Master in Internet of Things for eHealth

M5. Smart Data Knowledge / Analytics

Classical Neural Networks

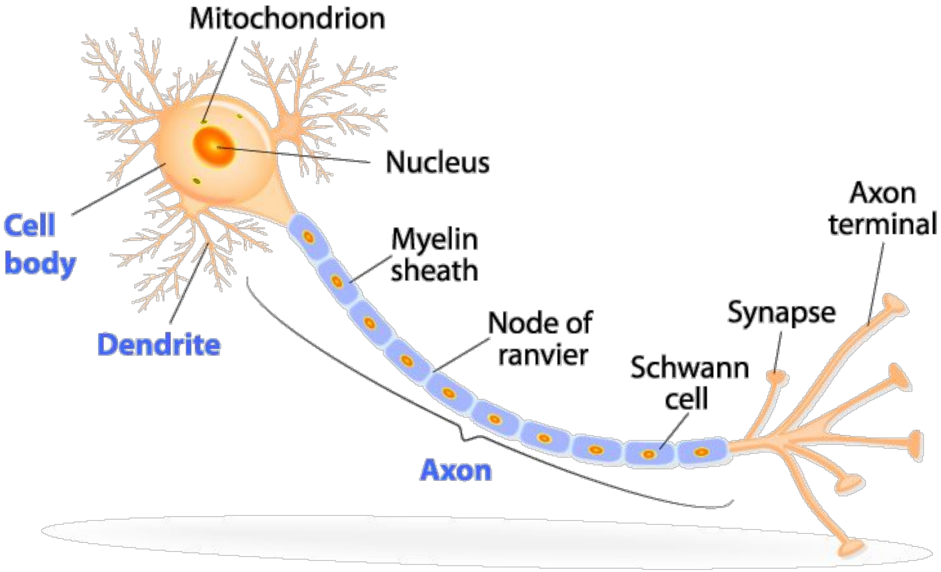
Instructor **David Gerónimo**
research@davidgeronimo.com

