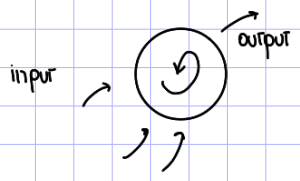
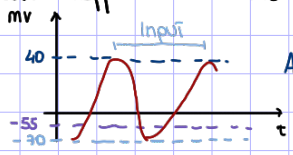
7. NEURAL NETWORKS : DEEP LEARNING → We're going to talk about deep learning based on the artificial neural network, a model inspired by the biological network of the human brain

Terminology –
1. AI : artificial intelligence → simulation of the human intelligence in a machine
2. ML : machine learning → the machine learn something by NOT give it directly the instruction
3. DL : deep learning → the machine learn by using neural networks

Biological model of NEURON



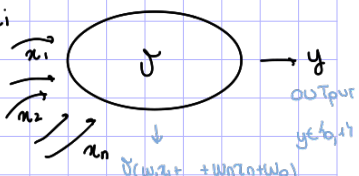
The IDEA is to simulate the neural inside our head
There are many inputs that are called **dendrides** an there is only one ourputs that is the **axon** → High parallel processed
What happen inside the neuron ? The voltage (electrical) increases a lot ↓



Electricity in the neuron — at each input the potential increase
Action potential : that is the activation 3.
This is the threshold potential : the neuron activates 2.
Membrane potential at rest : the neuron is "sleeping" 1.

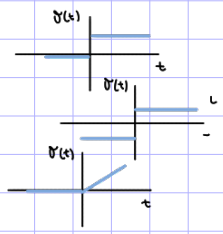
7.1 What is the first model of NN? PERCEPTON → This is the first mathematical surrogate

↳ We have different input x_i that represent the neuron (binari : 0-1)
In the middle there is the γ : activation function that takes a linear combination of x_i
$\gamma(w_1x_1 + \dots + w_nx_n + w_0)$ ← weight of the rest potential
The ourput is a binary one : active (1) or not active (0)
All the weights are collected in an array $w = (w_1, \dots, w_n)$, BUT the w_0 is called bias

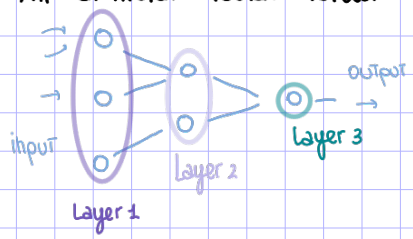


Example of activation function — usually NON-LINEAR

1. Step function — discontinue
$$\gamma(t) = \begin{cases} 0 & t \le 0 \\ 1 & t > 0 \end{cases}$$
2. Sign function
$$\gamma(t) = \begin{cases} -1 & t \le 0 \\ 1 & t > 0 \end{cases}$$
3. ReLU function — continuous
$$\gamma(t) = \max(0, t)$$



An artificial neural network is a connection of different perceptons collected in different layers



7.2 A little bit of history

- 1943 : Description of the neural model → Pitts & McCulloch propose the first and simplest mathematical model of a neuron
- 1949 : Study the «activation» memory → Hebb proposed a mathematical formula to describe the learning process (memory of activation)
- 1957 : Presentation of the perceptron model → Rosenbatt — cannot simulate xor function
- 1969 : Study the limitation of the perceptron → STOP using NN because there are limitation in the mathematical model ↓ Minsky & Papert
- 1986 : Proposal - back propagation theory → STUDY START AGAIN : Rumel Hart
- 1990 : Unsupervised learning → Kohonen
- 1990 - 2000 : Study about the universal approximation theorems → mathematical results

=> Our purpose is to use NN to solve PDE in a faster way

Notation –
- y := ourput, generated by the activation function
- γ := activation function
- x := input
- w := weight and w_0 := bias

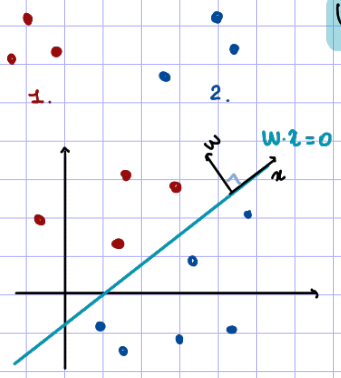
$y = \gamma(\tilde{w}\cdot\tilde{x} + w_0) = \gamma(w \cdot x)$ where $w = (w_0, \tilde{w}) \in \mathbb{R}^{n+1}$, $x = (1, \tilde{x}) \in \mathbb{R}^{n+1}$
↓
$y = \gamma(w_1x_1 + \dots + w_nx_n + w_0)$

How we can use perceptron to CLASSIFY data?

