Introduction to Artificial Neural Networks

Sudong Lee

sudonglee@ulsan.ac.kr

https://dais.ulsan.ac.kr/

Goal

"Review the history of artificial neural networks and

understand how a multilayer perceptron works."

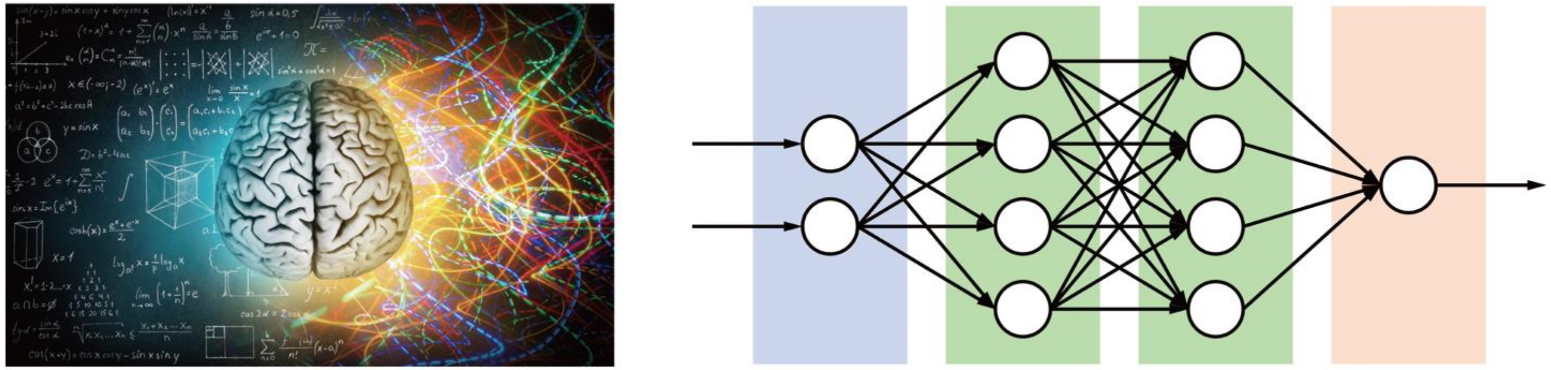
Contents

- History of artificial neural networks
- Multilayer perceptron

History of Artificial Neural Networks

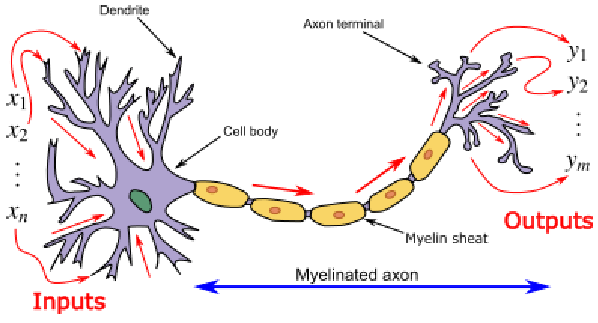
What is an Artificial Neural Network (ANN)?

"An ANN is a collection of connected artificial neurons, which model the neurons in a biological brain."



A biological brain and an artificial neural network

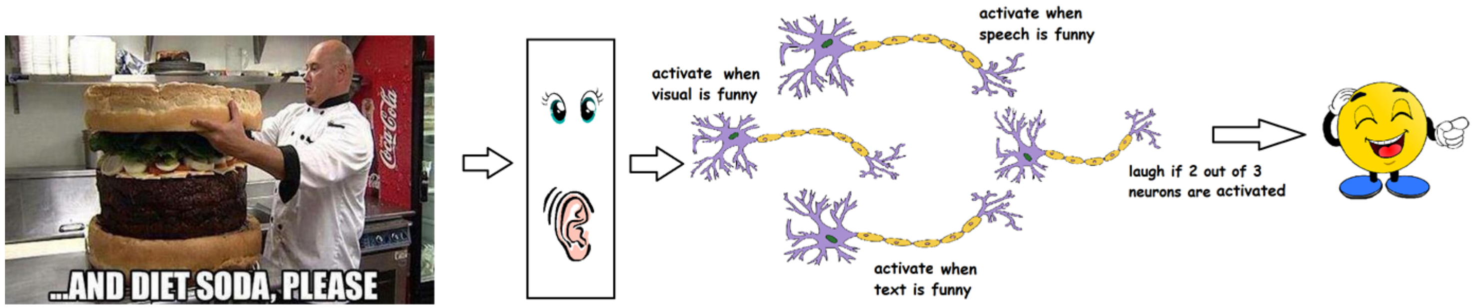
“How do human brains work?”