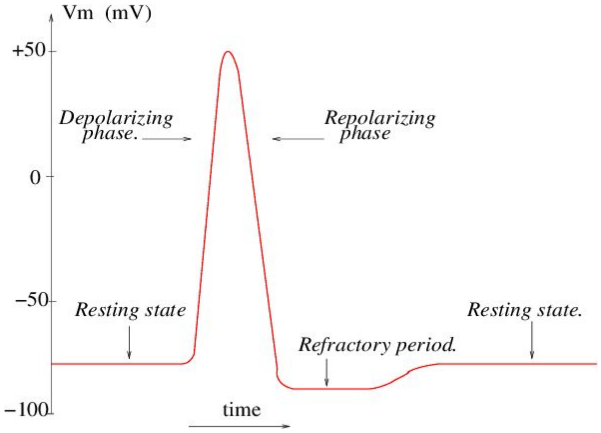
Universitat Autònoma
de Barcelona

January 18th 2019

Biological Neurons

- **Structure of a biological neuron**

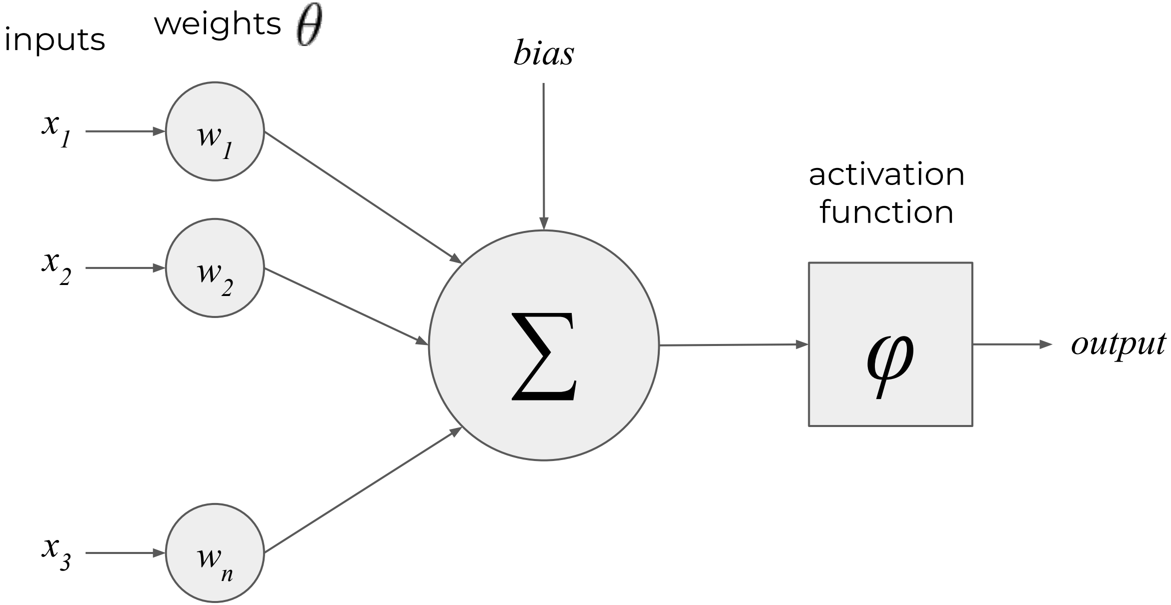




action potential of a neuron

Perceptron

- **Structure of an artificial neuron**

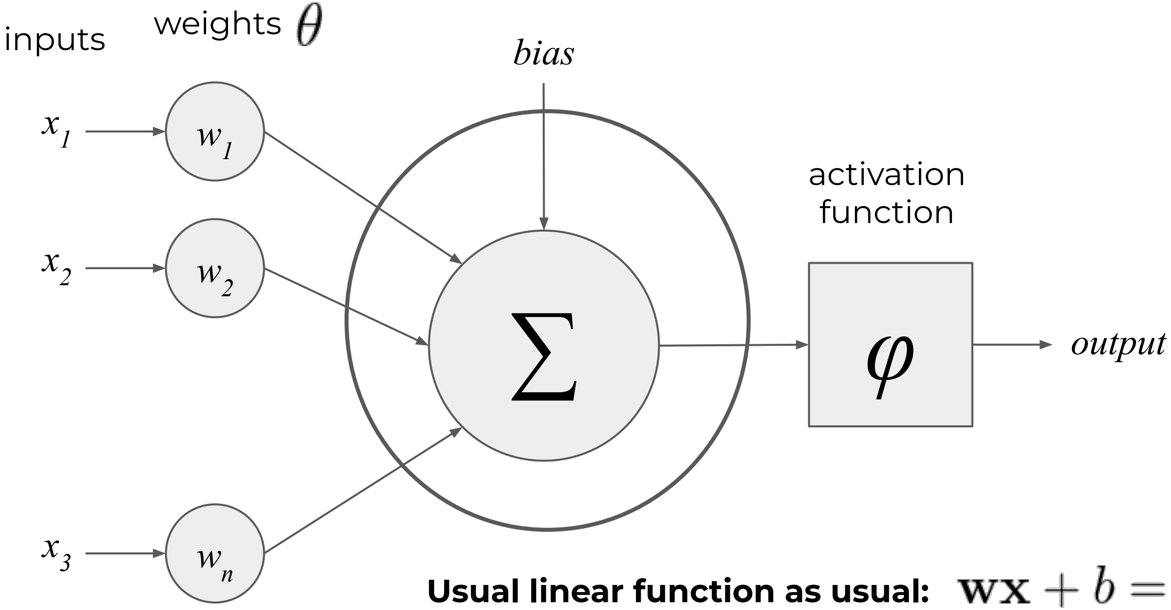


Frank Rosenblatt (1957)

Perceptron

- **Structure of an artificial neuron**

Frank Rosenblatt (1957)

