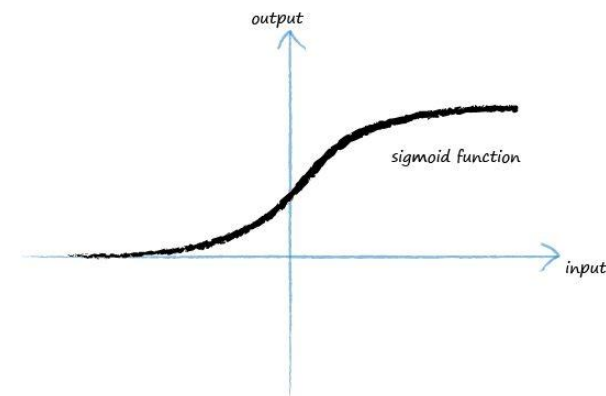
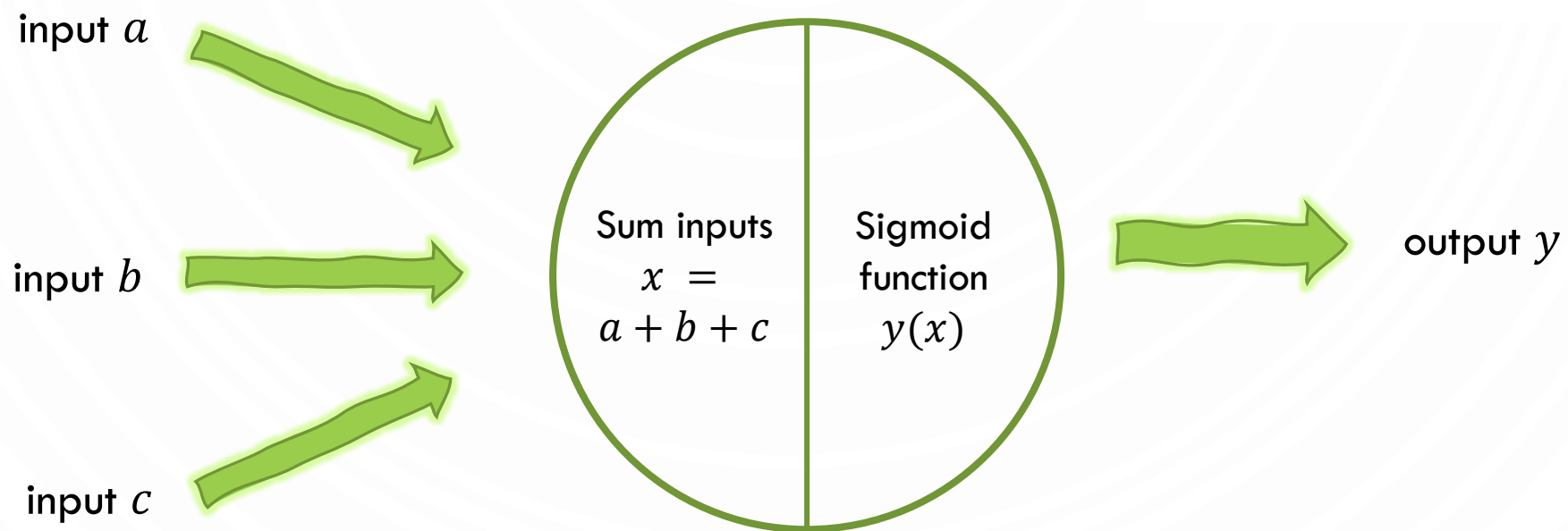


