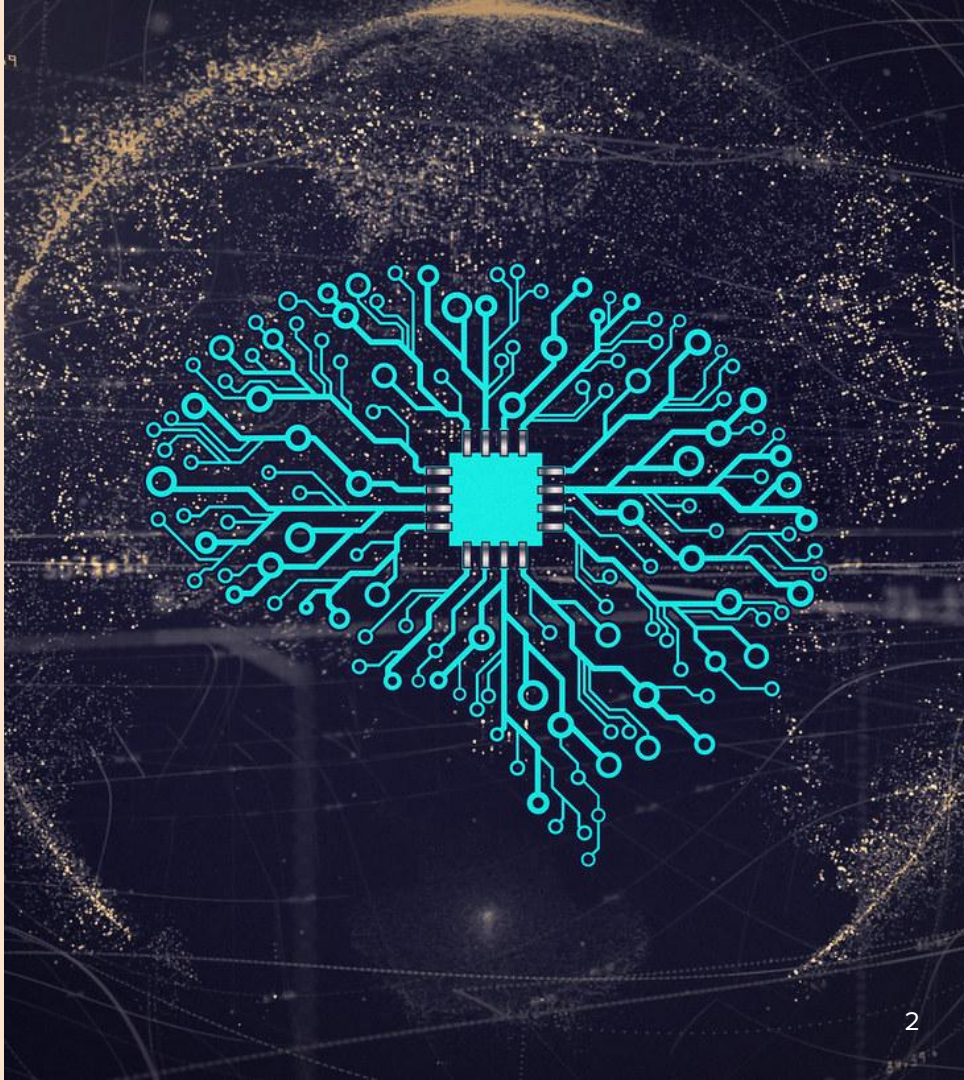
Introduction to Neural Network

Dr. Paisit Khanarsa

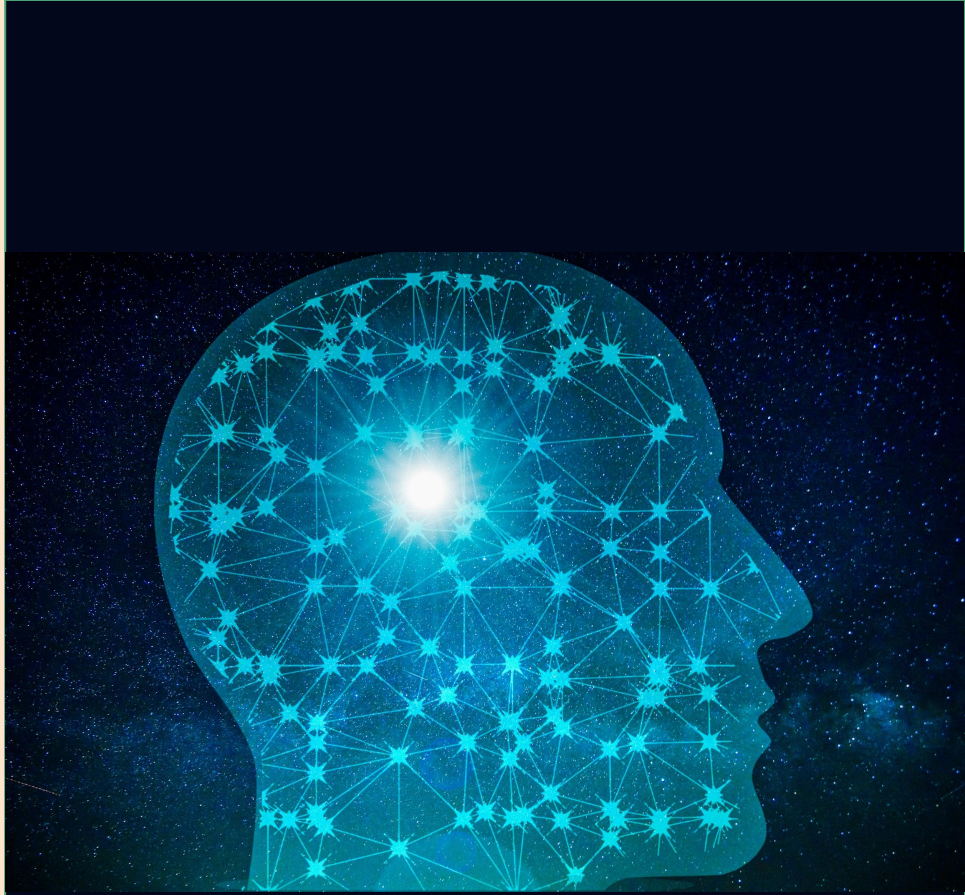
Fibo, Kmutt

Outline

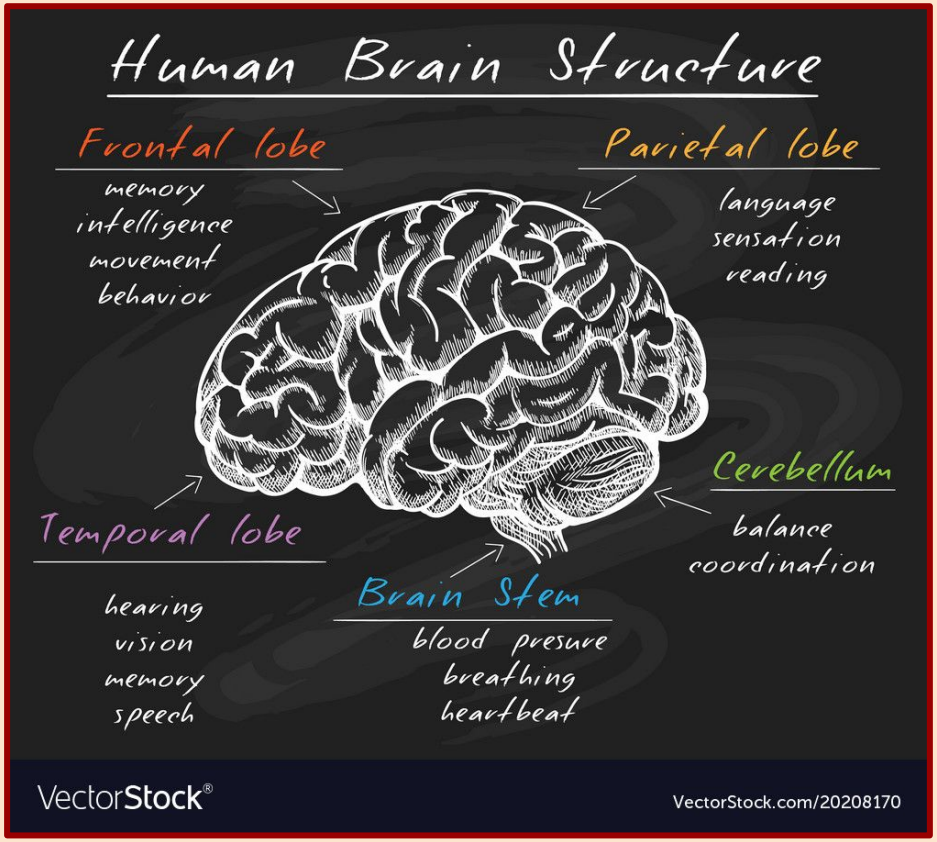
- Introduction to Neural Network
- Neuron and artificial neural network
- Neural network decision boundary



