Massachusetts Institute of Technology

ALFA
ANYSCALE LEARNING FOR ALL

CSAIL

Introduction to Neural Networks and Deep Learning

JAMAL TOUTOUH
toutouh@mit.edu
jamal.es
@jamtou

Jamal Toutouh, Ph.D.

- Postdoctoral Researcher at **Massachusetts Institute of Technology (MIT)**
  - MIT Computer Science & Artificial Intelligence Lab
  - Marie Sklodowska-Curie Postdoctoral fellowship

- PhD in Computer Science, **University of Malaga**
- M.Sc. in Software Engineering and Artificial Intelligence, **University of Malaga**
- M.Sc. in Information and Computer Sciences - Intelligent Systems, **University of Luxembourg**

- **Research Interests / Projects**
  - Scalable Machine/Deep Learning
  - Data Science and Large-scale Knowledge Mining
  - Smart Cities / Vehicular Communications
  - Evolutionary Algorithms / Swarm Intelligence

toutouh@mit.edu
www.jamal.es
www.necol.net
@jamtou

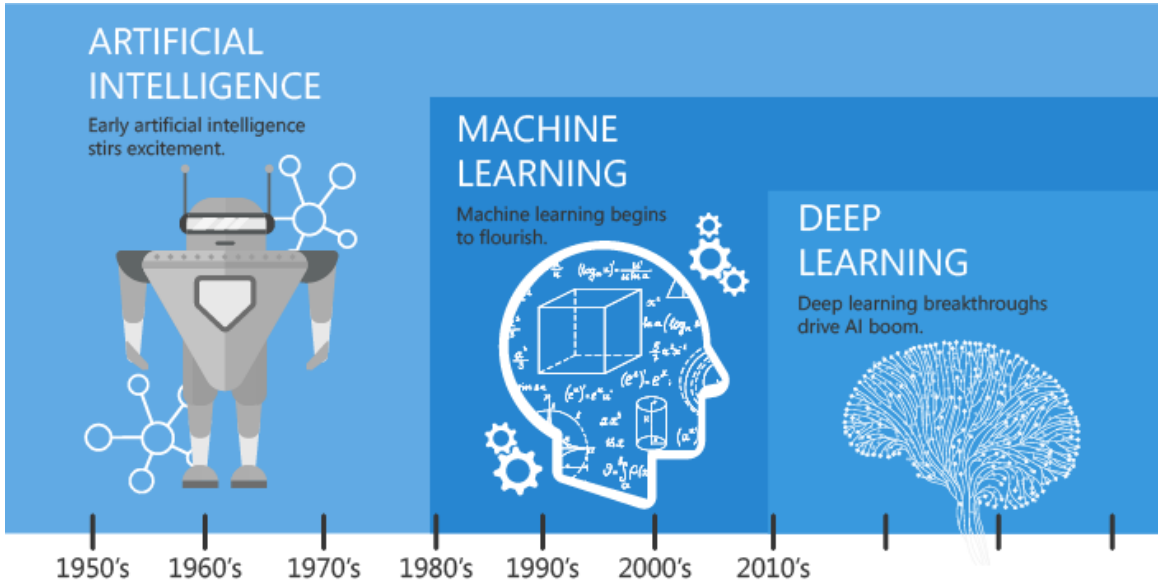
Outline

1. Main Concepts
2. Biological Neuron
3. Artificial Neuron
4. Activate Function
5. Artificial Neural Networks
6. How ANN Learn?

Find the code in my **GitHub** repository:
https://github.com/generative-adversarial-networks/introduction-neural-networks

1. Machine Learning, Deep Learning, AI ….

- **Artificial Intelligence** is human intelligence exhibited by computers
- **Machine Learning** is the field of study that gives the computers the ability to learn without being explicitly programmed
- **Deep Learning** uses deep neural networks to implement machine learning