ACTIVATION FUNCTION

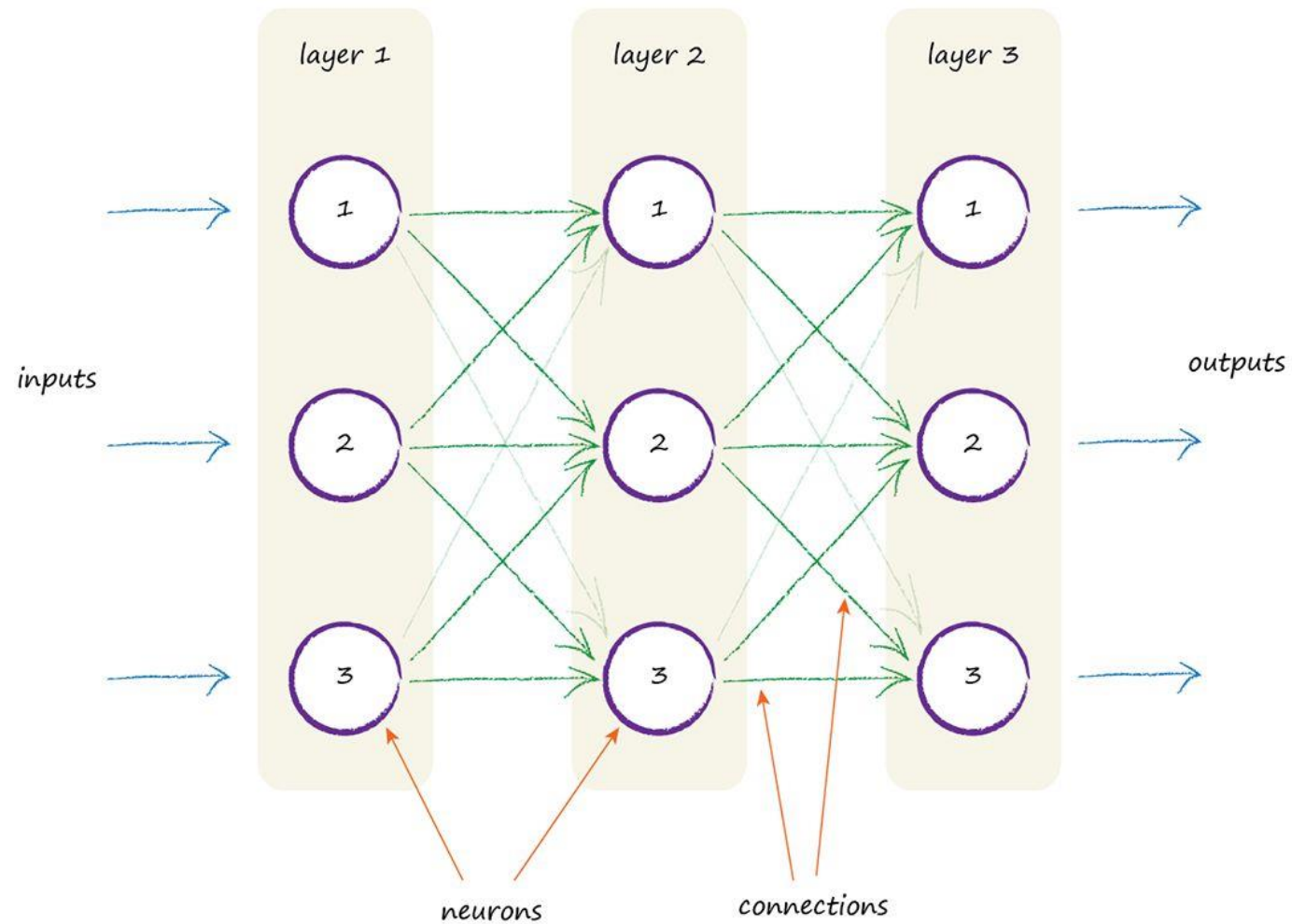


$$y = \frac{1}{1 + e^{-x}}$$

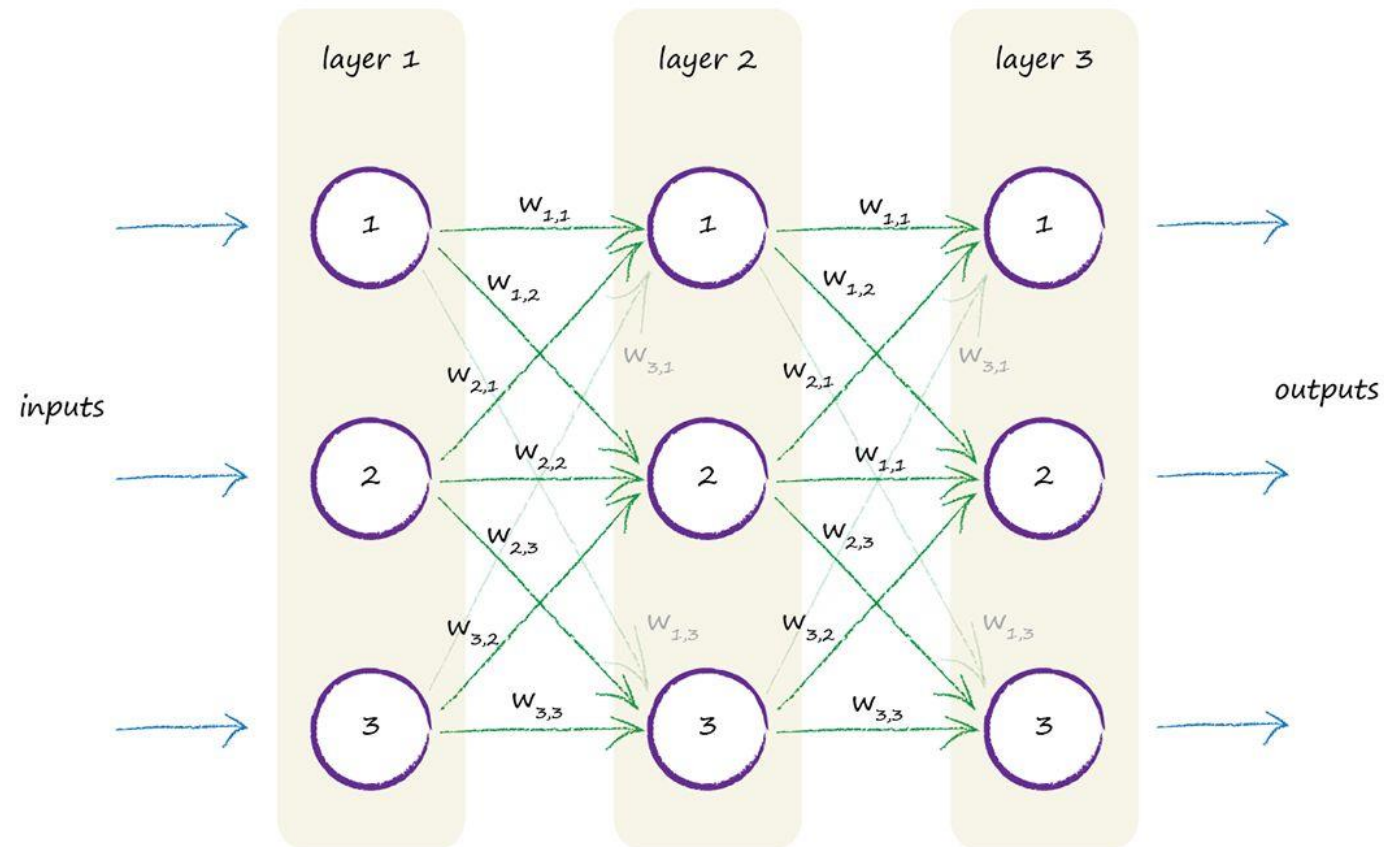
ARTIFICIAL NEURON



ARTIFICIAL NEURON NETWORK



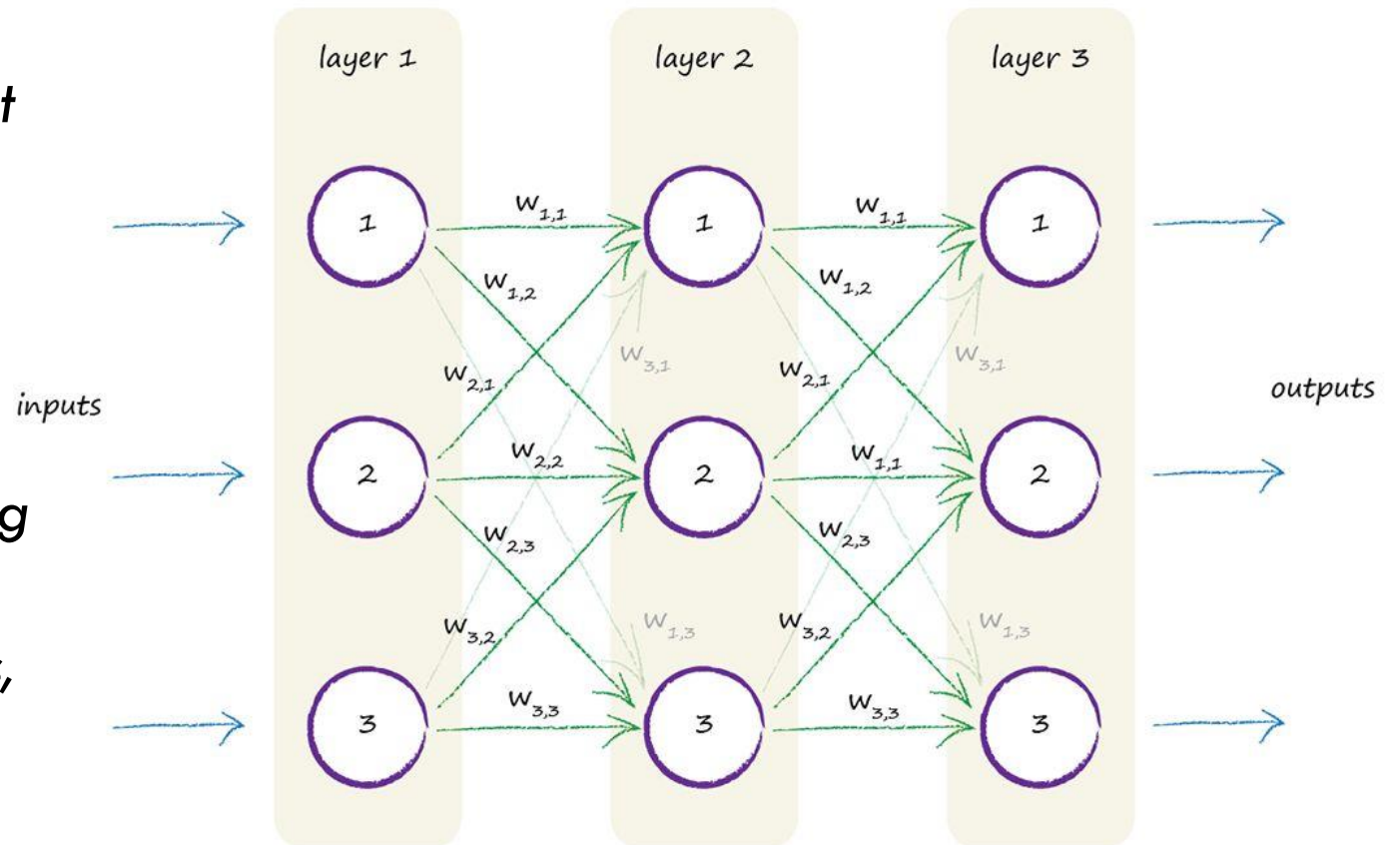
ARTIFICIAL NEURON NETWORK



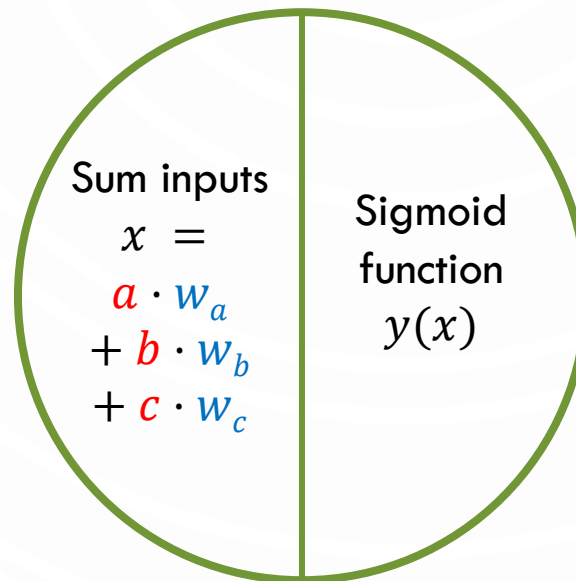
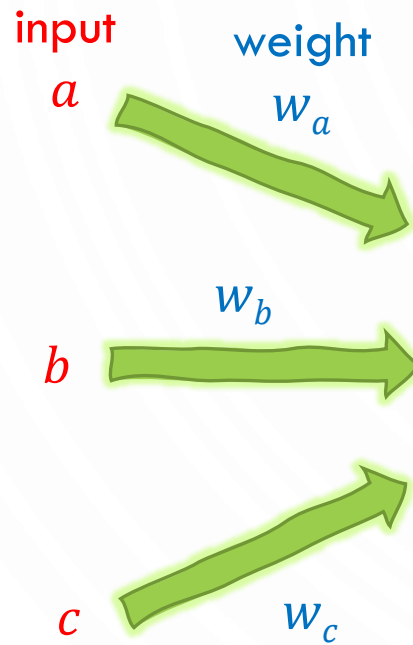
UNIVERSAL APPROXIMATION THEOREM

Neural networks can represent a wide variety of interesting functions when given appropriate weights.

It is capable of approximating any continuous functions between two Euclidean spaces, as long as having enough depths and nodes.