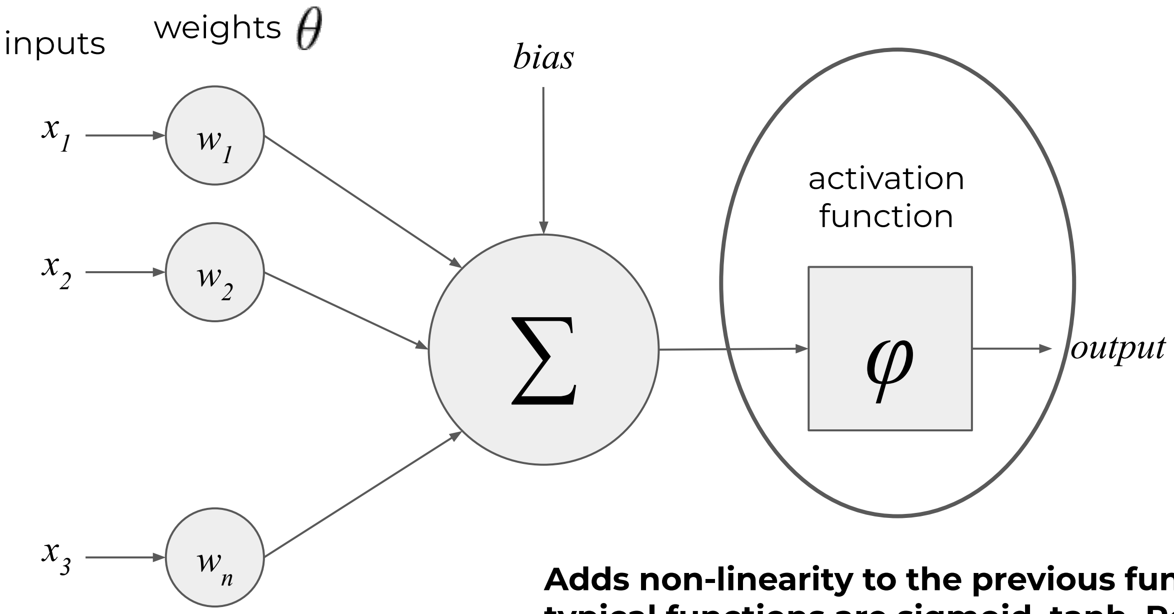


Usual linear function as usual: $\mathbf{wx} + b = \sum_i w_i x_i + b$

Perceptron

- **Structure of an artificial neuron**

Frank Rosenblatt (1957)