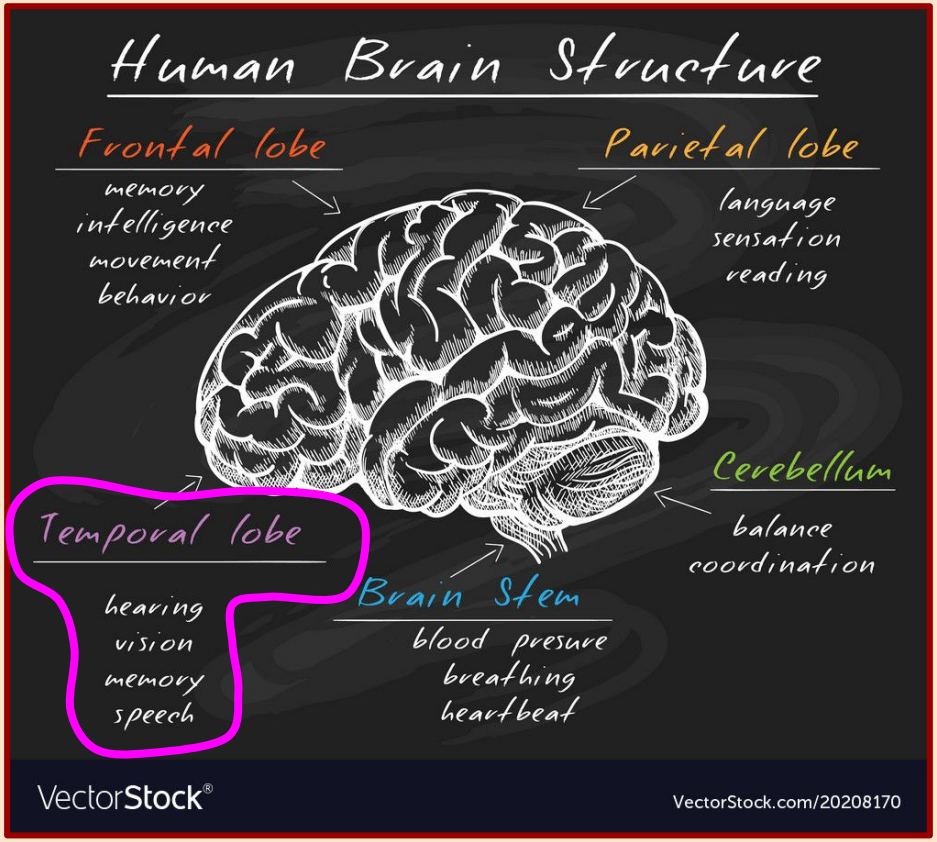
Introduction to Neural Network



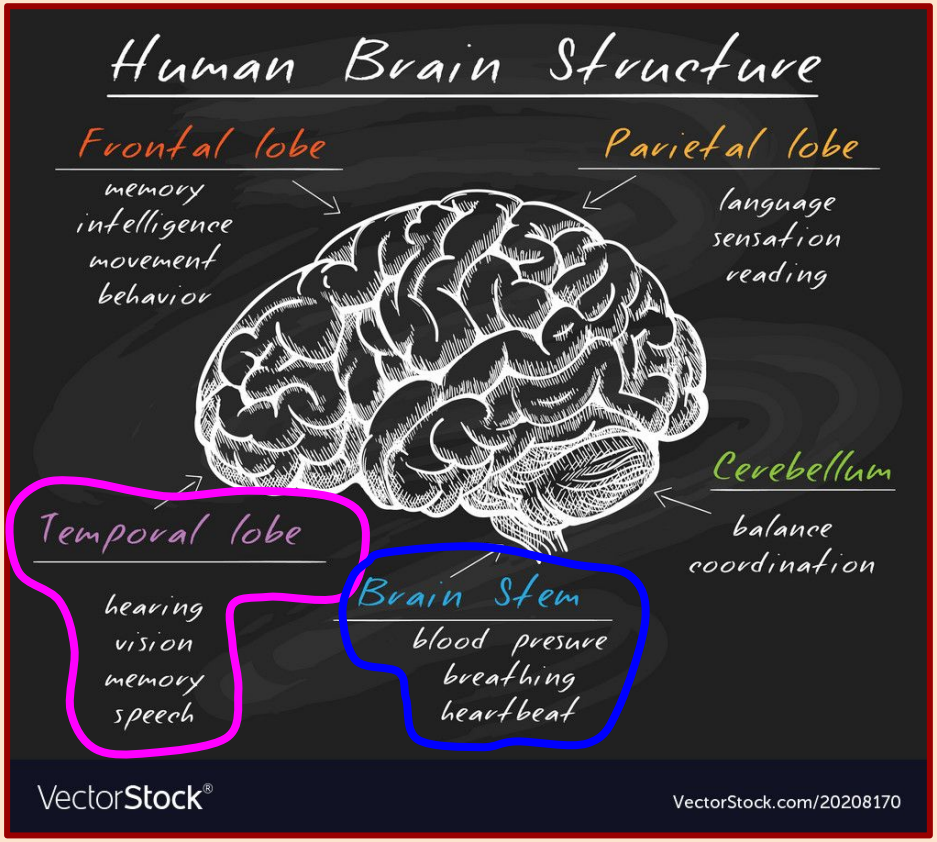
Natural intelligence = Neural network



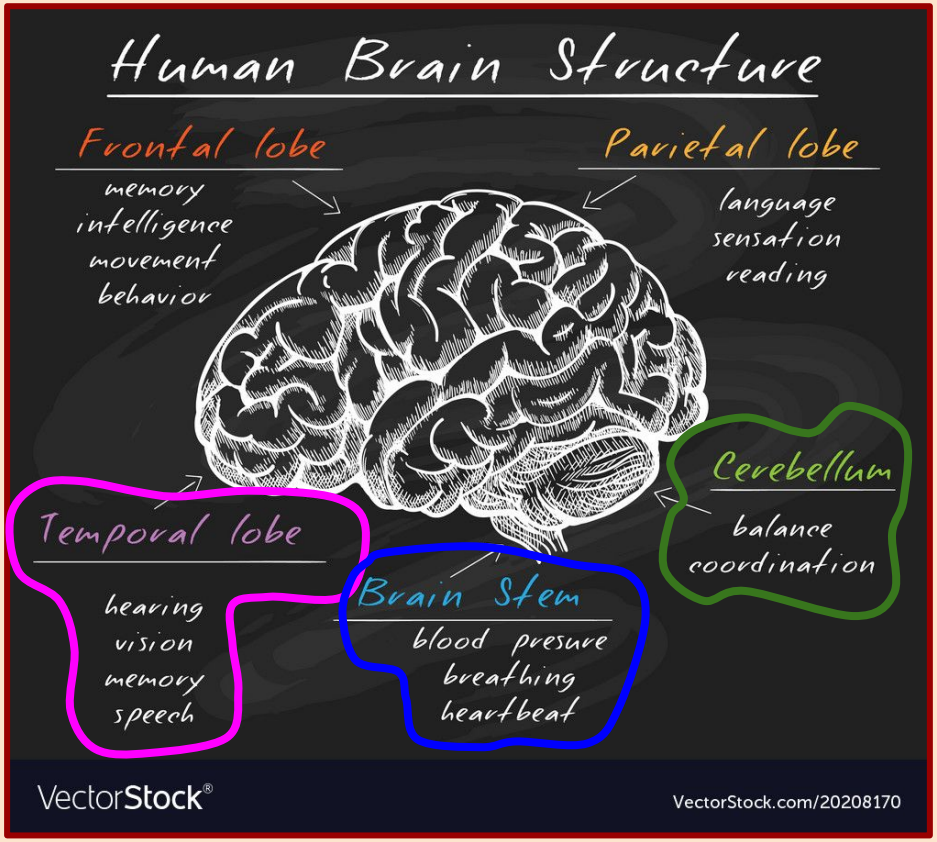
Natural intelligence = Neural network



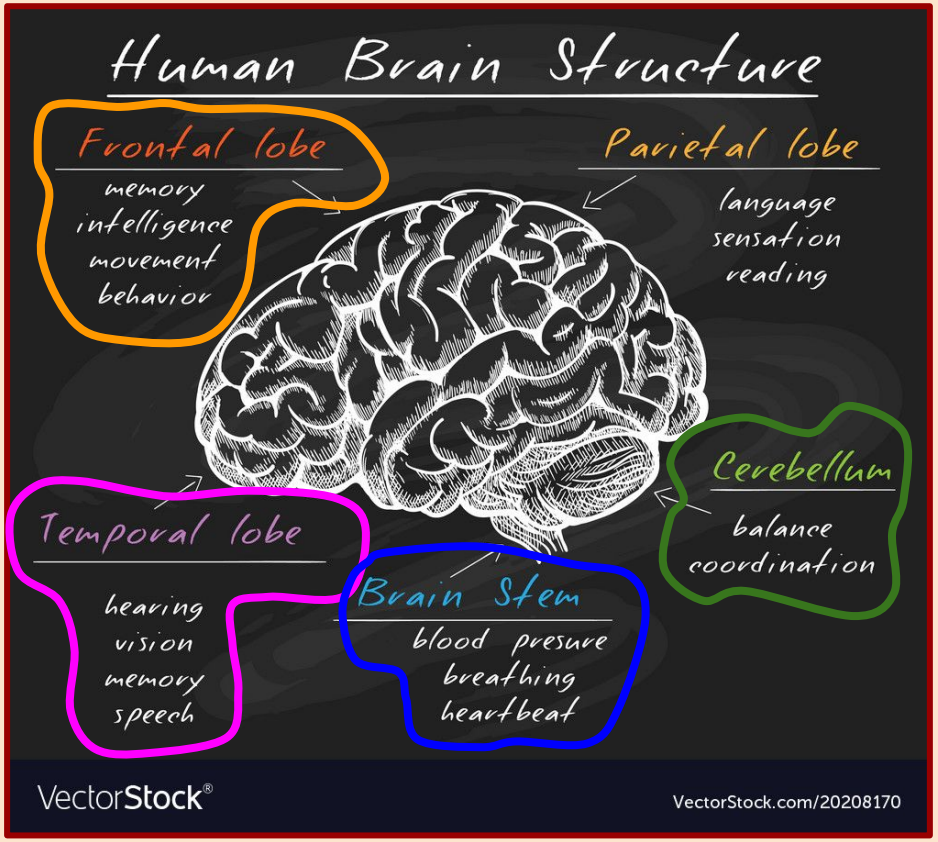
Natural intelligence = Neural network



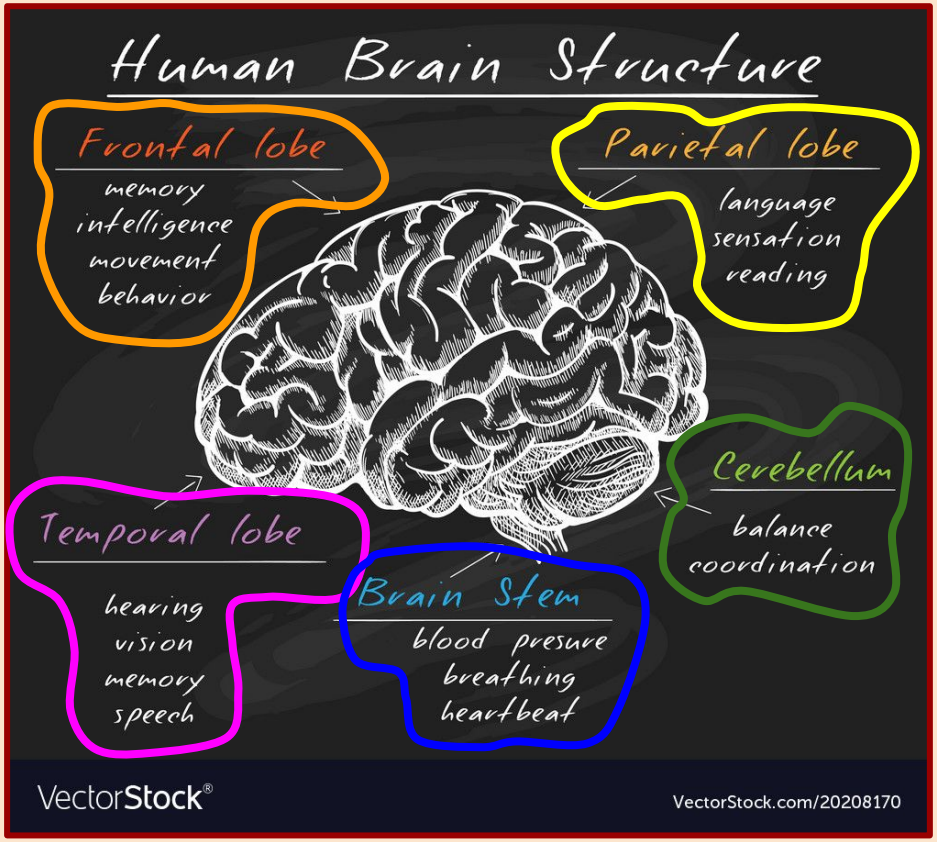
Natural intelligence = Neural network