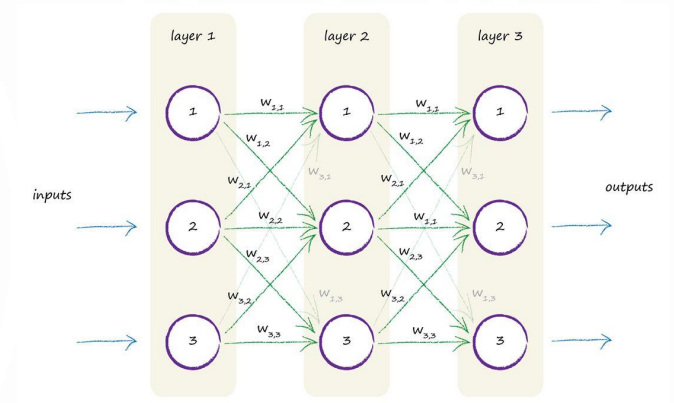


ARTIFICIAL NEURON

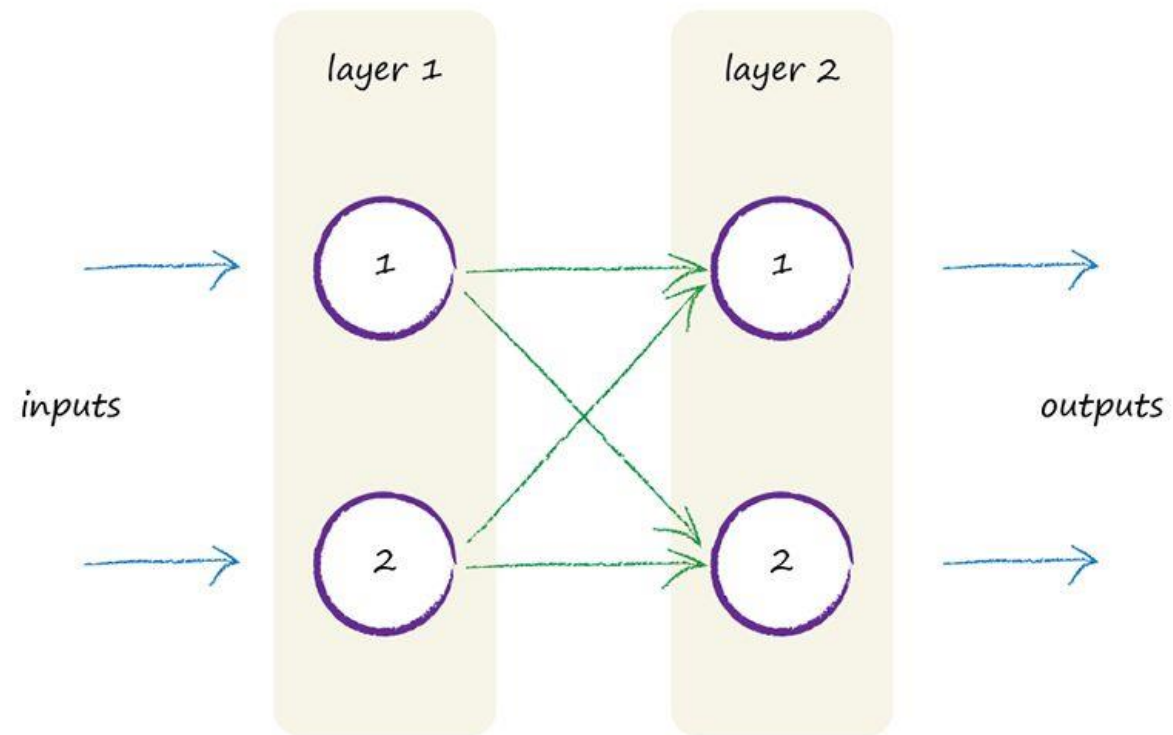


$$y = \text{sigmoid}(a \cdot w_a + b \cdot w_b + c \cdot w_c)$$

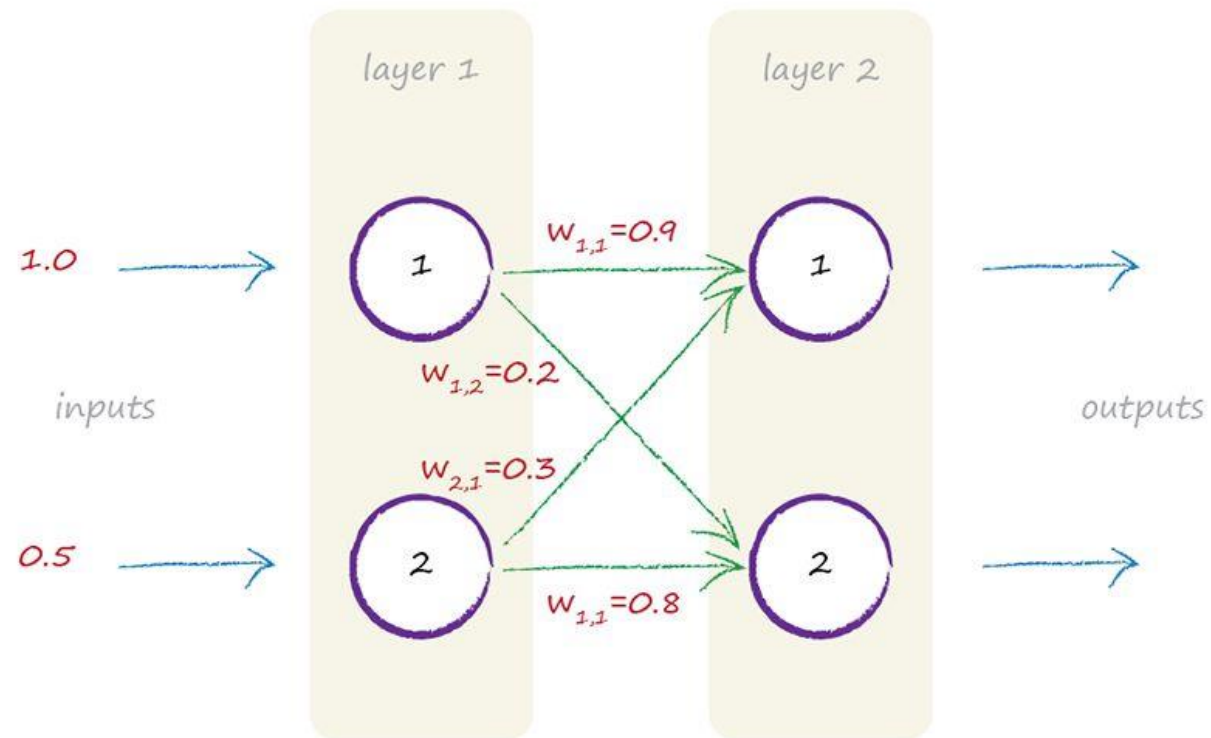
$$\text{sigmoid} = \frac{1}{1 + e^{-x}}$$