A biological neuron of a human brain

Neurons in the brain ***aggregate*** and ***activate*** signals.

- Aggregation: perceive the inputs from the previous neurons
- Activation: propagate the output to the next neurons if activated.



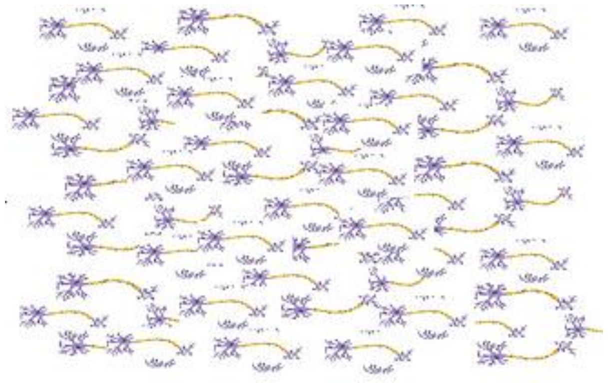
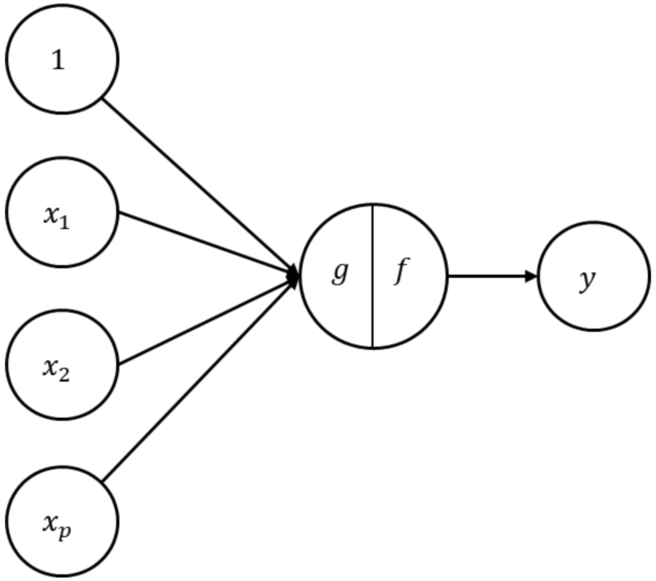


Illustration of the human brain’s division of work

(Source | towardsdatascience)

McCulloch-Pitts Neuron

"The first computational model of a neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943."



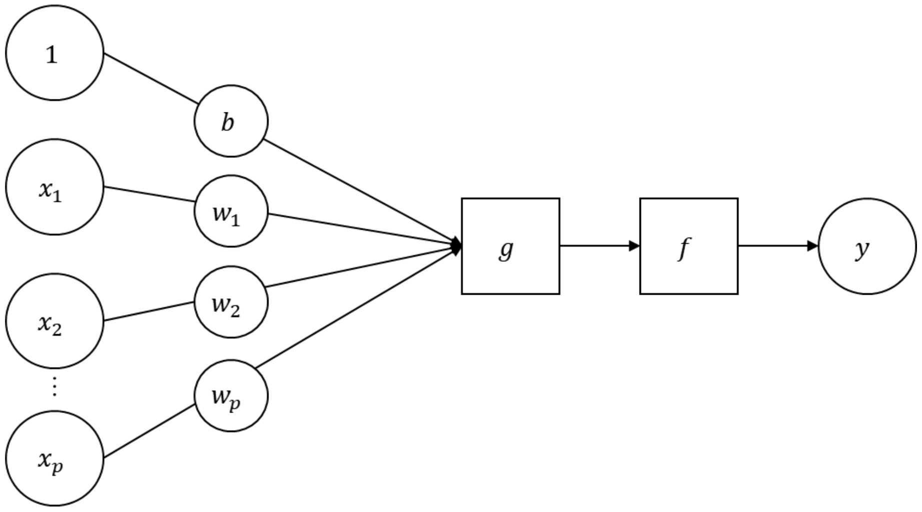
McCulloch-Pitts neuron

- $g(\mathbf{x}) = g(x_1, x_2, \dots, x_p) = \sum_{j=1}^{p} x_j$ where $x_j \in \{0, 1\}\ \forall j$
- $f(x) = \begin{cases} 1 & if\ \ x \geq \theta \\ 0 & if\ \ x < \theta \end{cases}$

$\Rightarrow\ y = f(g(x))$

Perceptron

"In 1958, Frank Rosenblatt proposed the *perceptron* model for binary classification."