Natural intelligence = Neural network

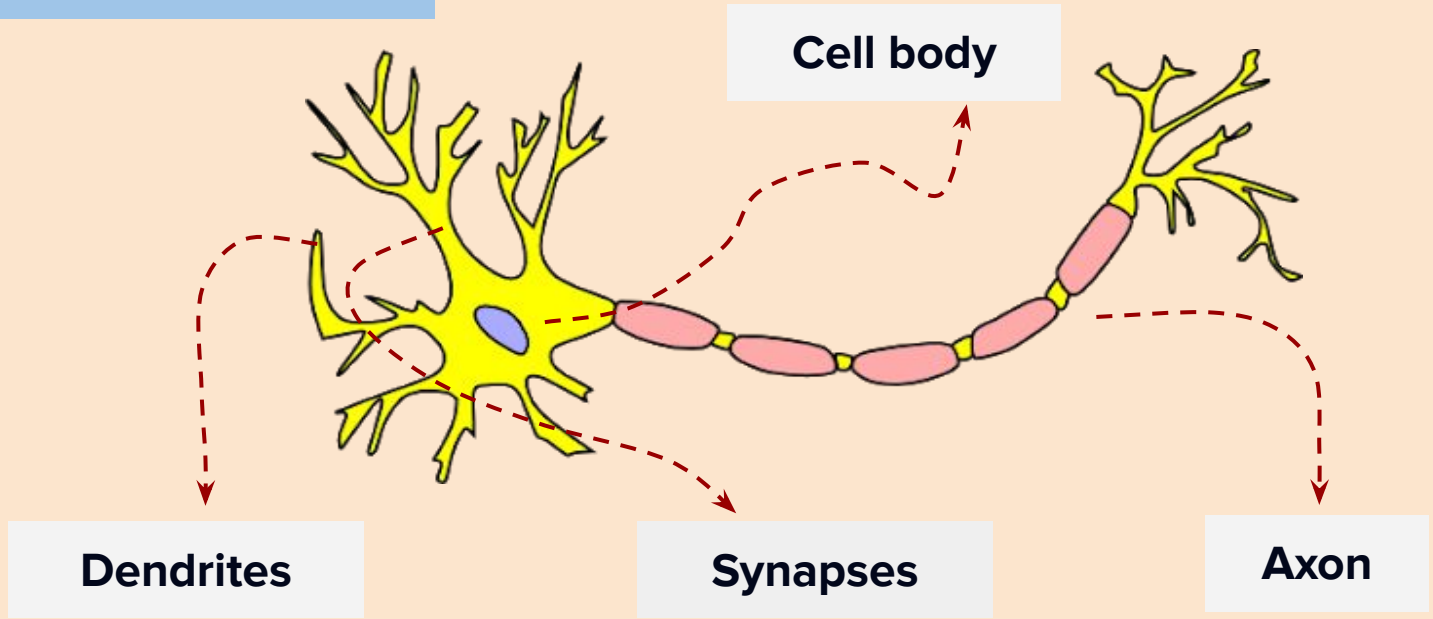


Natural intelligence = Neural network



Human neuron cell

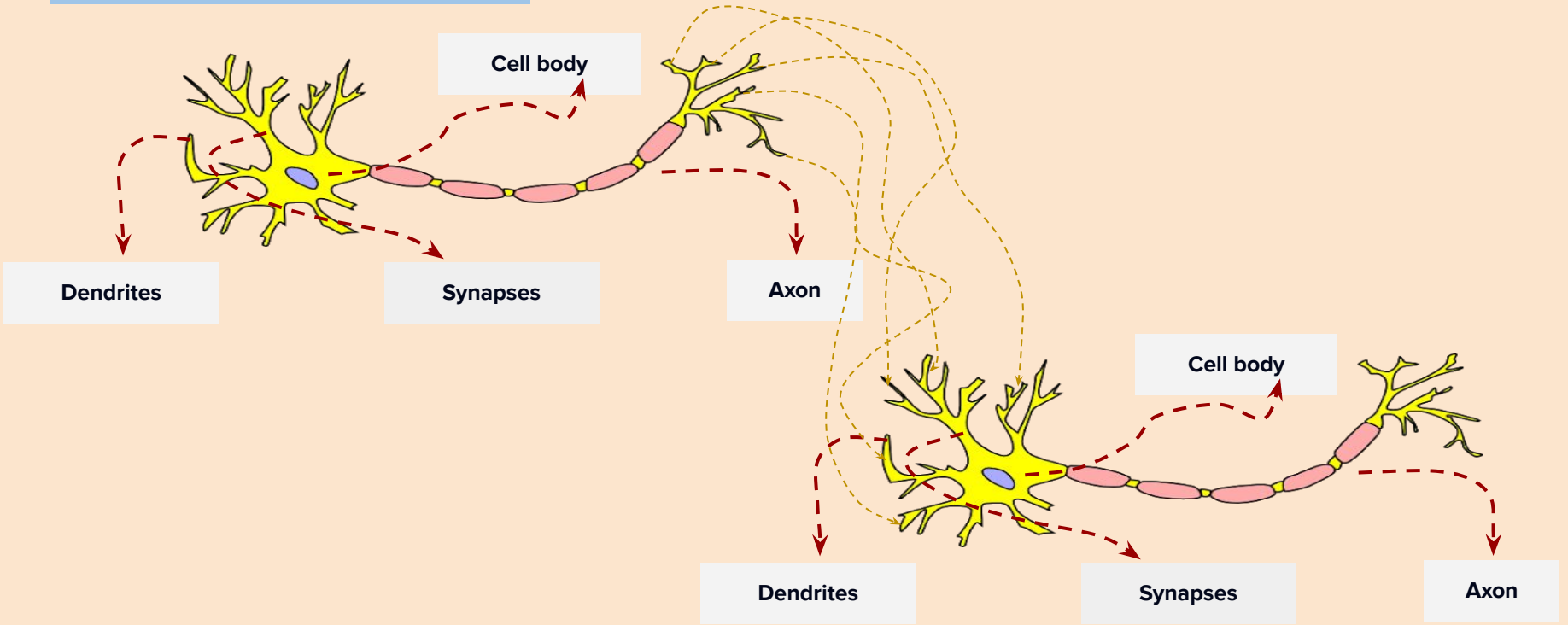
Biological neuron



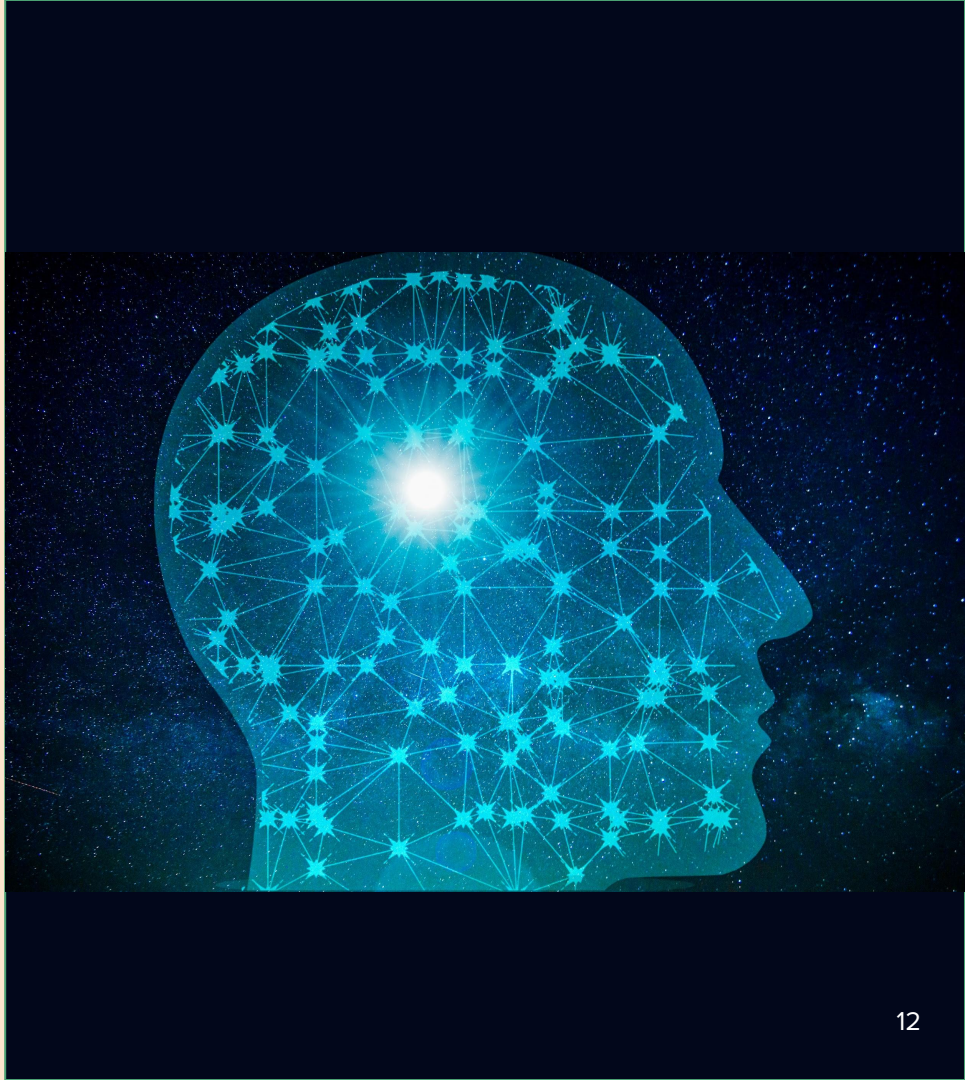
Brain’s computational unit

Human neuron interaction

Biological neuron

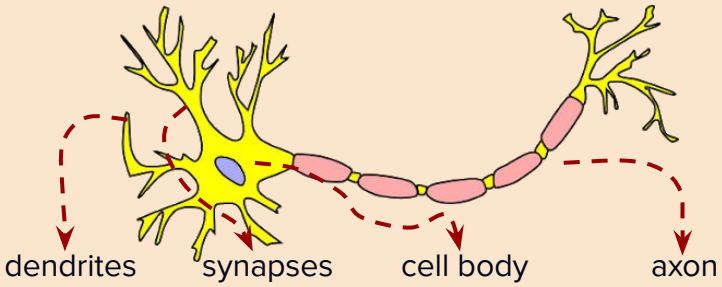


Neuron and artificial neural network