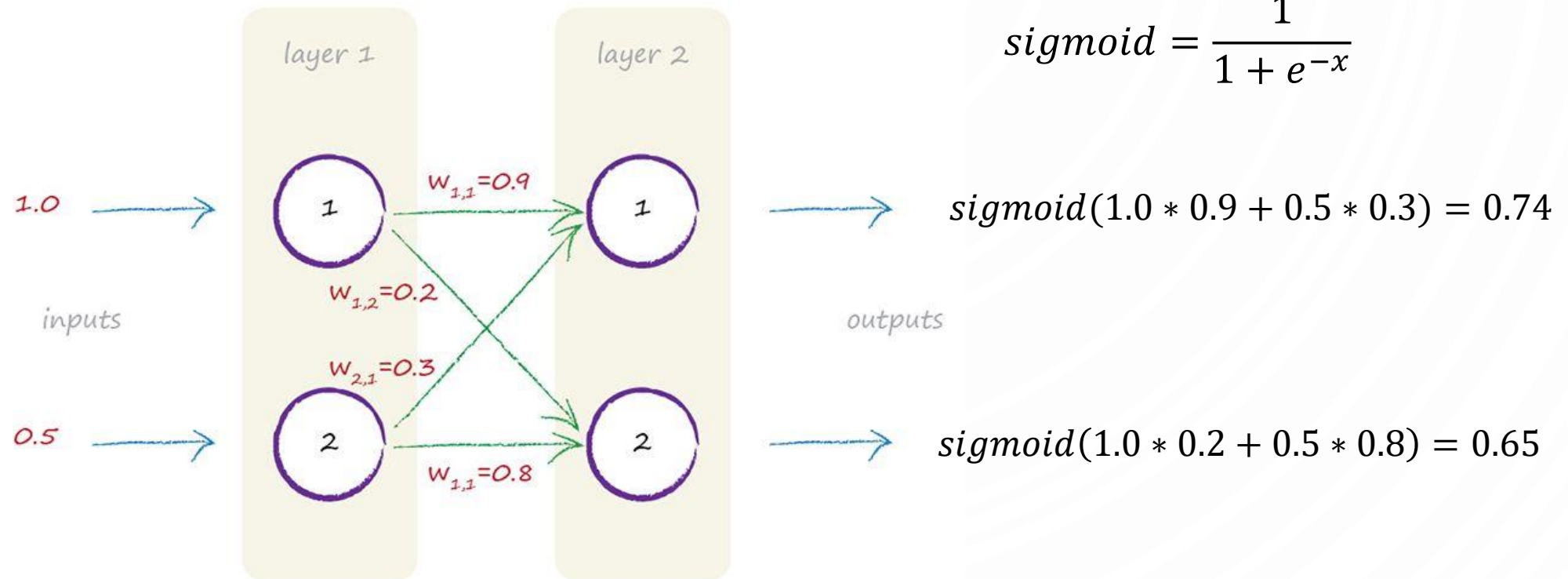
EXAMPLE - FEEDFORWARD SIGNAL



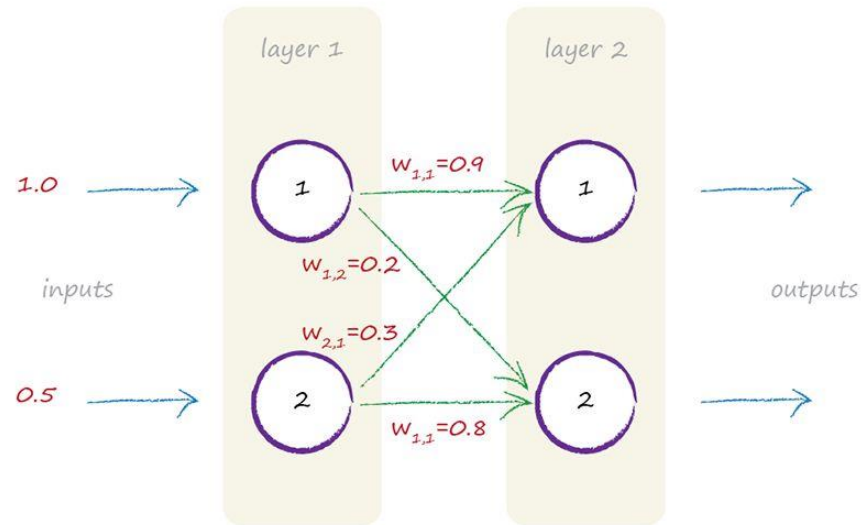
EXAMPLE - FEEDFORWARD SIGNAL



EXAMPLE - FEEDFORWARD SIGNAL



USING MATRIX MULTIPLICATION



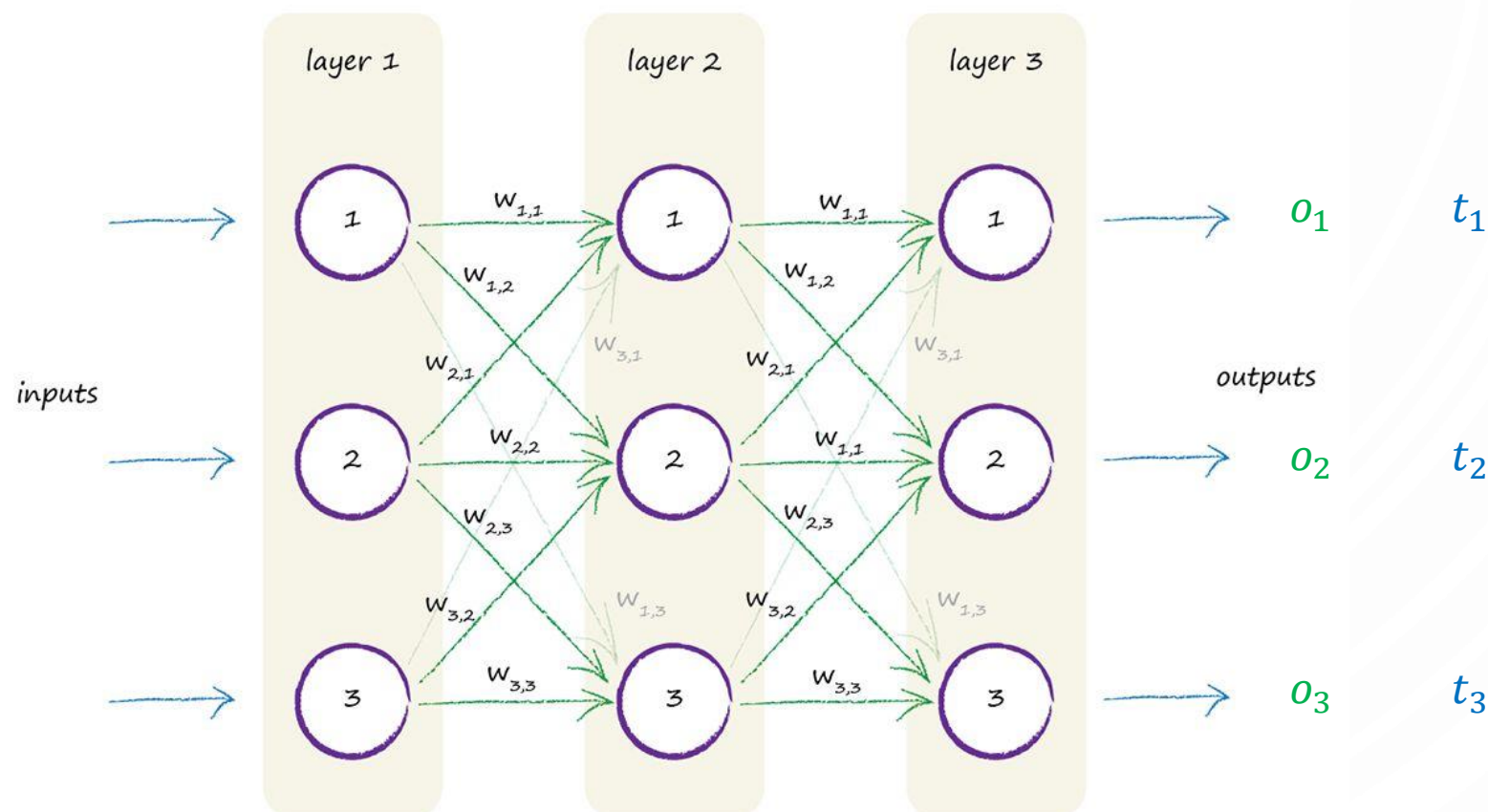
$$\begin{pmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \end{pmatrix} \begin{pmatrix} \text{input_1} \\ \text{input_2} \end{pmatrix} = \begin{pmatrix} (\text{input_1} * w_{1,1}) + (\text{input_2} * w_{2,1}) \\ (\text{input_1} * w_{1,2}) + (\text{input_2} * w_{2,2}) \end{pmatrix}$$