Rosenblatt's perceptron

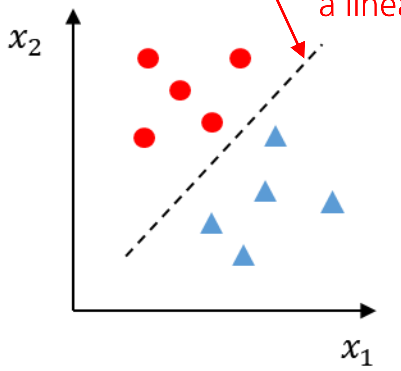
Affine transformation

- $g(\mathbf{x}) = g(x_1, x_2, \dots, x_p) = \boxed{\sum_{j=1}^{p} w_j x_j + b}$, $\mathbf{x} \in \mathbb{R}^p$

- $f(x) = \begin{cases} 1 & if \ \ x \geq \theta \\ 0 & if \ \ x < \theta \end{cases}$

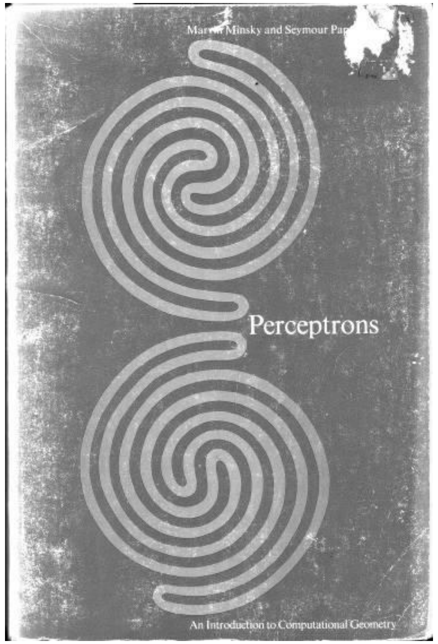
$\Rightarrow \ y = f(g(x))$

A *decision boundary* with a linear hyperplane



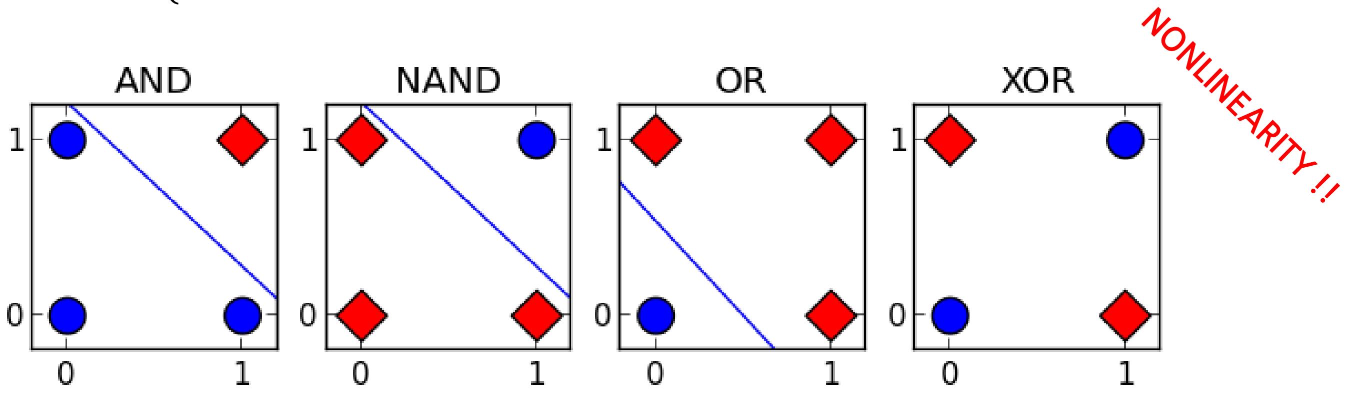
Limitation of the Perceptron Model

"Minsky and Papert criticized that the perceptron model can't solve the XOR problem."