inputs

weights θ

bias

x_1 x_2 x_3

w_1 w_2 w_n

Σ

activation function

φ

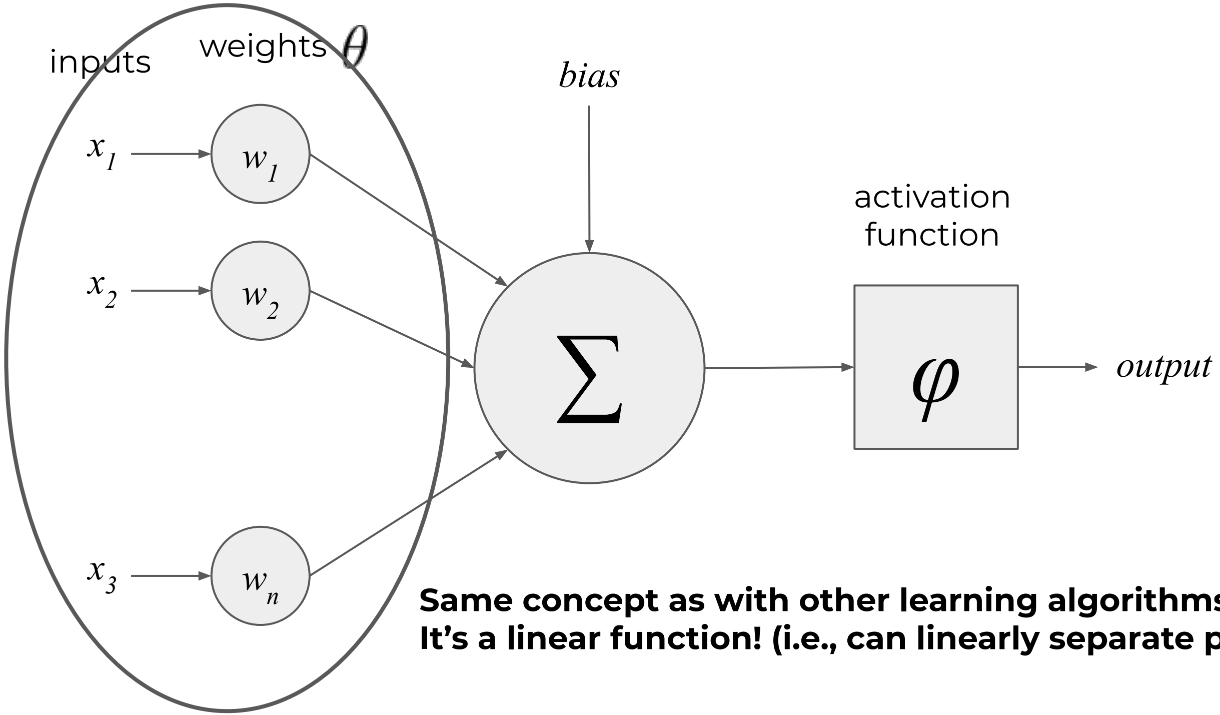
output

Adds non-linearity to the previous function. The typical functions are sigmoid, tanh, ReLU, etc. Sigmoid is the most used:

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

Perceptron

- **Structure of an artificial neuron**

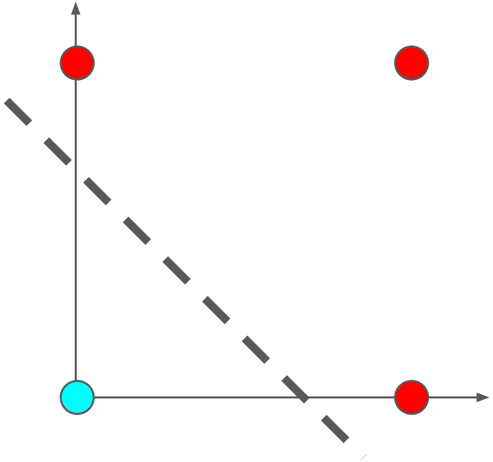


Frank Rosenblatt (1957)

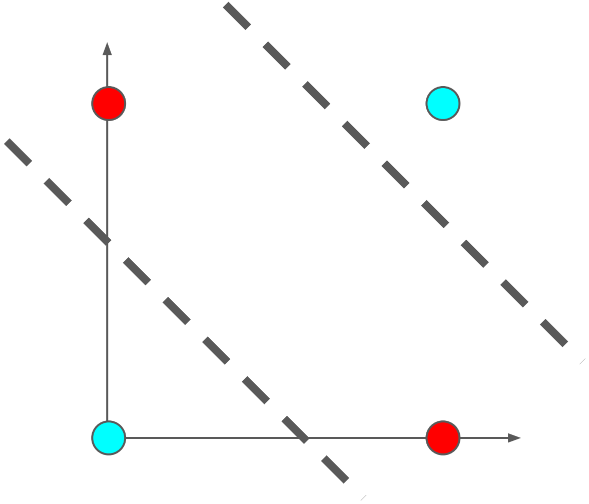
Same concept as with other learning algorithms: w and x.
It’s a linear function! (i.e., can linearly separate points)

Perceptron

- **Problem with non-linear functions**



OR function

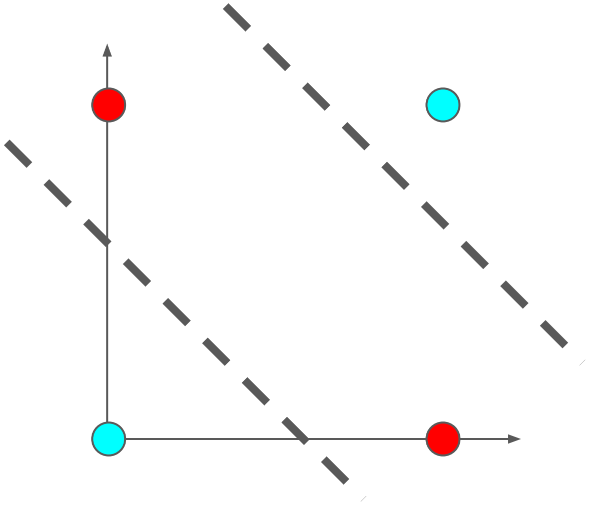


XOR function

Perceptron

- **A single layer neural network (NN) cannot solve non-linear problems (e.g., XOR)**