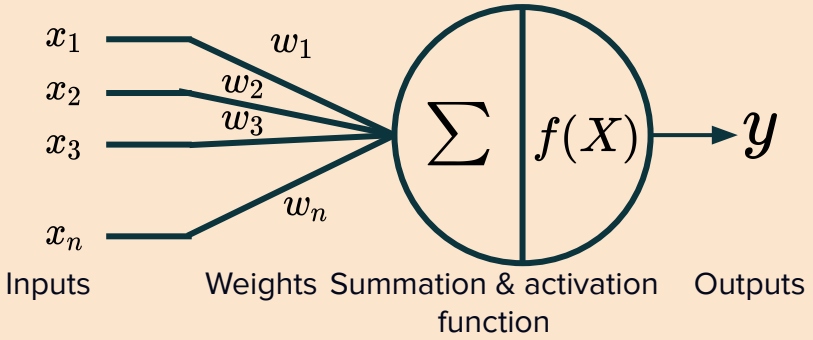


An artificial neuron

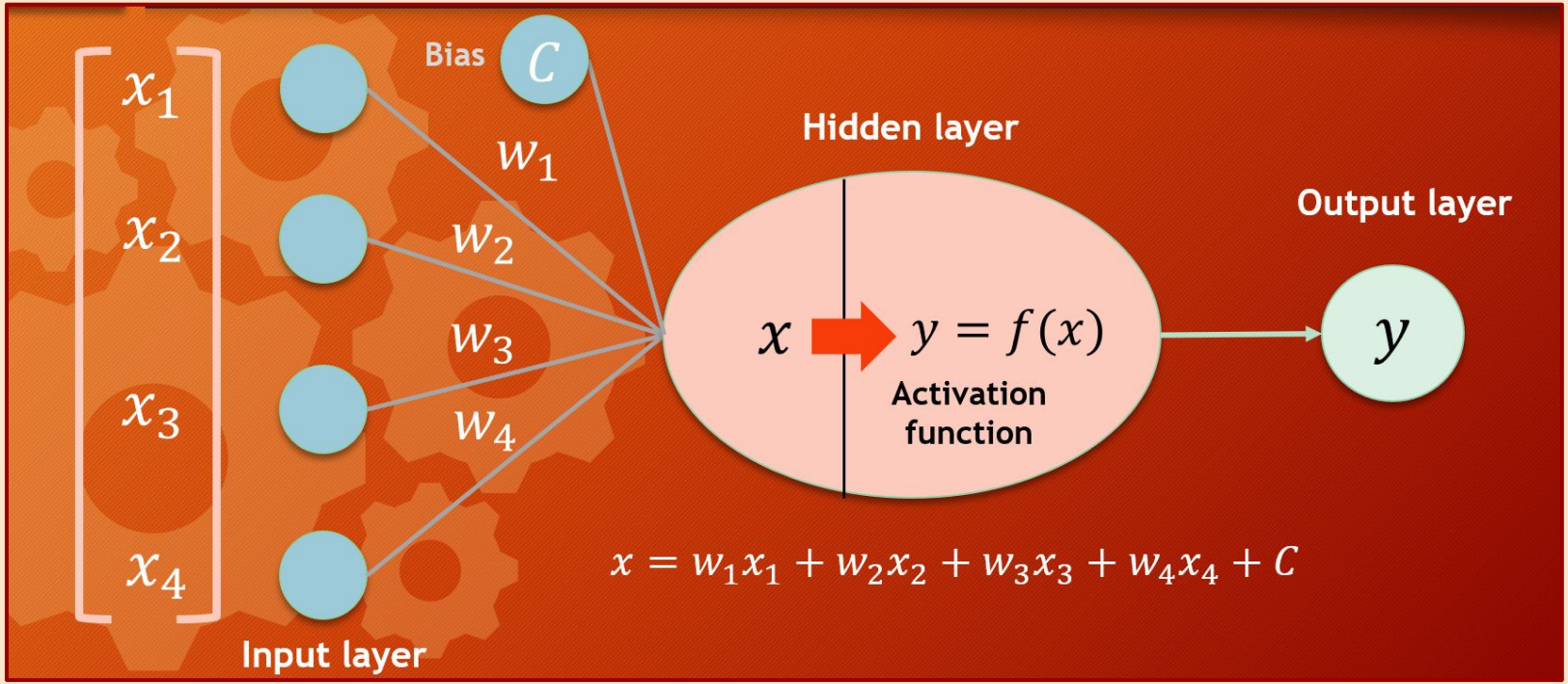
Biological neuron



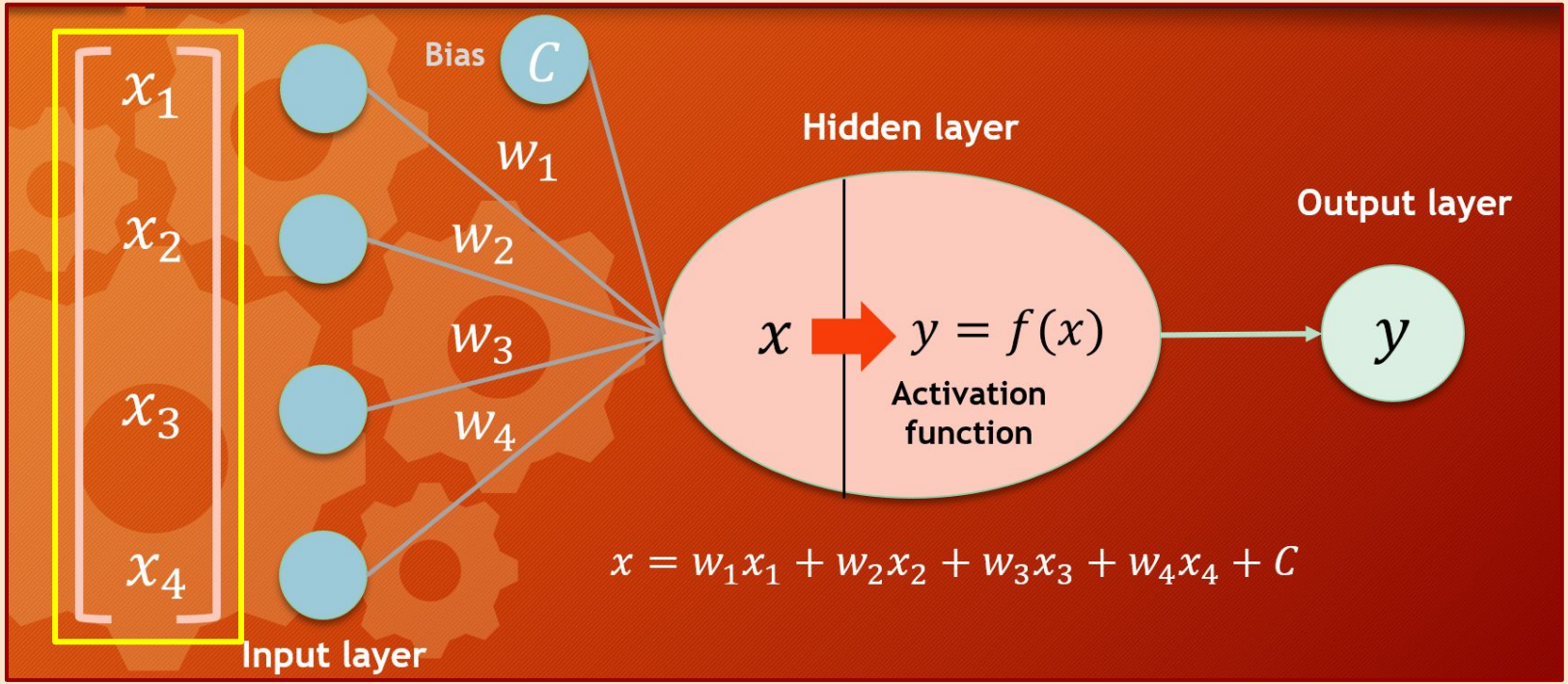
Artificial neuron



An artificial neuron

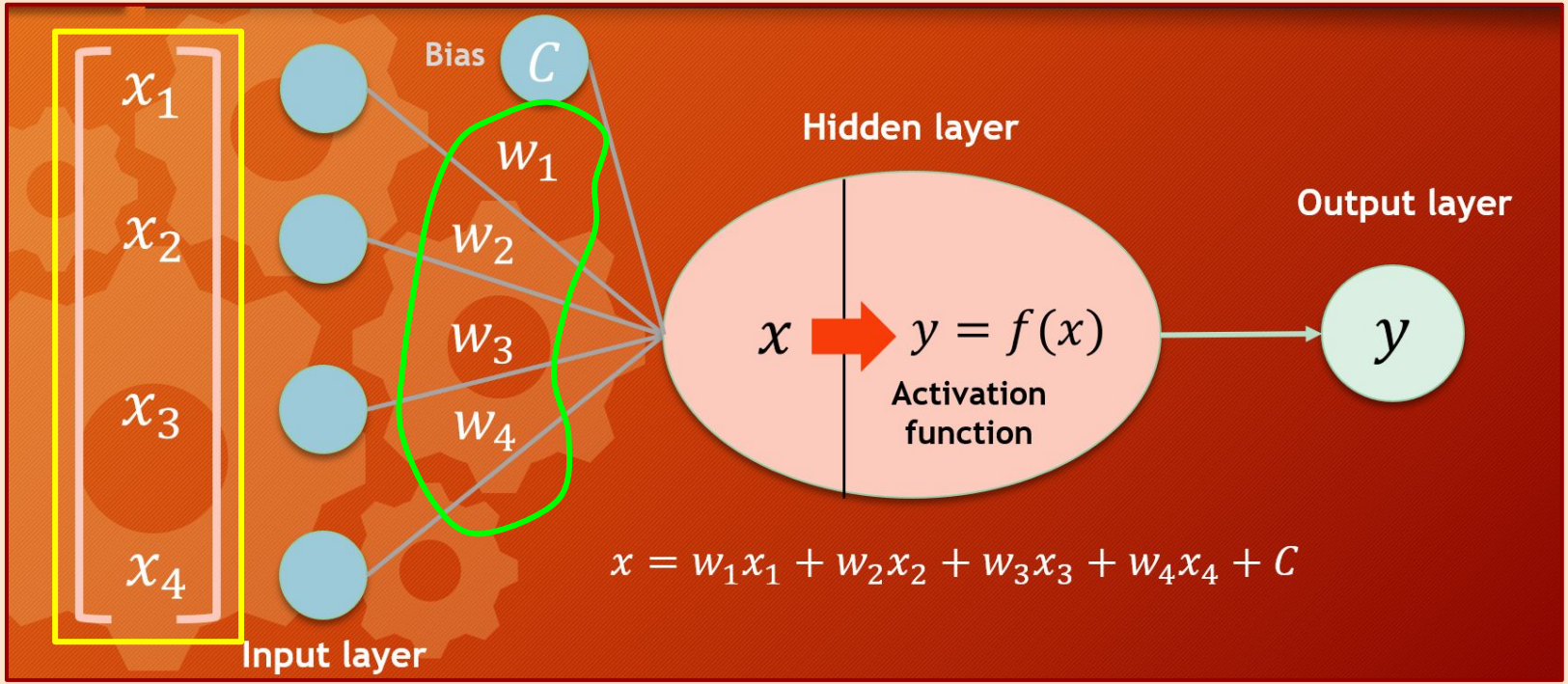


An artificial neuron



Feature

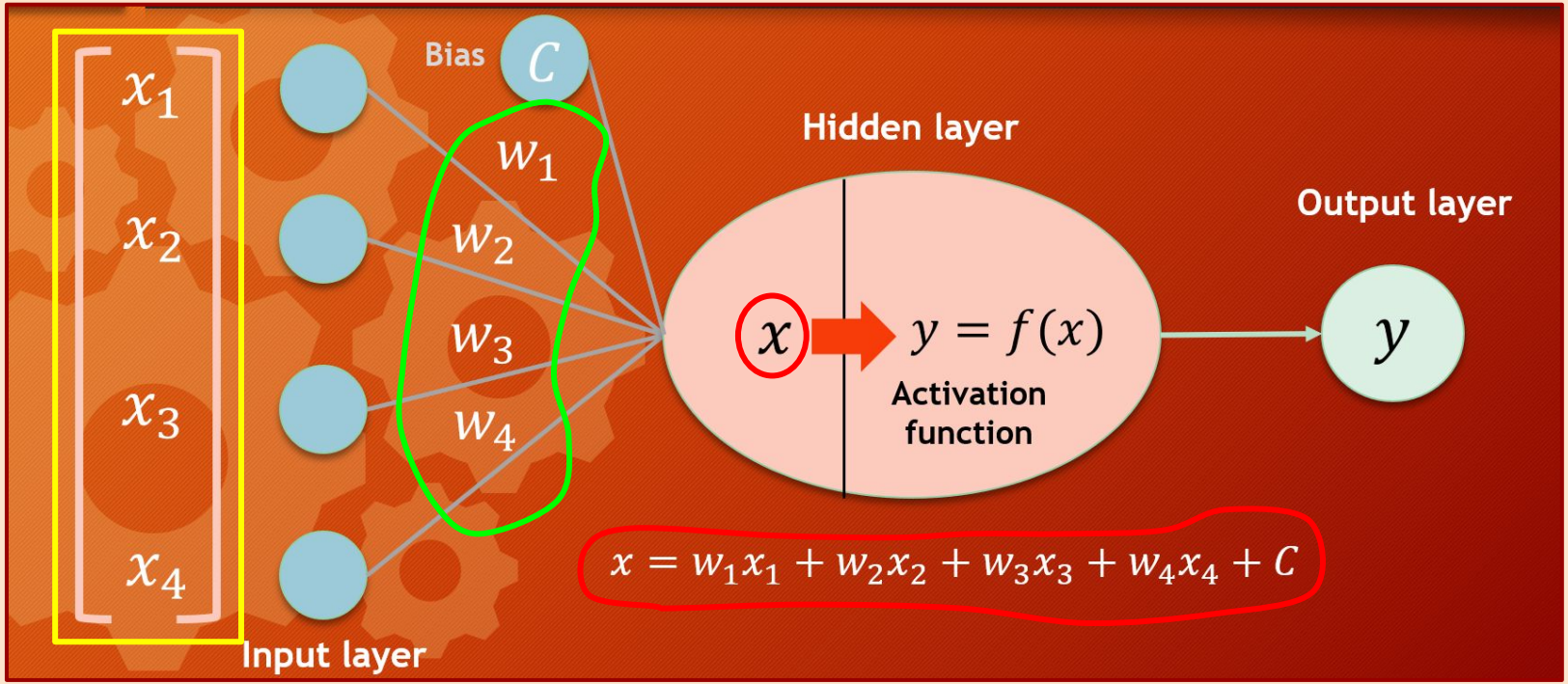
An artificial neuron



Feature

Weights (parameters)