Cover of "Perceptrons"
by Minsky and Papert (1969)

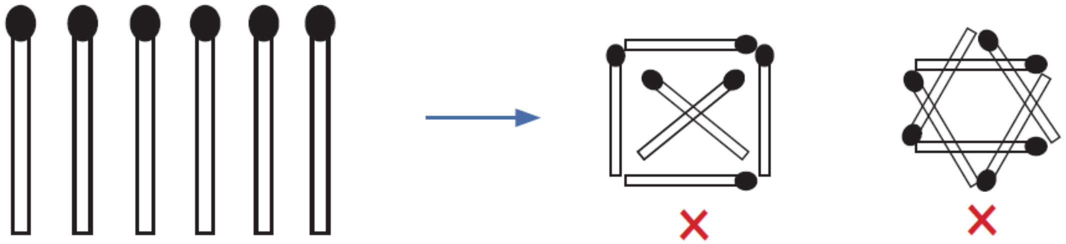
$$y = \begin{cases} 0 & if\ b + w_1x_1 + w_2x_2 \leq 0 \\ 1 & if\ b + w_1x_1 + w_2x_2 > 0 \end{cases} \qquad (b, w_1, w_2) = ?$$

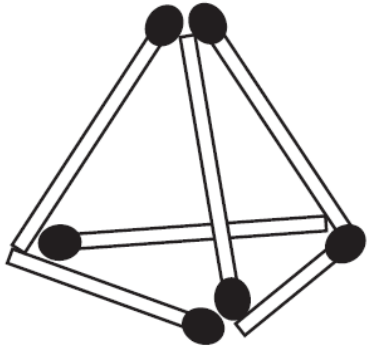


$(-0.7, 0.5, 0.5)$ $(0.7, -0.5, -0.5)$ $(-0.2, 0.5, 0.5)$?

Application of Hidden Layers

"Make 4 equilateral triangles with 6 matchsticks."



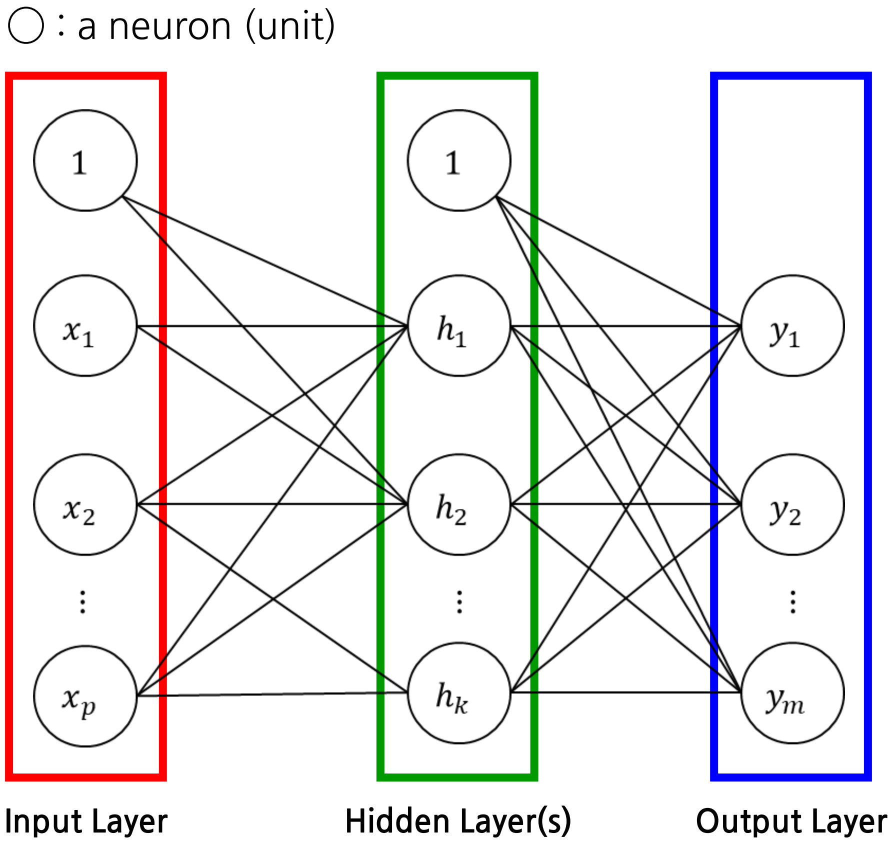


!!