$$X = W \cdot I$$

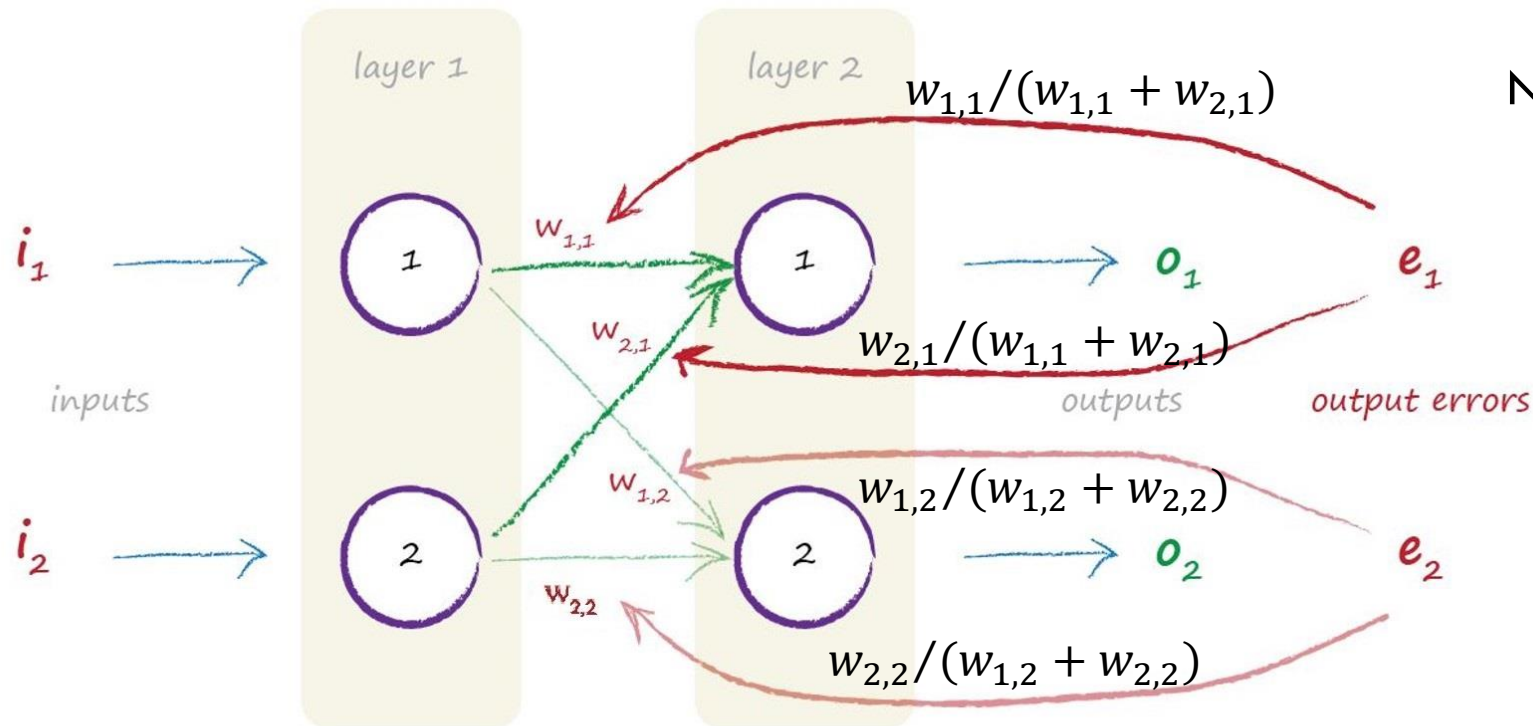
$$O = \text{sigmoid}(X)$$

BACKPROPAGATION OF LOSS

$$o_k = \frac{1}{1 + e^{-\sum_{j=1}^3 (w_{j,k} \cdot \frac{1}{1 + e^{-\sum_{i=1}^3 (w_{i,j} \cdot x_i)})}}$$