An artificial neuron

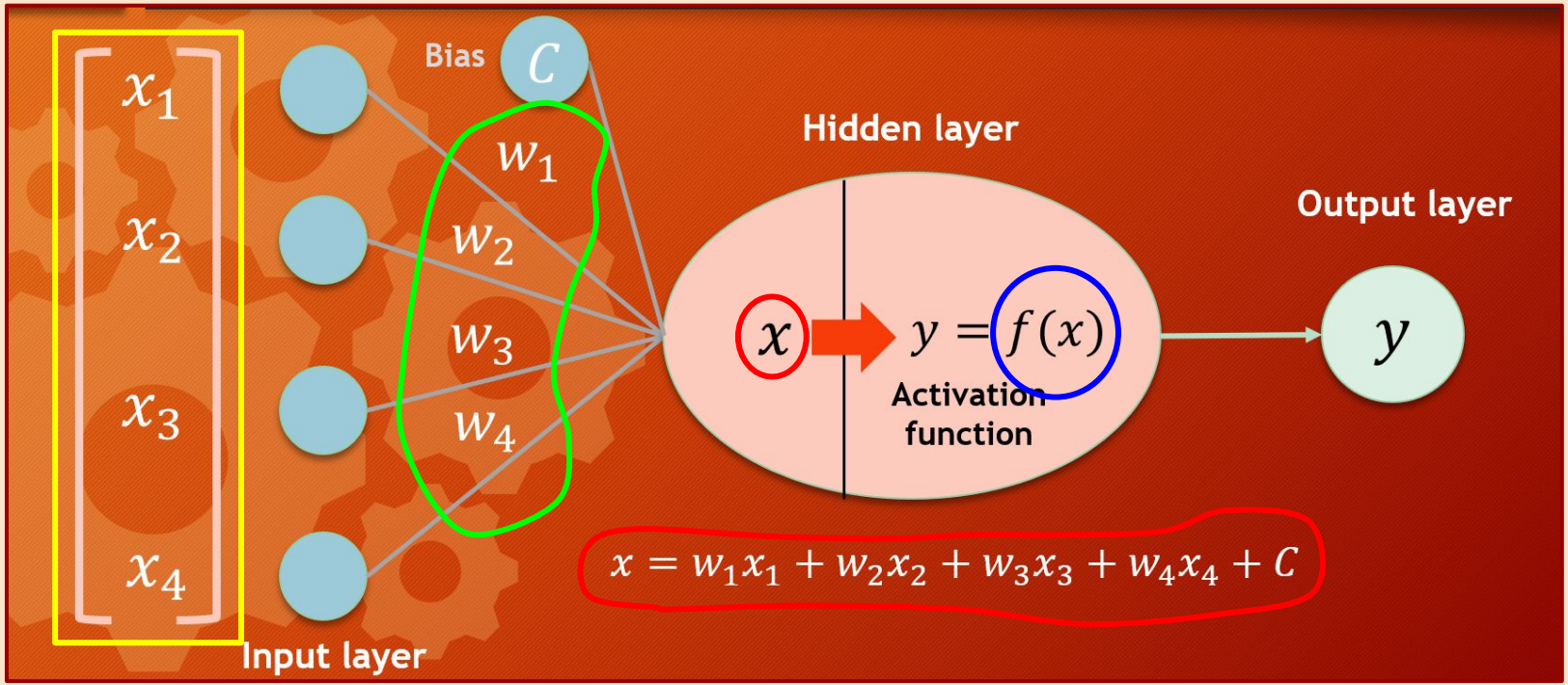


Feature

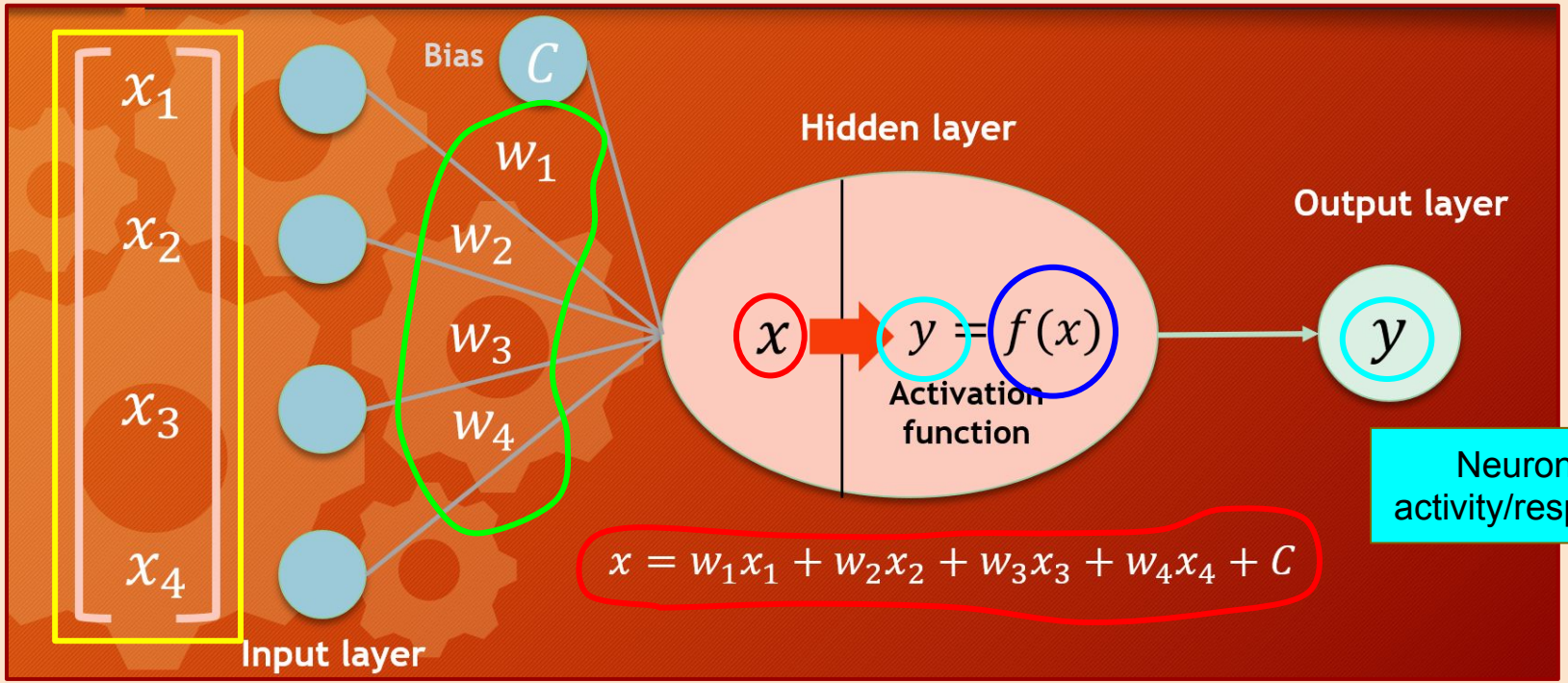
Weights (parameters)

Sum of weights * features

An artificial neuron



An artificial neuron

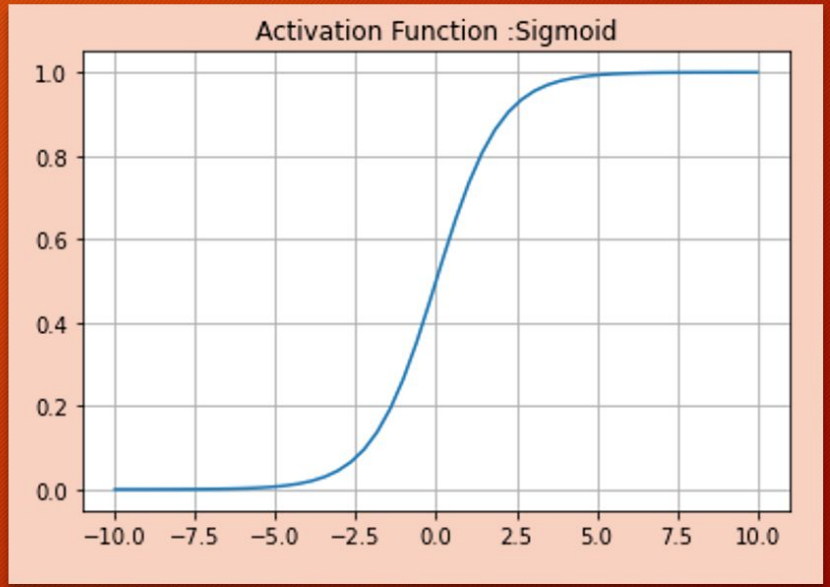


Activation function

- **Sigmoid function**

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

- $Domain: x \in \mathbb{R}$ (Real Number)
- $Range: S(x) \in (0,1)$
- $x \to -\infty$ then $S(x) \to 0$
- $x \to \infty$ then $S(x) \to 1$

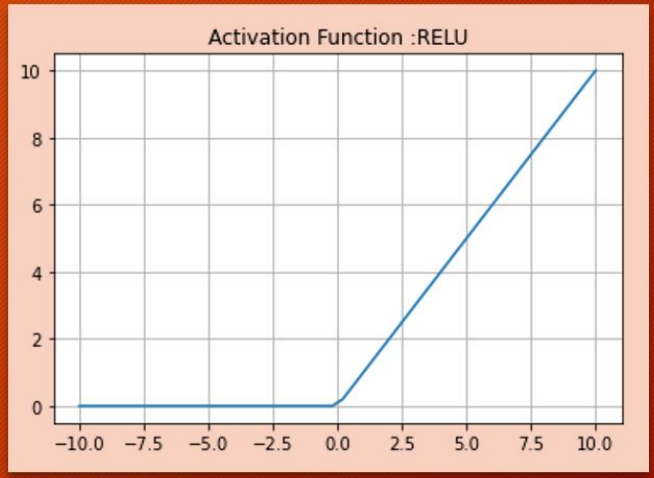


Activation function

- **ReLu function**

$$ReLU(x) = \begin{cases} 0 \ if \ x < 0 \\ x \ if \ x \geq 0 \end{cases}$$

$$ReLU(x) = max(0, x)$$



- $Domain: x \in \mathbb{R}$ (Real Number)
- $Range: S(x) \in (0, \infty)$
- $x \to -\infty$ then $ReLU(x) \to 0$
- $x \to \infty$ then $ReLU(x) \to \infty$

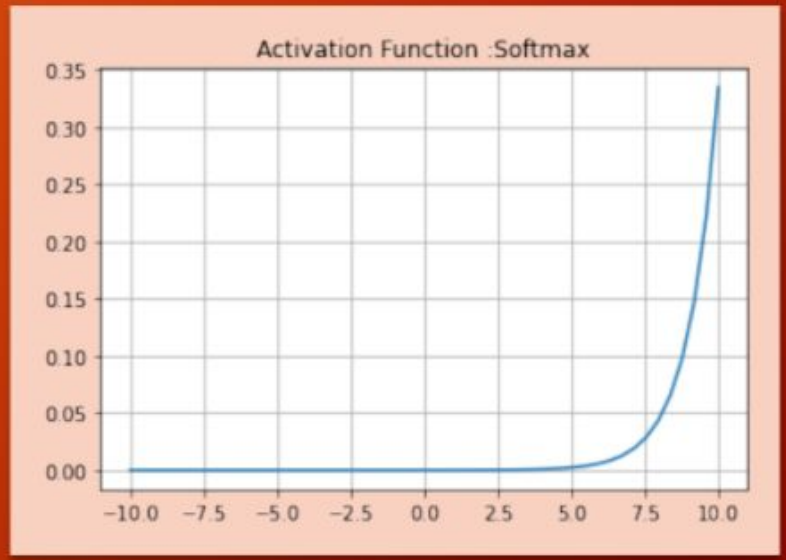
Activation function

- **Softmax function**

$$Softmax(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}}$$

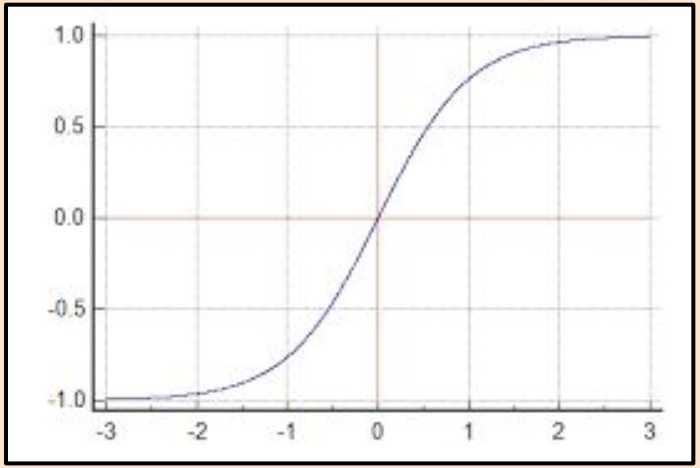
where $j = 1,2, \ldots, n$
and n is the number of nodes in output layer

- $Domain: z_j \in \mathbb{R}$ (Real Number)
- $Range: Softmax(z_j) \in (0,1)$
- $z \to -\infty$ then $S(z) \to 0$
- $z \to \infty$ then $S(z) \to 1$