Multilayer Perceptron

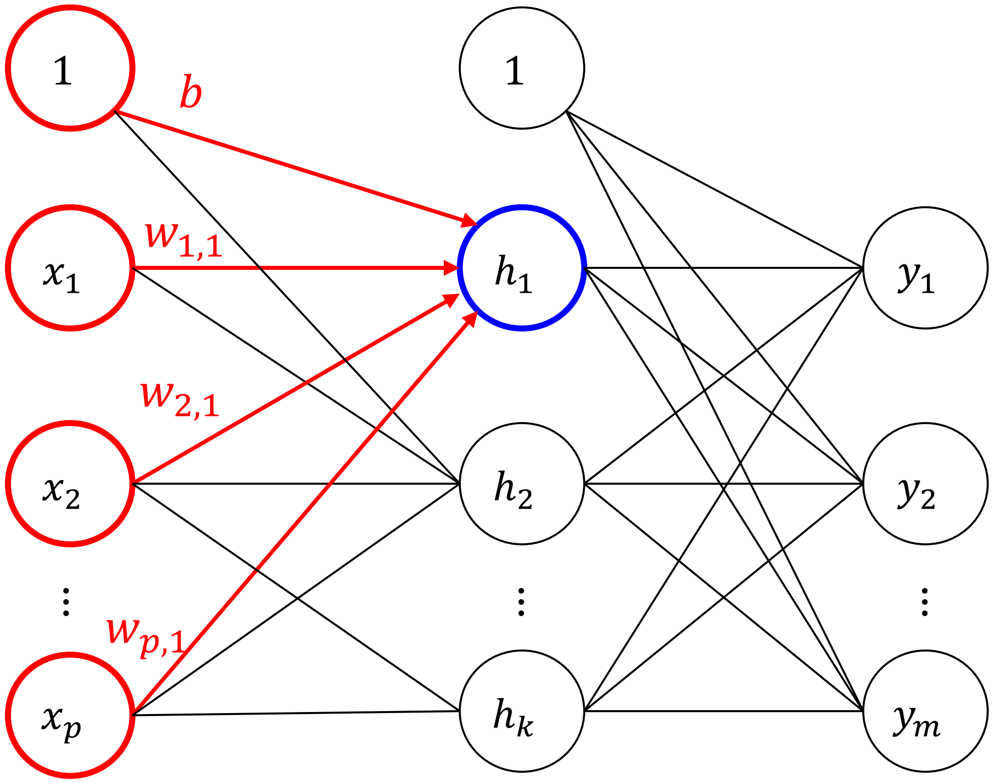
Multilayer Perceptron (MLP)

"An MLP is a feedforward network with at least three layers of neurons: an input, hidden and output layer."



Multilayer Perceptron (MLP)

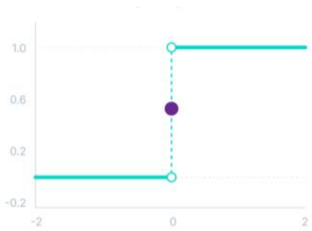
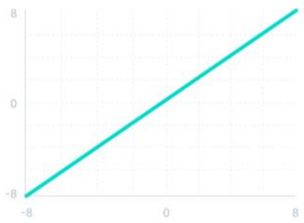
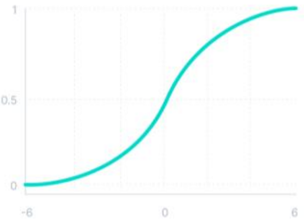
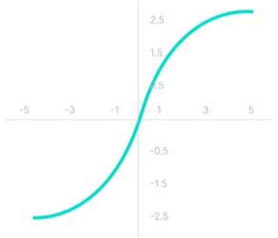
"An MLP consists of fully connected neurons with activation functions."