Imagine to have two class and we want to associated to the ≠ element of the class the corresponding layer
(randomly distributed in the space)

PURPOSE: I want that a machine, given an input, tells me if it is blue or red, so the belonging class of the element

We want to performe the line that separate the two classes



$w \cdot x = 0$ hyperplane in higher dimension

When I found w, given the input x I can have as answer the label
$\Rightarrow$ I create the classifier $\longrightarrow$ Let's use $\sigma(t) = \begin{cases} 1 & t>0 \\ 0 & t\leq 0 \end{cases}$

How to build the CLASSIFIER? Hence, I we can find the weights? **TRAINING PHASE**

Consider to have a known-sample $(x^n, y^n)_{n=1,\dots,N}$

The **goal** is to find w

We can build the functional $E(w) = \frac{1}{N}\sum_{n=1}^{N} |y_n - \sigma(w \cdot x^n)|^2 \geq 0$

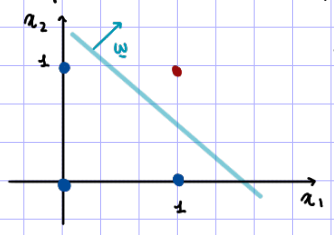
LOSS FUNCTION: measures the mismatch between the data that I know and the one product by my classifier

Hence, we want $w := \underset{w \in \mathbb{R}^{n+1}}{\arg\min}\ E(w)$ so to find the minimization of the LOSS function

this is why NN is difficult: we have to minimize in higher space

ISSUE: no globale minimum in the majority of the situation

Example. We want to approximate with the NN the **AND function** (logical operator)

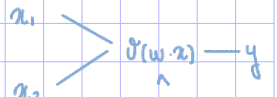


We're looking for a line
But it's evident that we have different solutions

x_1	x_2	$x_1 \wedge x_2$	
0	0	0	y_1
0	1	0	y_2
1	0	0	y_3
1	1	1	y_4

We can build the LOSS function: $E(w) = \frac{1}{N}\sum_{n=1}^{N} |y_n - \sigma(w \cdot x^n)|^2 \longrightarrow N=4,\ w=(w_0, w_1, w_2),\ x=(1, x_1, x_2)$

$$E(w) = \frac{1}{4}\left(|0-\sigma(w_0+0+0)|^2 + |0-\sigma(w_0+0+w_2)|^2 + |0-\sigma(w_0+w_1+0)|^2 + |1-\sigma(w_0+w_1+w_2)|^2\right)$$



We decide as activation function the step function: $\sigma(t) = \begin{cases} 0 & t\leq 0 \\ 1 & t>0 \end{cases}$

Hence, to solve the problem $w := \underset{w \in \mathbb{R}^N}{\arg\min}\ E(w)$ we have to solve the linear system

$$\begin{cases} |0-\sigma(w_0)|^2 = 0 \\ |0-\sigma(w_0+w_2)|^2 = 0 \\ |0-\sigma(w_0+w_1)|^2 = 0 \\ |0-\sigma(w_0+w_1+w_2)|^2 = 0 \end{cases} \Rightarrow \begin{cases} w_0 \leq 0 \\ w_0+w_2 \leq 0 \\ w_0+w_1 \leq 0 \\ w_0+w_1+w_2 > 0 \end{cases}$$

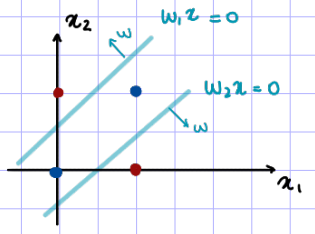
Taking in account $\sigma(t)$

This is the linear system that we have to solve to find the green line
Moreover, there are ≠ solutions, i.e. $w = \left(-\frac{2}{3}, \frac{1}{2}, \frac{1}{2}\right)$ & $w = (-1, 1, 1)$

7.4 Feed forward neural network ⟶ We need at least two perceptrons

Limitation - XOR operator.

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

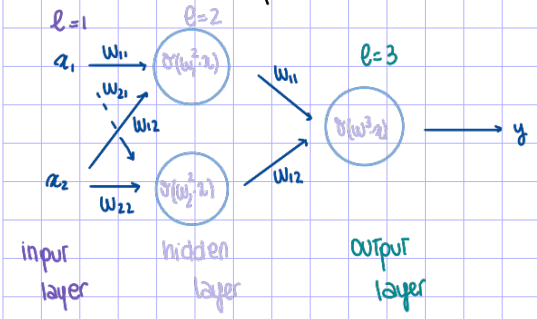


BUT using a line we cannot separate the point that XOR generate ⟶ one perceptron fails

↓

SOLUTION: If we use two line (two perceptron) we can face this limitation

Now we can represent the NN of the XOR



So we have different layer and 3 perceptron to have the solution and we build feed forward NN in which we can use $\neq \sigma$ in different layer ⟶ This is difficult to solve!
BUT in each layer, to solve fastly and parallelite the comparation, we usually use the same activation function

For more details on the generalization of FF-NN see the lecture note's lesson 7