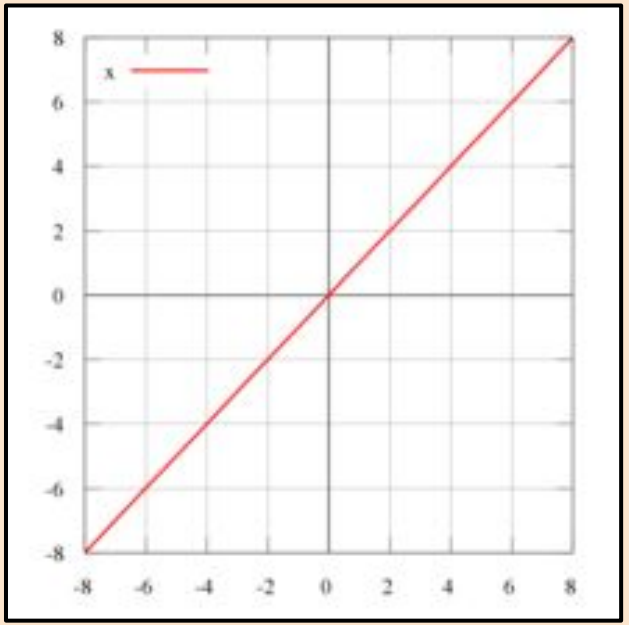


Activation function

tanh

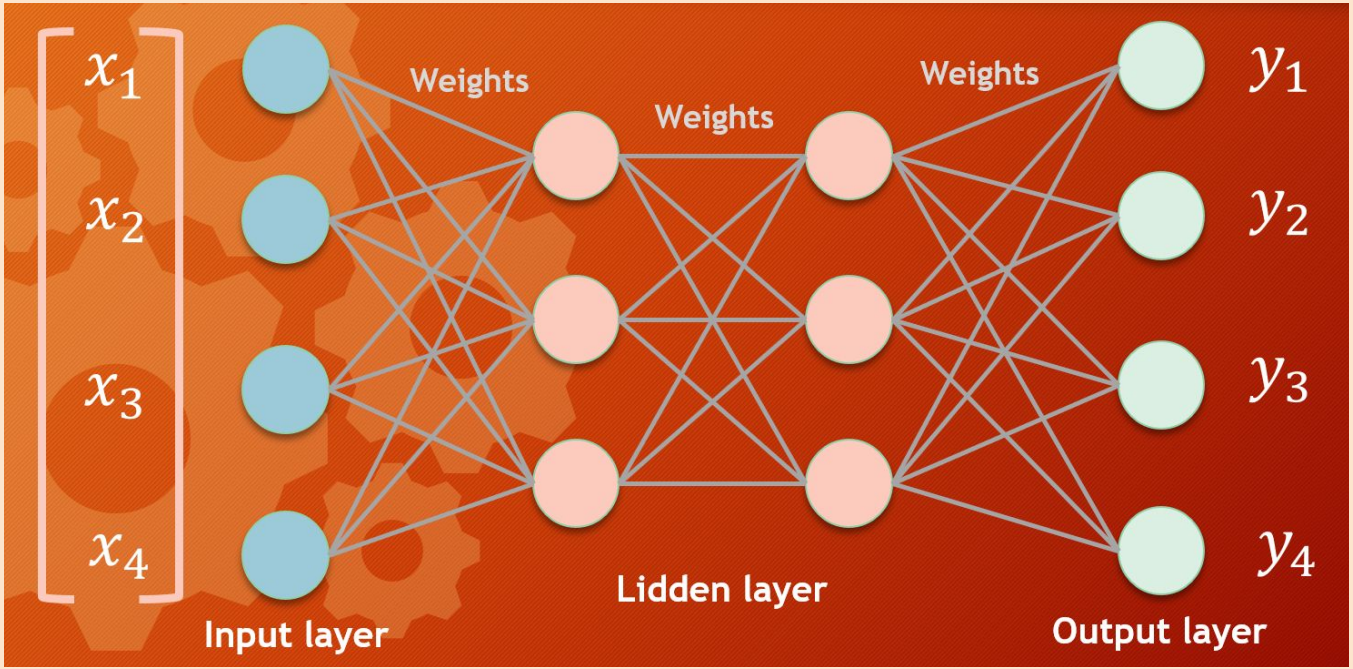


identity



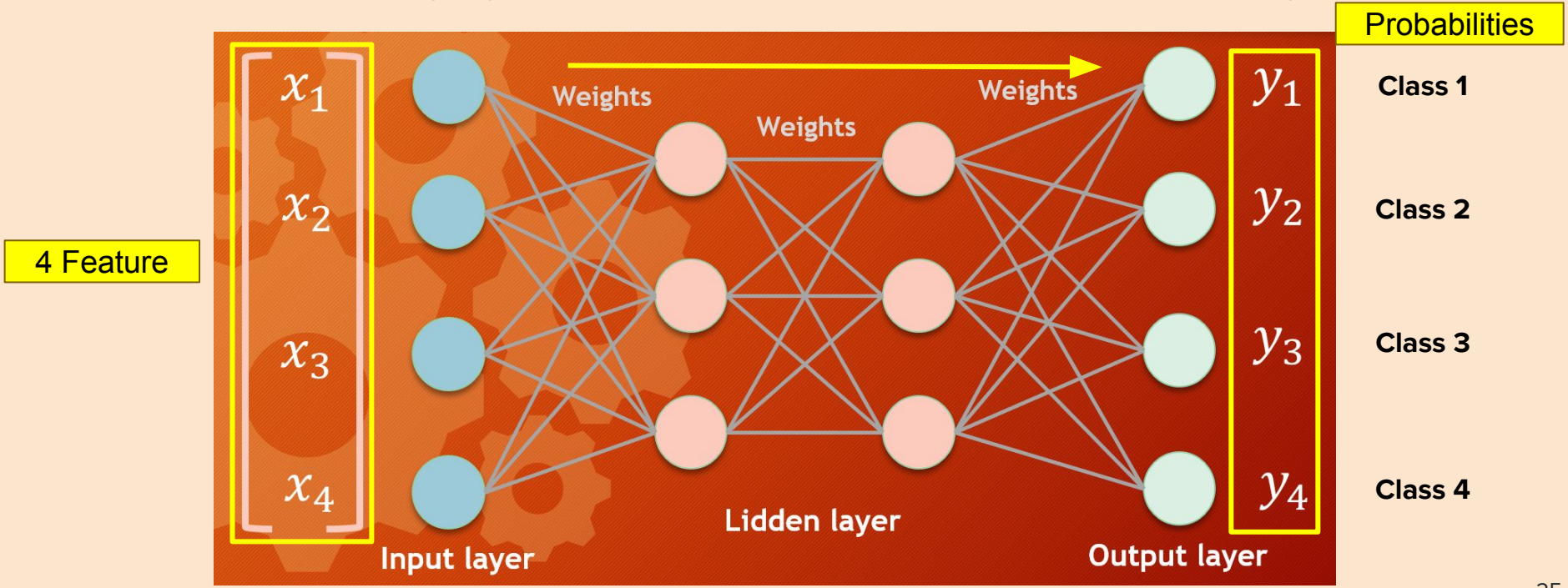
Neural network

A network of single neuron units. Engineers must define **network architecture** they want to use. How many layers the network will have and how units each layer will have



Neural network

A network of single neuron units. Engineers must define **network architecture** they want to use. How many layers the network will have and how units each layer will have

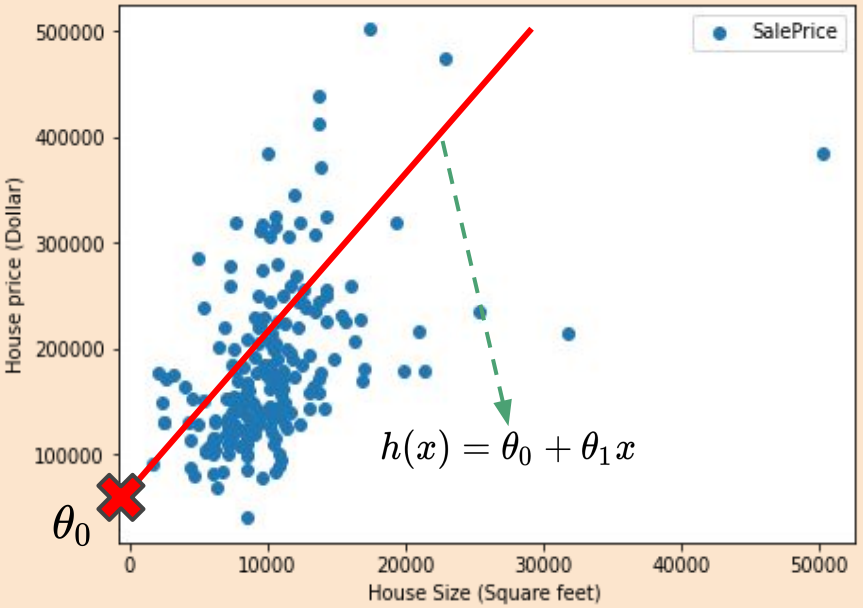


Bias units

Bias units allows us to add any constant to the computation of each layer.

In regression, this is similar to the term θ_0 .

This provides a baseline for activity of neurons in each layer



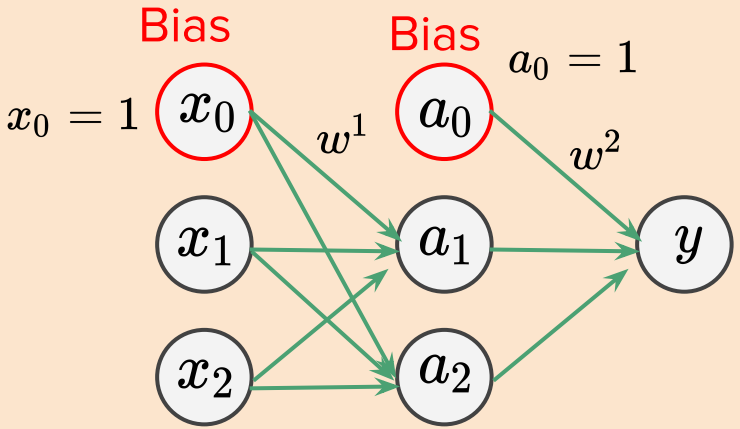
Linear regression

Bias units

Bias units allows us to add any constant to the computation of each layer.

In regression, this is similar to the term θ_0 .

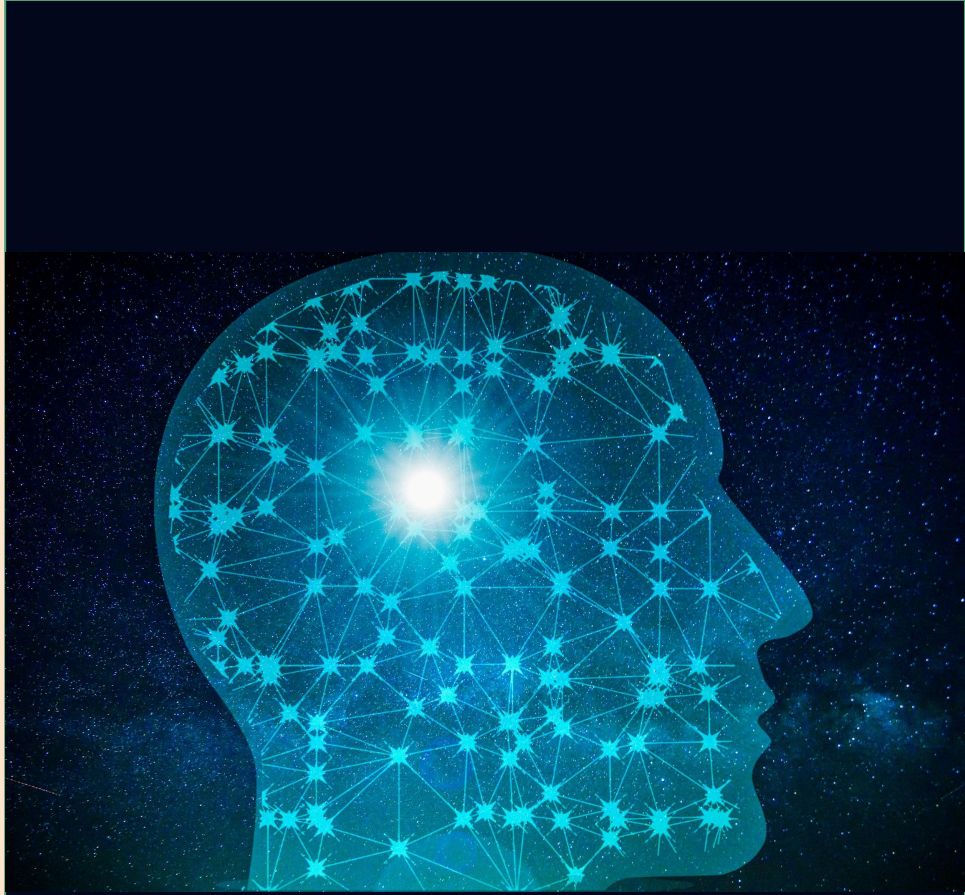
This provides a baseline for activity of neurons in each layer



$$a_1 = f(w_{01}^1 x_0 + w_{11}^1 x_1 + w_{21}^1 x_2)$$
$$a_2 = f(w_{02}^1 x_0 + w_{12}^1 x_1 + w_{22}^1 x_2)$$
$$y = f(w_{01}^2 a_0 + w_{11}^2 a_1 + w_{21}^2 a_2)$$

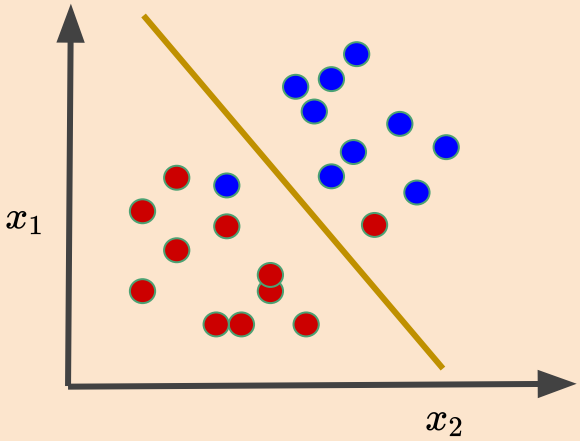
Neural network

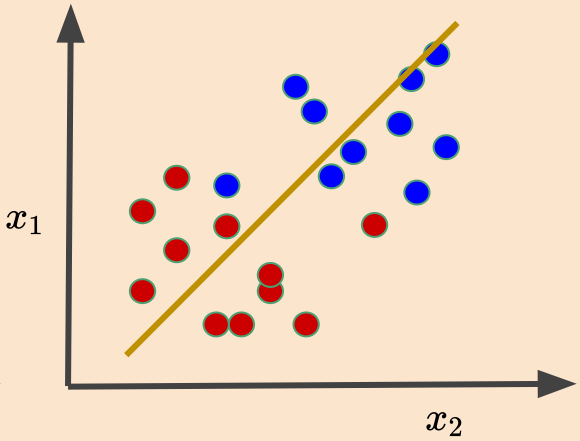
Neural network decision boundary

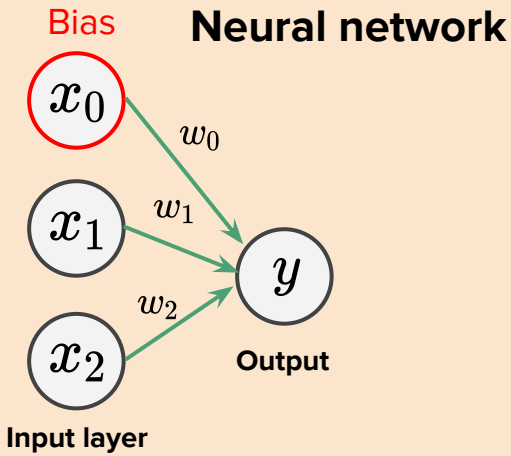


Linear decision boundary

2-layer neural network fits a linear decision boundary. It is called “Linear Classifier”.
For high-dimension, you might think of it as a decision hyperplane.