$h = f(g(\mathbf{x}))$

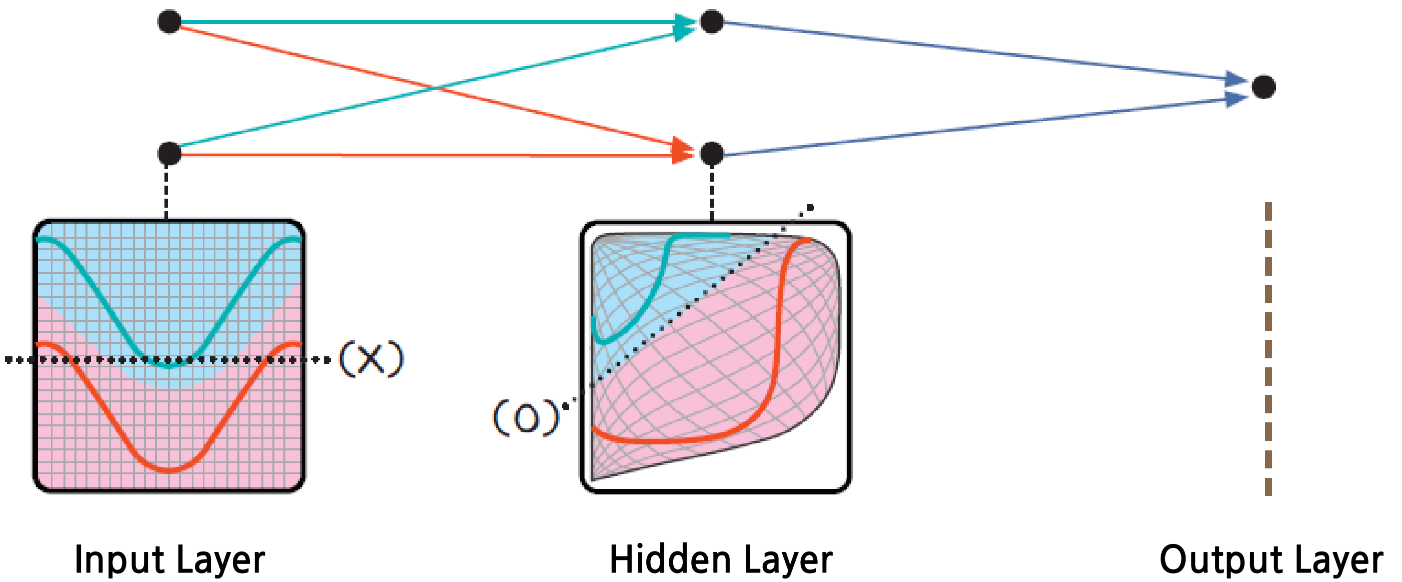
- $g(\mathbf{x}) = g(x_1, x_2, \dots, x_p) = \sum_{j=1}^{p} w_j x_j + b$, $\mathbf{x} \in \mathbb{R}^p$
- $f(x)$: an activation function

Different Types of Activation Functions

	Binary Step Function	Linear	Sigmoid	Hyperbolic Tangent (Tanh)
				
Formula	$f(x) = \begin{cases} 1 & if\ \ x \geq \theta \\ 0 & if\ \ x < \theta \end{cases}$	$f(x) = x$	$f(x) = \dfrac{1}{1 + e^{-x}}$	$f(x) = \tanh(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
Property	• Binary activation with a threshold • No multi-value outputs • The gradient is zero.	• All layers will collapse into one layer. (*i.e.*, the linear of linear is linear.) • The gradient is constant.	• S-shape • The output range of 0 and 1 • The gradient is smooth. 	• S-shape. • Zero-centered output • The gradient is smooth. 

Nonlinear Decision Boundary by MLP

"An MLP can represent a nonlinear decision boundary using a hidden layer(s) with a nonlinear activation function(s)."



Universal Approximation Theorem

- *Theorem*. Universal Approximation Theorem (Hornik, Stinchcombe, & White, 1989)