BACKPROPAGATION OF LOSS



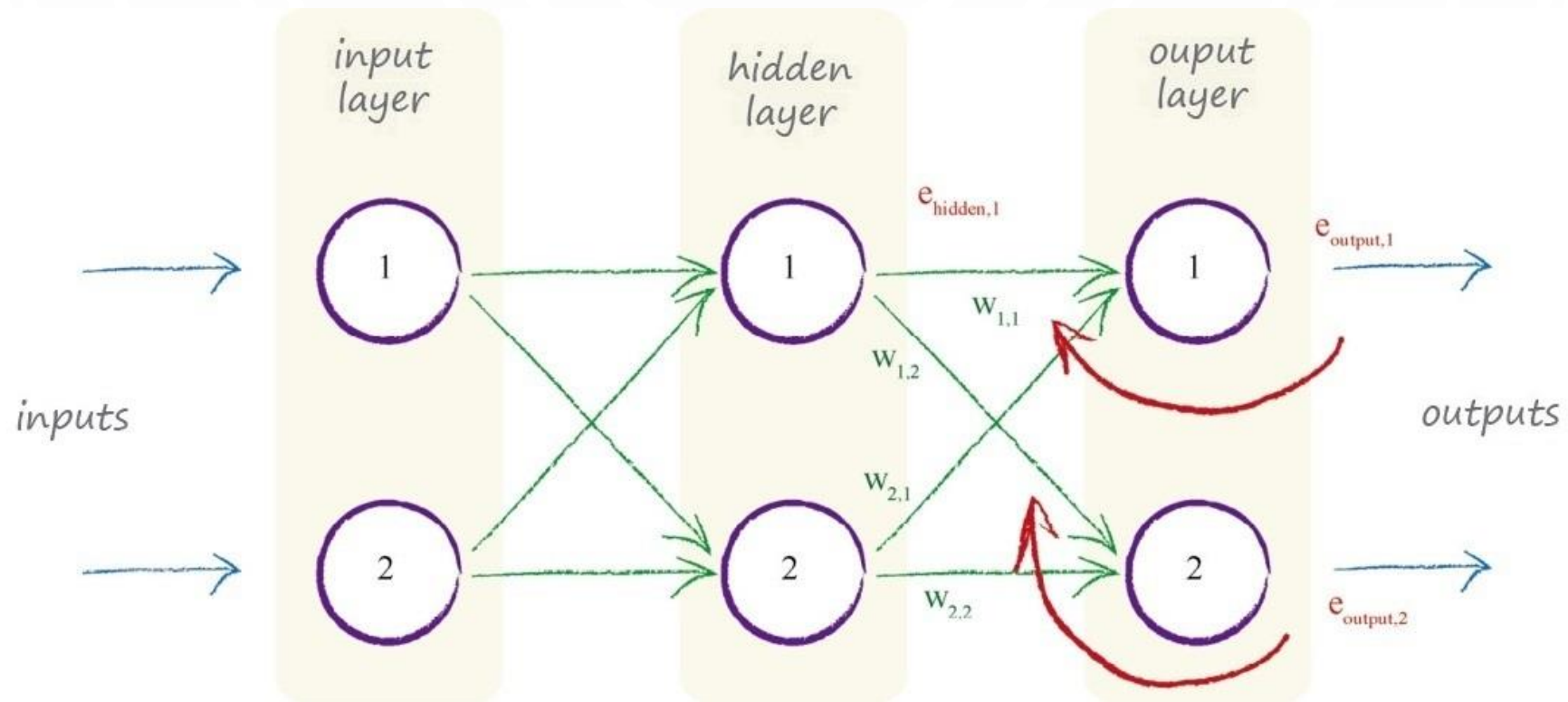
Node Loss = (actual - real)²

$$e_{2,k} = (t_k - o_k)^2$$

$$e_{1,1} = e_{2,1} * \frac{w_{1,1}}{w_{1,1} + w_{2,1}} +$$

$$e_{2,2} * \frac{w_{1,2}}{w_{1,2} + w_{2,2}}$$

BACKPROPAGATION OF LOSS



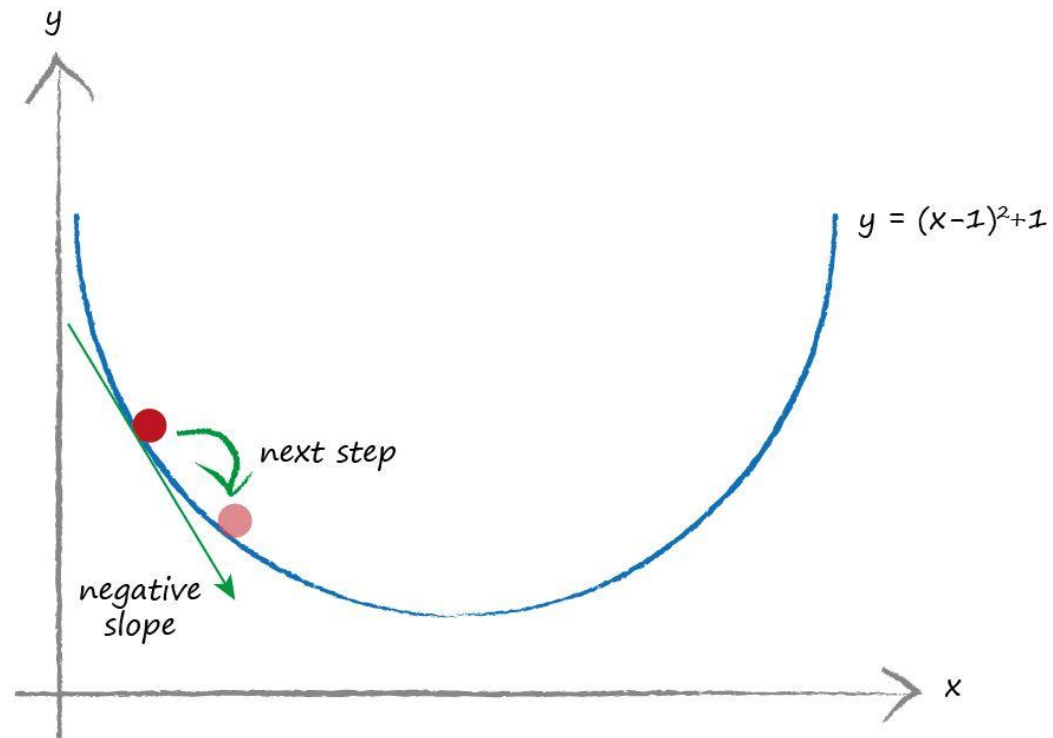
MATRIX MULTIPLICATION AGAIN

$$e_h = \begin{pmatrix} \frac{w_{1,1}}{w_{1,1} + w_{2,1}} & \frac{w_{1,2}}{w_{1,2} + w_{2,2}} \\ \frac{w_{2,1}}{w_{1,1} + w_{2,1}} & \frac{w_{2,2}}{w_{1,2} + w_{2,2}} \end{pmatrix} \cdot \begin{pmatrix} e_{o,1} \\ e_{o,2} \end{pmatrix}$$

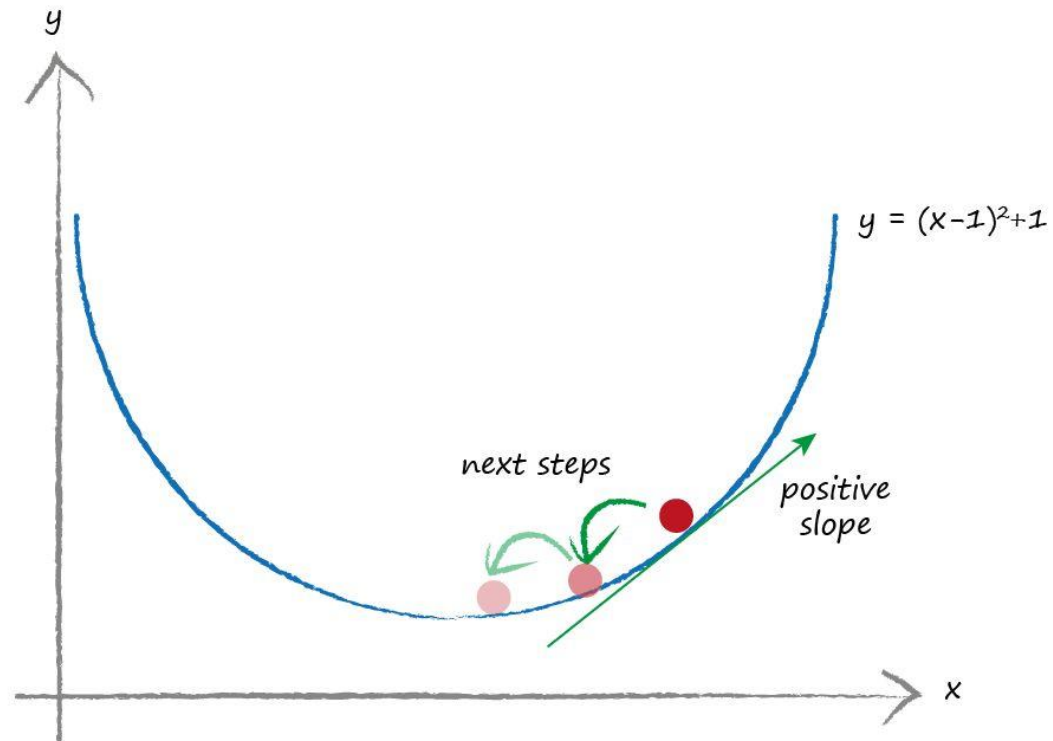


$$e_h = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix} \cdot \begin{pmatrix} e_{o,1} \\ e_{o,2} \end{pmatrix} = w^T \cdot e_o$$