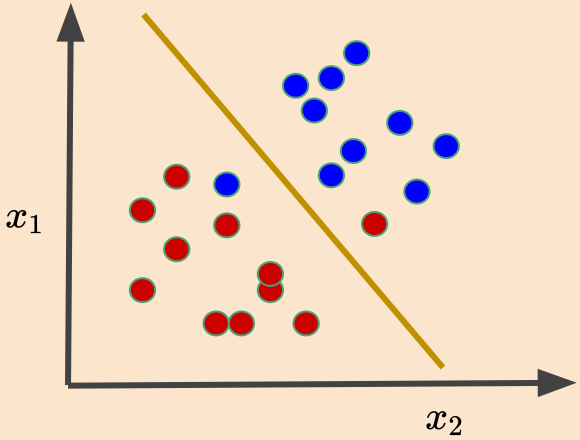


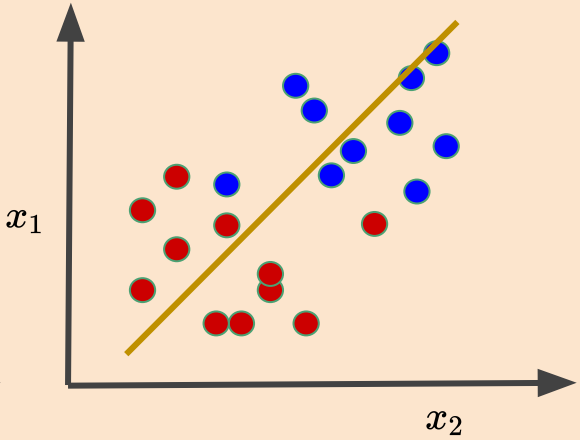


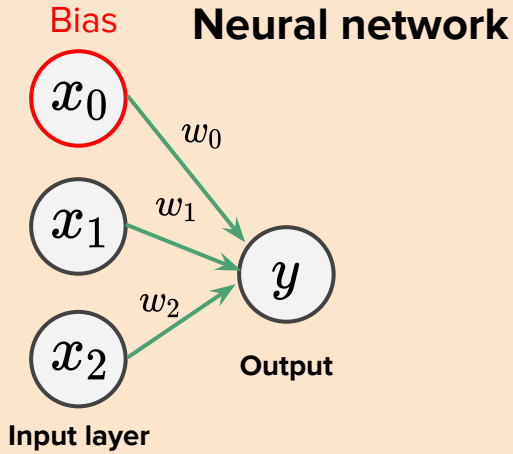


Linear decision boundary

2-layer neural network fits a linear decision boundary. It is called “Linear Classifier”.
For high-dimension, you might think of it as a decision hyperplane.



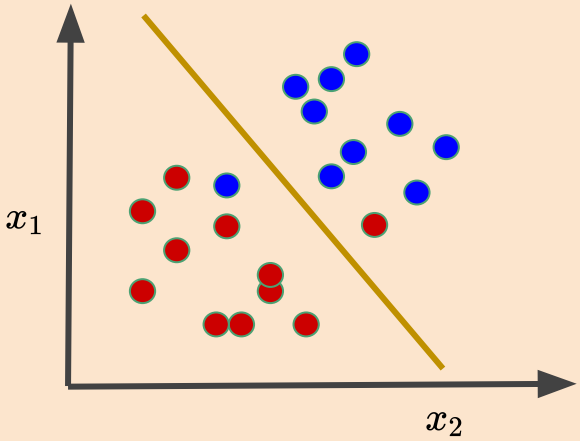




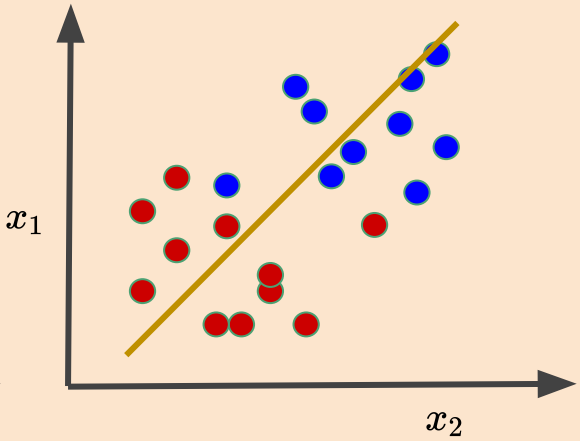
$$y = f(x_0 w_0 + x_1 w_1 + x_2 w_2)$$
$$= f(w_0 + x_1 w_1 + x_2 w_2)$$
$$f : Sigmoid\ function$$

Linear decision boundary

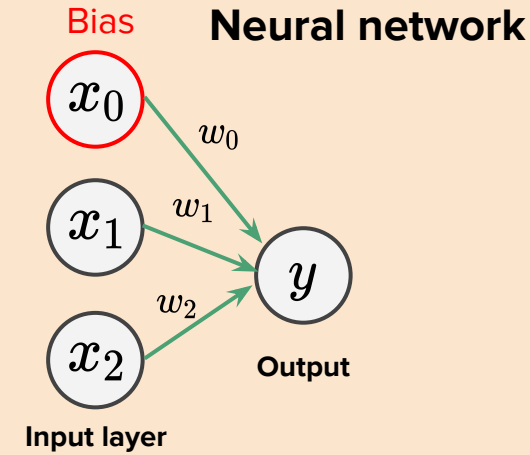
2-layer neural network fits a linear decision boundary. It is called “Linear Classifier”.
For high-dimension, you might think of it as a decision hyperplane.



$w_1 = 1 \quad w_2 = -1$



$w_1 = 0 \quad w_2 = 1$



$$y = f(x_0 w_0 + x_1 w_1 + x_2 w_2)$$
$$= f(w_0 + x_1 w_1 + x_2 w_2)$$
$$f : Sigmoid\ function$$

Logistic regression

Live demo

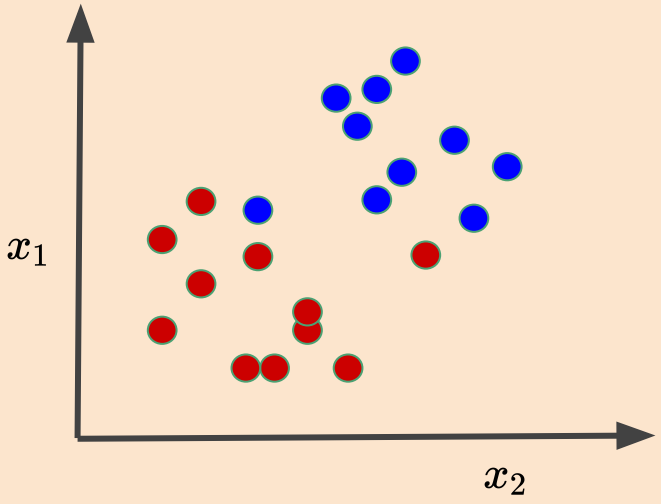
2-layer perceptron cannot fit nonlinear decision boundary.

Live Demo: A Neural Network Playground

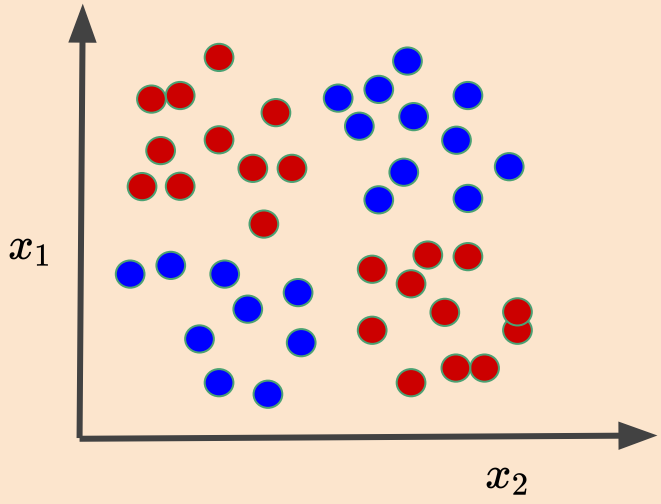
Exercise

- Try picking nonlinear model
- Try adding an extra with 2 neurons in the hidden layer
- Make hidden layer with 5 neurons
- Make two hidden layers