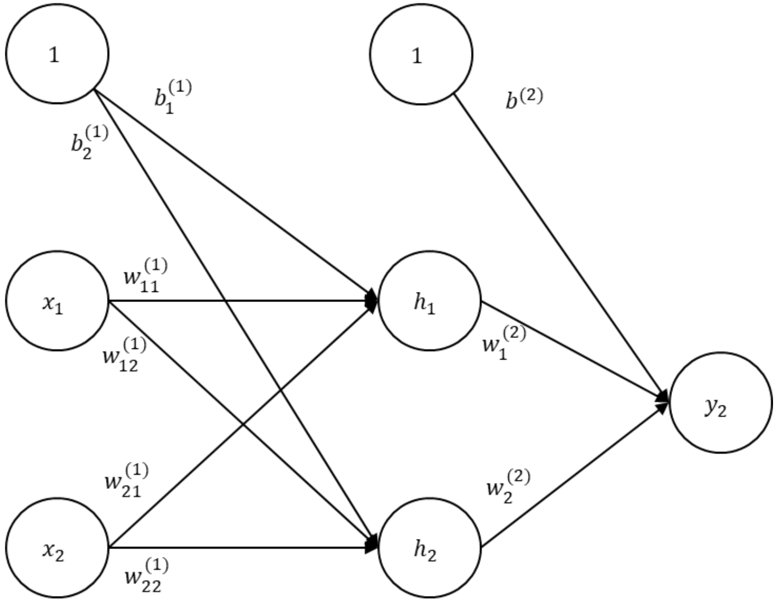
  A feedforward neural network with a single hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets of $\mathbb{R}^n$*, given appropriate non-linear activation functions.*

  $\Rightarrow$ An MLP can represent an arbitrary nonlinear decision boundary!

- **"However, there are some limitations."**
  - Existence theorem
    - They state that such a neural network exists and do not provide any way to find it.
    - They also do not guarantee that any method, such as backpropagation, might find such a neural network.
  - Limit theorem
    - There is no guarantee that any finite size, say, 10,000 neurons, is enough.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." *Neural networks* 2.5 (1989): 359-366.

MLP for XOR Problem



x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Input layer → Hidden layer

$$h_1 = \sigma(w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + b_1^{(1)})$$

$$h_2 = \sigma(w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + b_2^{(1)})$$

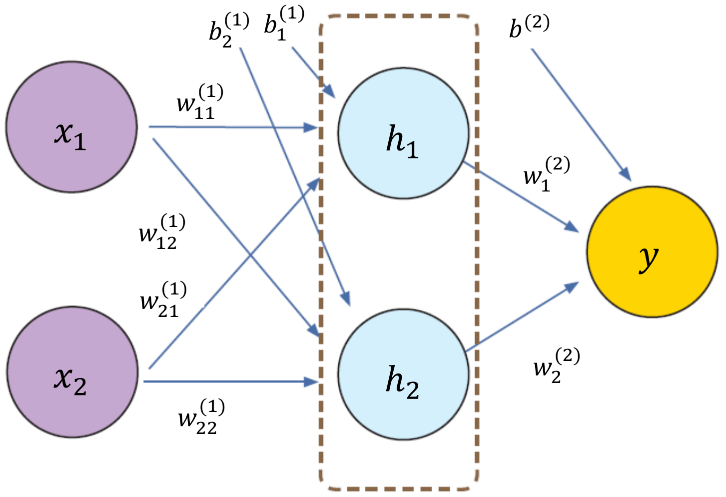
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Hidden layer → Output layer

$$y = \sigma(w_1^{(2)} h_1 + w_2^{(2)} h_2 + b^{(2)})$$

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

MLP for XOR Problem



$$\boldsymbol{X}^T = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \\ x_{4,1} & x_{4,2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\boldsymbol{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 3 & 5 \end{bmatrix}, \boldsymbol{W}^{(2)} = \begin{bmatrix} w_1^{(2)} \\ w_2^{(2)} \end{bmatrix} = \begin{bmatrix} -7 \\ 5 \end{bmatrix}$$

$$\boldsymbol{B}^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix} = \begin{bmatrix} -5 \\ -2 \end{bmatrix}, \boldsymbol{B}^{(2)} = \begin{bmatrix} b^{(2)} \end{bmatrix} = [-3]$$

x_1	x_2	h_1	h_2	y	XOR
0	0	0.0067	0.1192	0.0794	0
0	1	0.1192	0.9526	0.7168	1
1	0	0.1192	0.9526	0.7168	1
1	1	0.7311	0.9997	0.0423	0

Takeaways

Takeaways

1. **The history of artificial neural networks**
    - McCulloch-Pitts Neuron: binary inputs, summation, binary output
    - Perceptron: linear decision boundary using affine transformation
    - MLP: nonlinear decision boundary using a hidden layer(s) and a nonlinear activation function(s).
2. **How an MLP works**
    - Hidden layers with nonlinear activation functions.
    - Universal approximation theorem.

Thank you! 😀