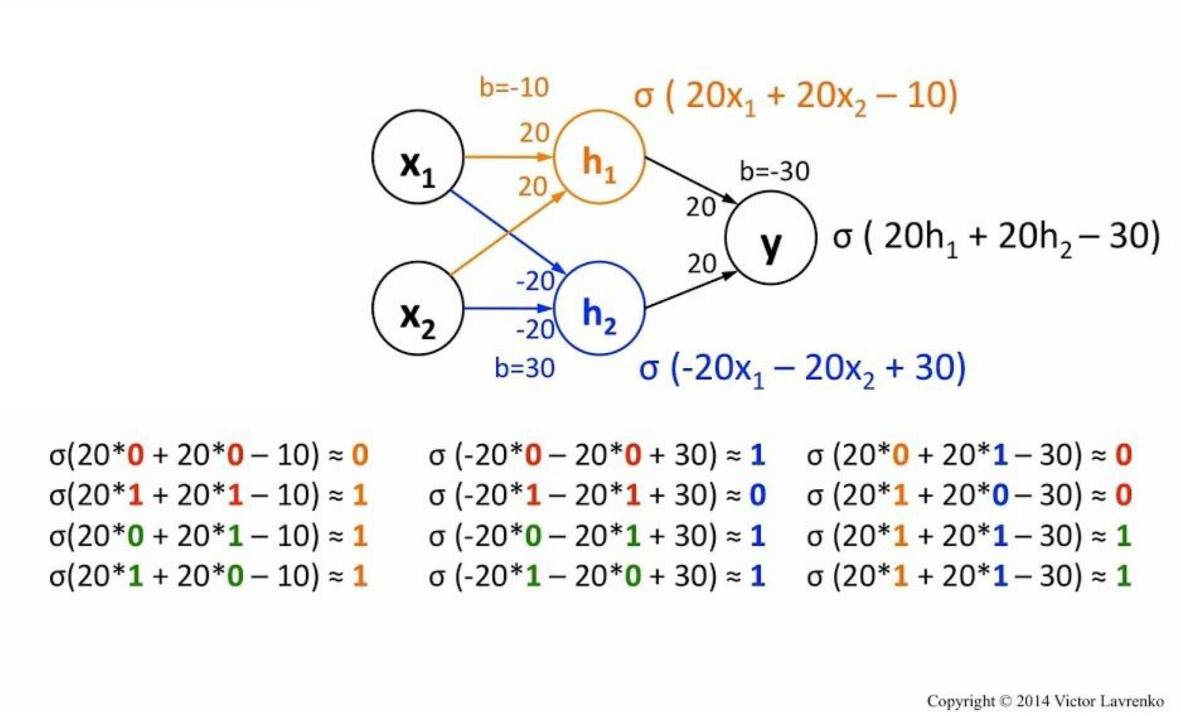
$$\begin{aligned}
0 \times w_1 + 0 \times w_2 + w_0 \leq 0 \quad &\Longleftrightarrow \quad w_0 \leq 0, \\
0 \times w_1 + 1 \times w_2 + w_0 > 0 \quad &\Longleftrightarrow \quad w_0 > -w_2, \\
1 \times w_1 + 0 \times w_2 + w_0 > 0 \quad &\Longleftrightarrow \quad w_0 > -w_1, \\
1 \times w_1 + 1 \times w_2 + w_0 \leq 0 \quad &\Longleftrightarrow \quad w_0 \leq -w_1 - w_2.
\end{aligned}$$