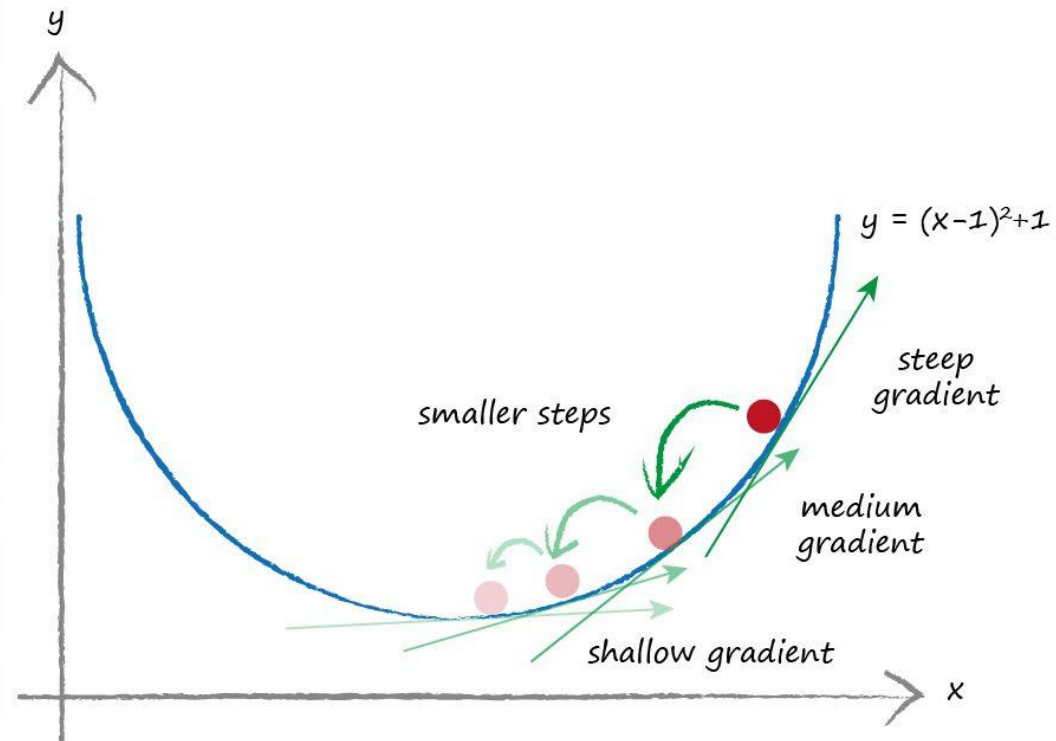
UPDATE WEIGHTS - GRADIENT DESCENT



UPDATE WEIGHTS - GRADIENT DESCENT



UPDATE WEIGHTS - GRADIENT DESCENT



UPDATE WEIGHTS - GRADIENT DESCENT

$$\begin{aligned}\frac{\partial E}{\partial w_{j,k}} &= \frac{\partial (t_k - o_k)^2}{\partial w_{j,k}} = \frac{\partial (t_k - o_k)^2}{\partial o_k} \cdot \frac{\partial o_k}{\partial w_{j,k}} = -2(t_k - o_k) \cdot \frac{\partial o_k}{\partial w_{j,k}} \\&= -2(t_k - o_k) \cdot \frac{\partial \text{sigmoid}(\sum_j w_{j,k} \cdot o_j)}{\partial w_{j,k}} \\&= -2(t_k - o_k) \cdot \text{sigmoid}(\sum_j w_{j,k} \cdot o_j) \left(1 - \text{sigmoid}(\sum_j w_{j,k} \cdot o_j)\right) \cdot \frac{\partial (\sum_j w_{j,k} \cdot o_j)}{\partial w_{j,k}} \\&= -2(t_k - o_k) \cdot \text{sigmoid}(\sum_j w_{j,k} \cdot o_j) \left(1 - \text{sigmoid}(\sum_j w_{j,k} \cdot o_j)\right) \cdot o_j\end{aligned}$$

$$\Delta w_{j,k} = -\alpha \cdot \frac{\partial E}{\partial w_{j,k}} = -\alpha \cdot (-E_k \cdot o_k (1 - o_k) \cdot o_j^T)$$

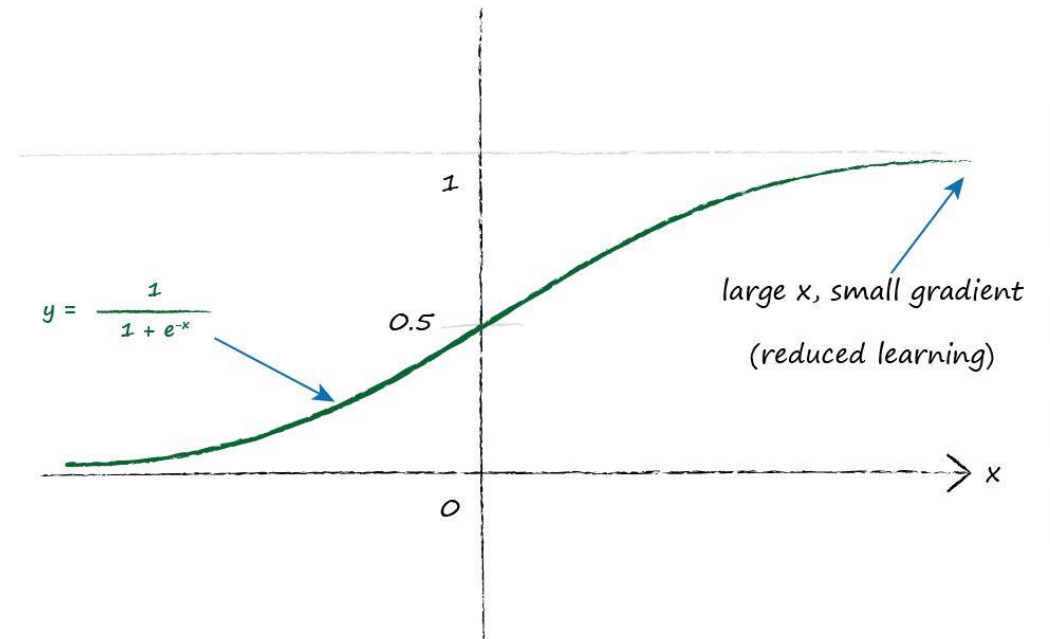
INPUT / OUTPUT & INITIAL WEIGHTS

$$-2(t_k - o_k) \cdot \text{sigmoid}(\sum_j w_{j,k} \cdot o_j) (1 - \text{sigmoid}(\sum_j w_{j,k} \cdot o_j)) \cdot o_j$$

Input/Output 0.01 ~ 0.99

Initial Weights $\frac{-1}{\sqrt{n}} \sim \frac{1}{\sqrt{n}}$

n = number of nodes in target layer



* *Break Symmetry*: never set initial weights to the same constant value, especially no zero.

The image features a light gray background with a subtle pattern of concentric circles. In the four corners, there are decorative purple circuit-like lines with small circles at various points, resembling a stylized electronic board.

CODE EXAMPLE - NUMPY



MUDSS
Data Science Society

THANK YOU!

References:

- Rashid, Tariq. *Make your own neural network*. CreateSpace Independent Publishing Platform, 2016.