Linear decision boundary

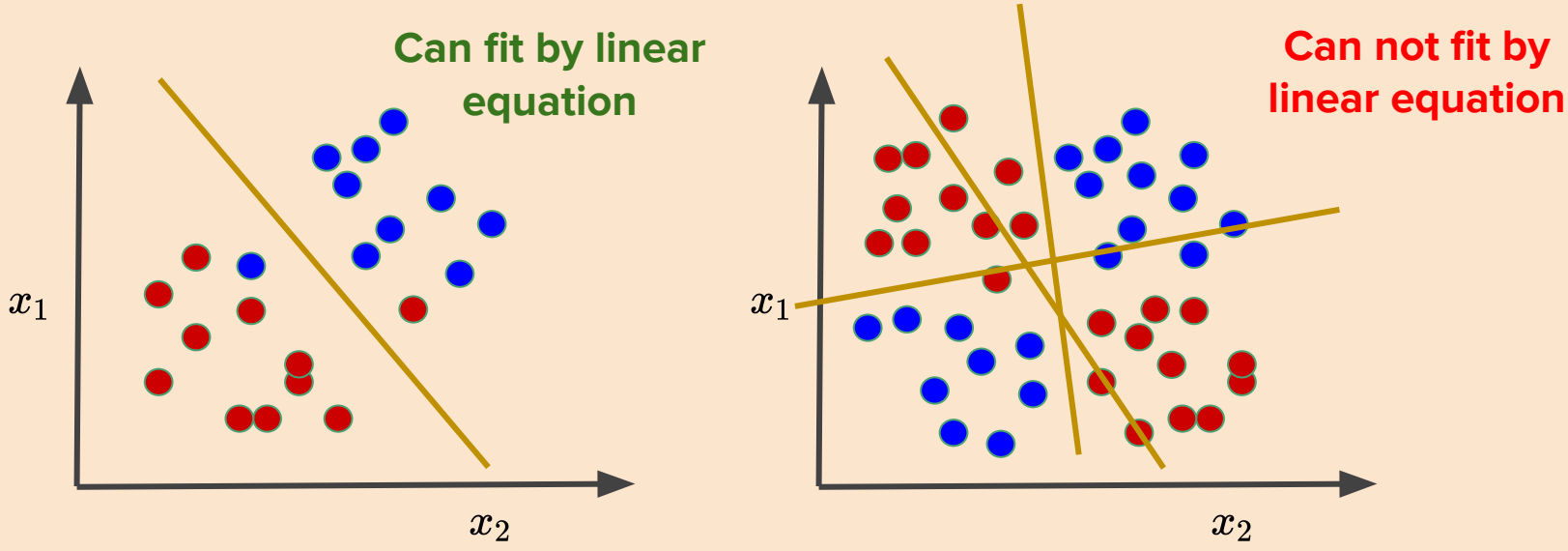


Linear Separable

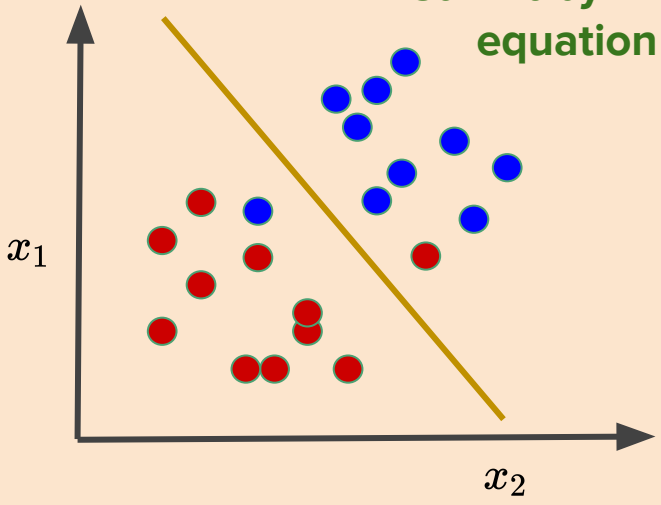


Linear Non- Separable

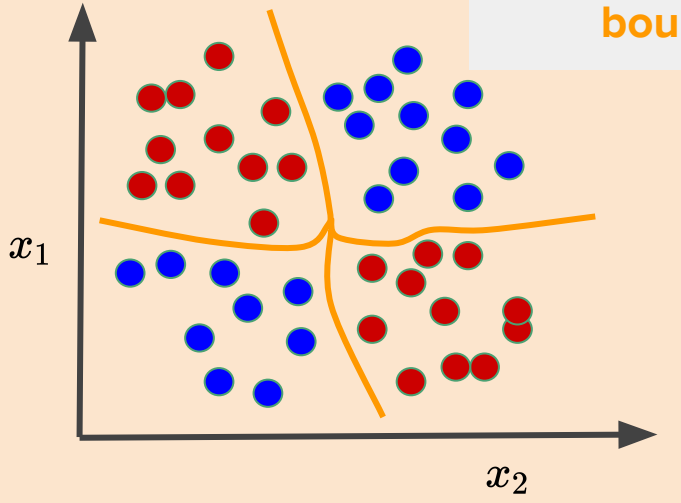
Linear decision boundary



Linear decision boundary

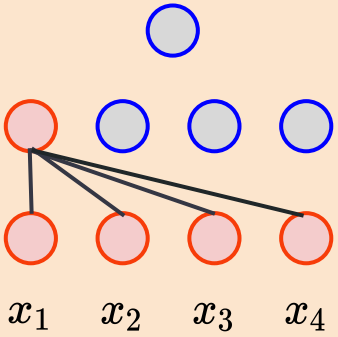


Linear Separable

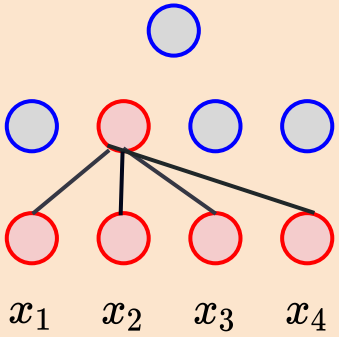


Linear Non- Separable

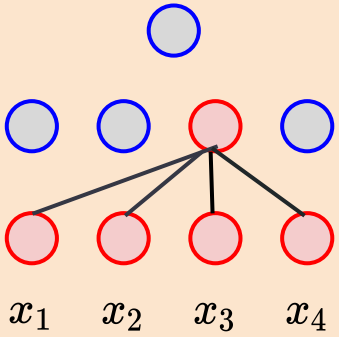
Multilayer perceptron



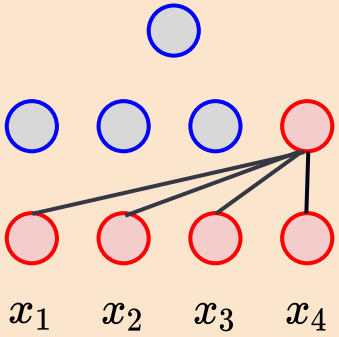
Classification 1



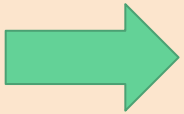
Classification 2

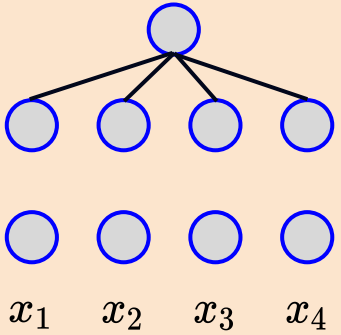


Classification 3



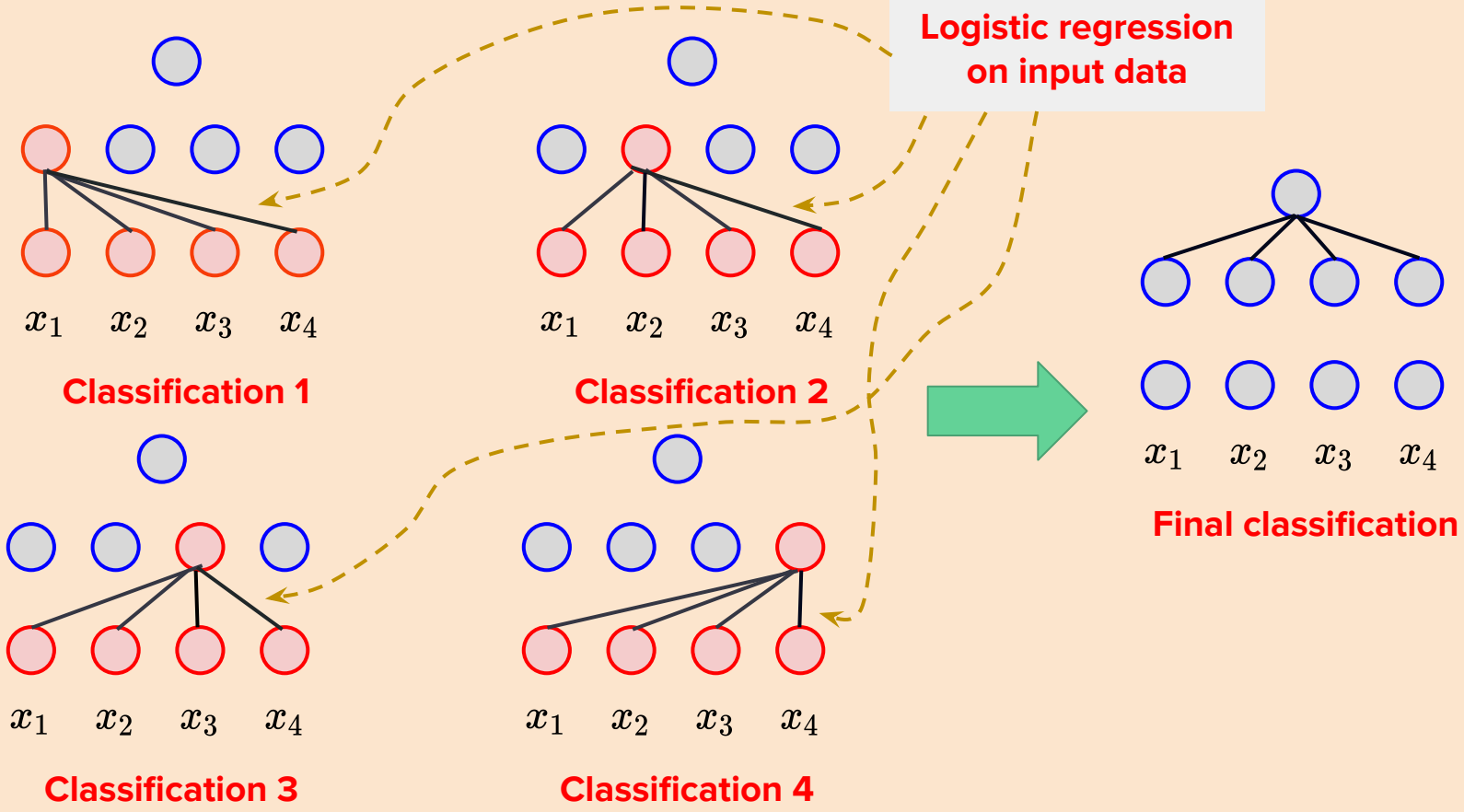
Classification 4



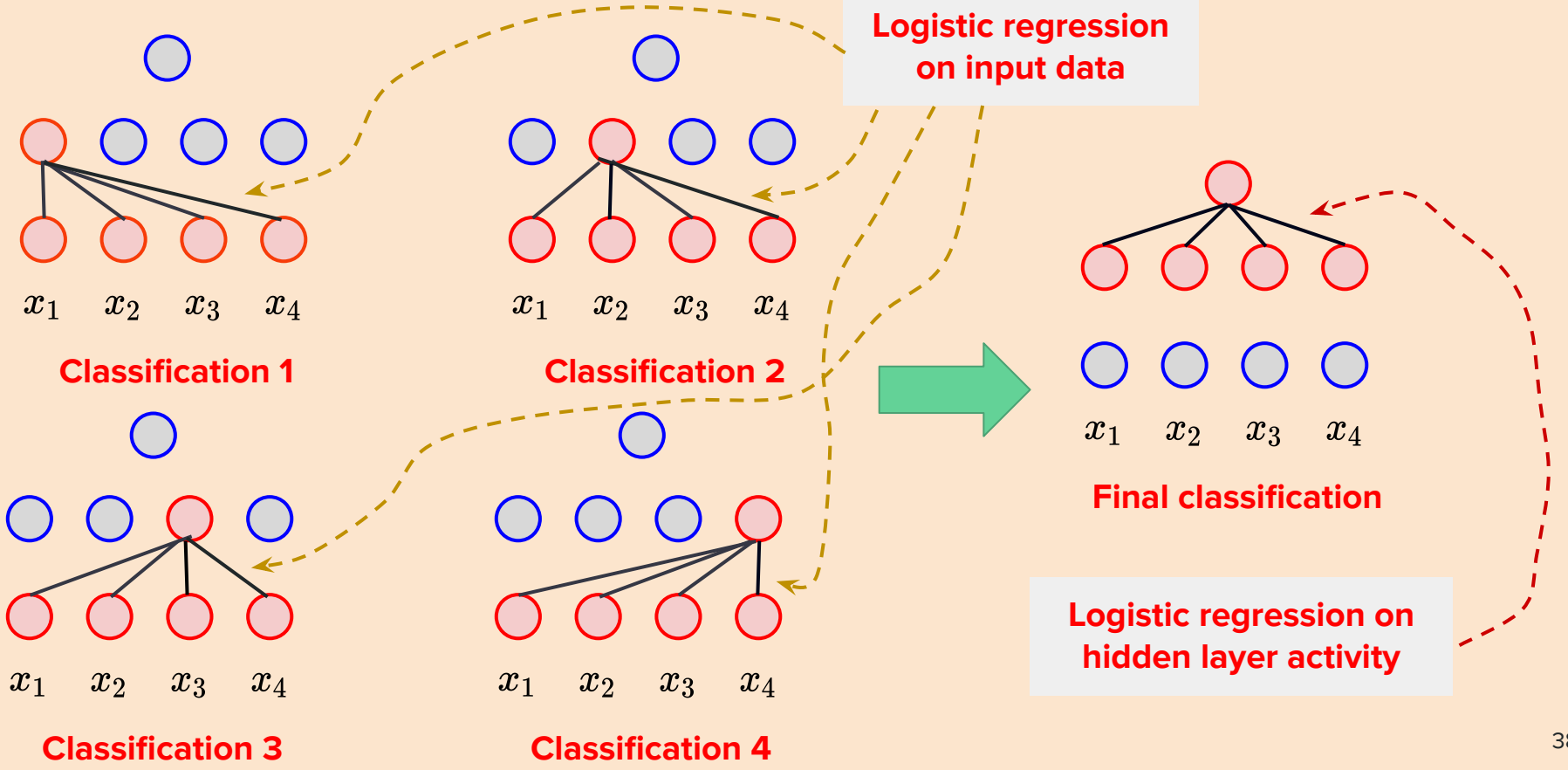


Final classification

Multilayer perceptron

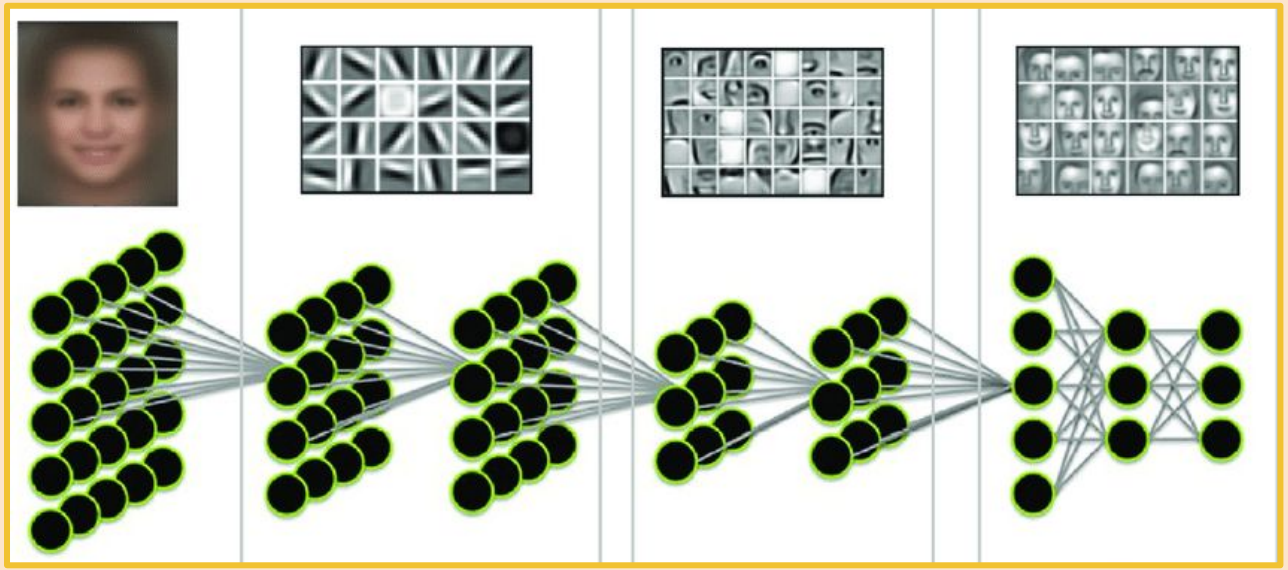


Multilayer perceptron



Network of neurons can understand anything

Facial Recognition



Input
Let's start to feed the face images into the neural networks.

Layer 1-2
The computer learns to identify edge and simple shapes.

Layer 3-4
The computer learns to identify more complex shapes and objects.

Layer 5-7
The computer learns which shapes and objects can be used to define a human face.

Good luck 😉