XOR function

XOR Problem

- **We need more than one layer to solve non-linear problems in 2D (e.g., XOR)**


b=-10
$\sigma\,(20x_1 + 20x_2 - 10)$
x_1
20
h_1
20
b=-30
20
y
$\sigma\,(20h_1 + 20h_2 - 30)$
20
-20
x_2
h_2
-20
b=30
$\sigma\,(-20x_1 - 20x_2 + 30)$

$\sigma(20*0 + 20*0 - 10) \approx 0$ $\quad$ $\sigma\,(-20*0 - 20*0 + 30) \approx 1$ $\quad$ $\sigma\,(20*0 + 20*1 - 30) \approx 0$

$\sigma(20*1 + 20*1 - 10) \approx 1$ $\quad$ $\sigma\,(-20*1 - 20*1 + 30) \approx 0$ $\quad$ $\sigma\,(20*1 + 20*0 - 30) \approx 0$

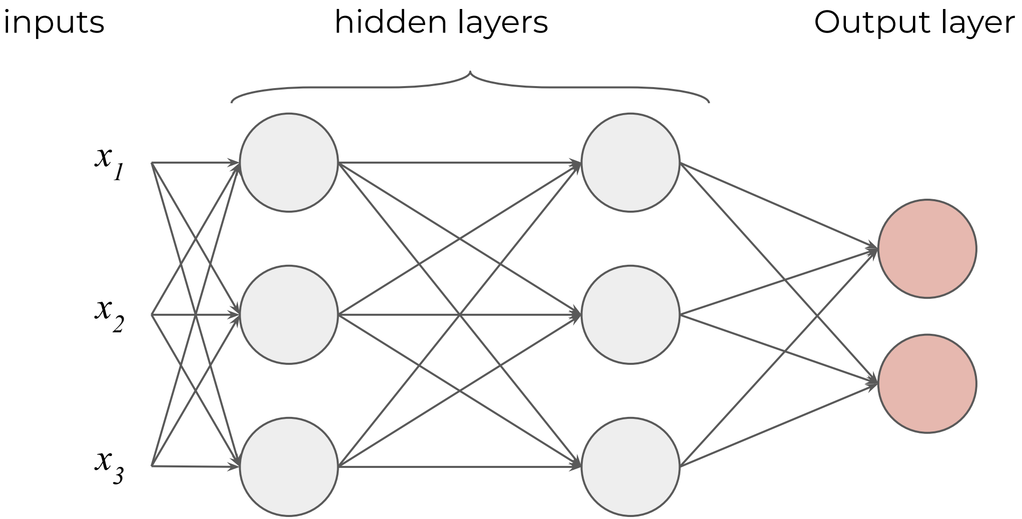
$\sigma(20*0 + 20*1 - 10) \approx 1$ $\quad$ $\sigma\,(-20*0 - 20*1 + 30) \approx 1$ $\quad$ $\sigma\,(20*1 + 20*1 - 30) \approx 1$

$\sigma(20*1 + 20*0 - 10) \approx 1$ $\quad$ $\sigma\,(-20*1 - 20*0 + 30) \approx 1$ $\quad$ $\sigma\,(20*1 + 20*1 - 30) \approx 1$

Copyright © 2014 Victor Lavrenko

Multi Layer Perceptron (MLP)

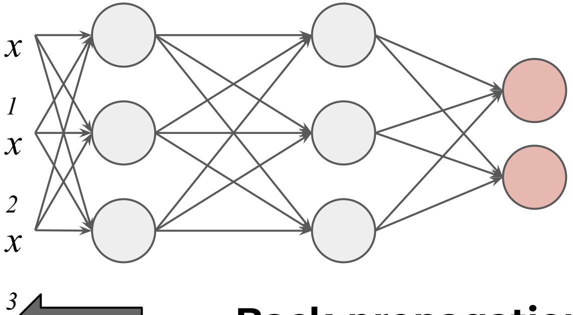
- **Connecting multiple neurons in multiple layers to model non-linear functions**



Back-Propagation

- Until 1980s, experts hand-engineered the features!
- There was a need to automatically learn the Internal weights of the networks

Feed forward
(Forward the outputs through layers)



Back-propagation
(Propagate back the gradients to update weights)

Hinton, Rumelhart, Williams and many other from 60s till 80s