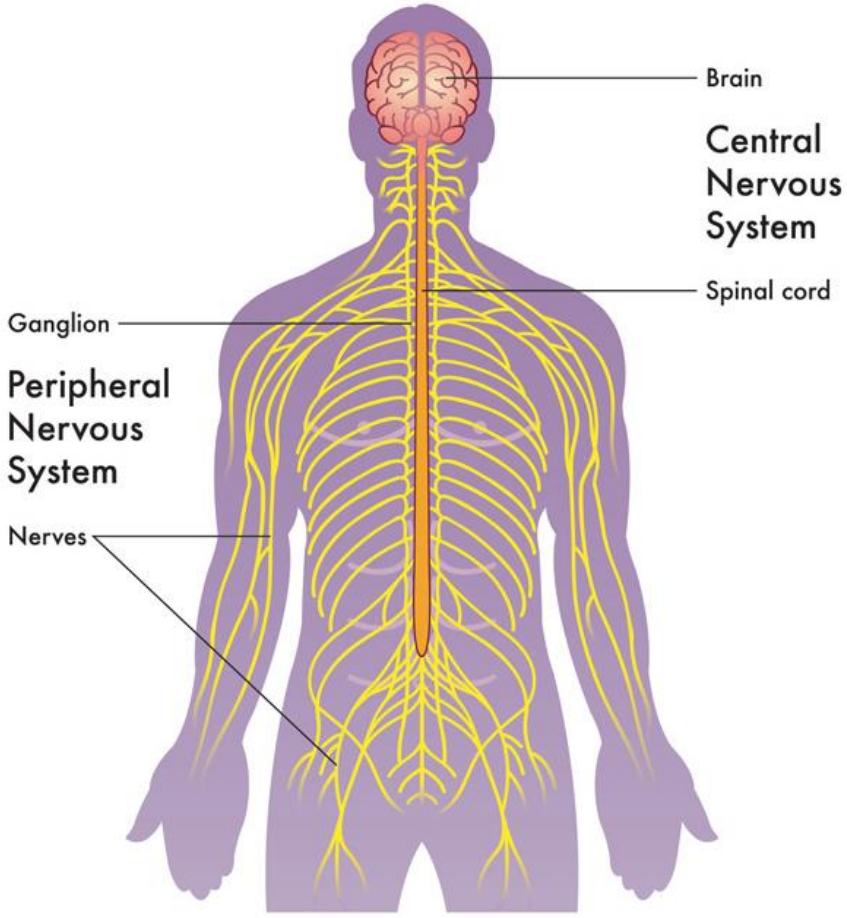
1. Artificial Intelligence Applications

- Facial recognition
- Game playing
- Speech recognition
- Language translation
- Self-driving cars
- Image translation: Edges to photo
- Fake images
- Fake videos
- …

2. Simple Biological Neuron

Biological Neurons are the **fundamental units of the brain and nervous system**, the cells responsible for receiving sensory input from the external world via dendrites, process it and gives the output through axons.

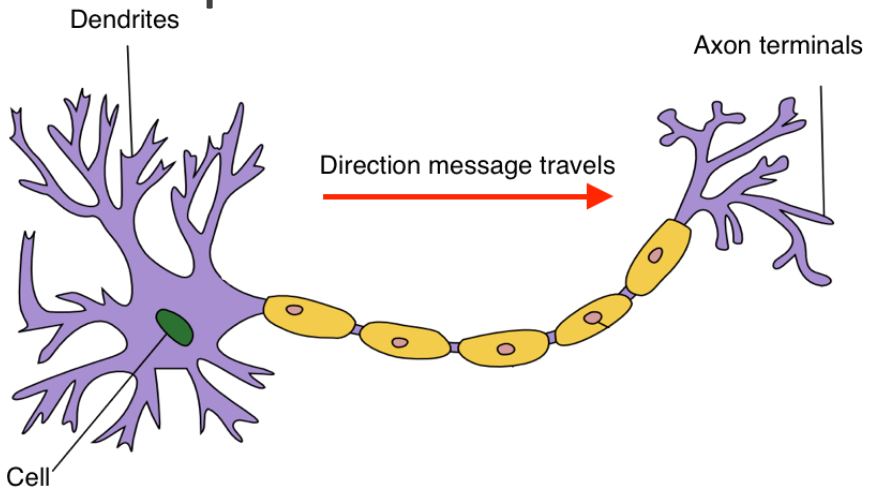
These cells, all 100 billion of them, are known as neurons. Each of these neurons can connect with up to 200,000 other neurons, although 1,000 to 10,000 is typical.

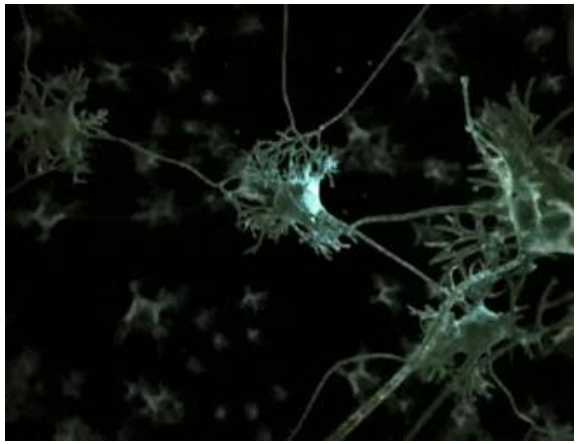


2. Simple Biological Neuron

A simple biological network has three major parts:

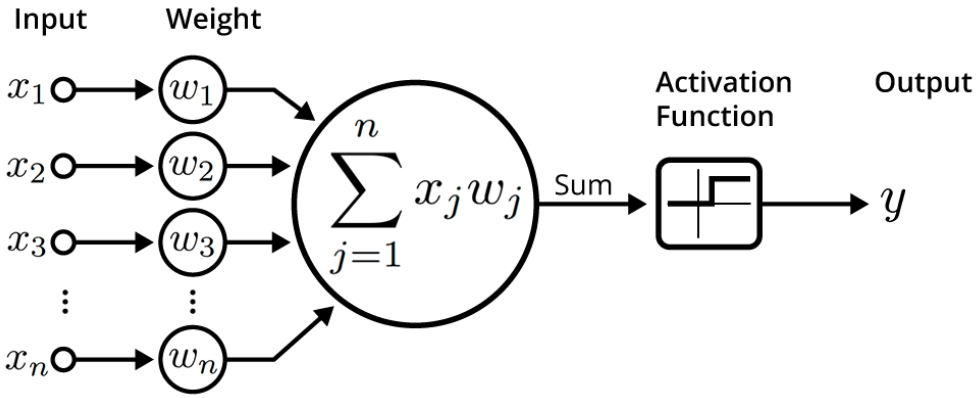
- **Dendrites:** They branch out into a tree around the cell body. They get incoming signals to cell body with their strength as weights.
- **Cell :** Collects input through dendrites and processes to produce output.
- **Axon:** Responsible for transmitting signals to other neurons.





3. Artificial Neuron

- An Artificial Neuron is a computational model of a biological neuron.
- The idea is that the artificial neuron receives input signals from other connected artificial neurons and via **a non-linear transmission function** emits a signal itself.
- Main operation:
  - Receives ***n inputs***
  - Computes the **weighted sum**
  - Passes through an **activation function**
  - Sends the signal to succeeding neurons



3. Artificial Neuron. Basic example

- Two-inputs neuron operation:
  1. Each input is multiplied by a weight

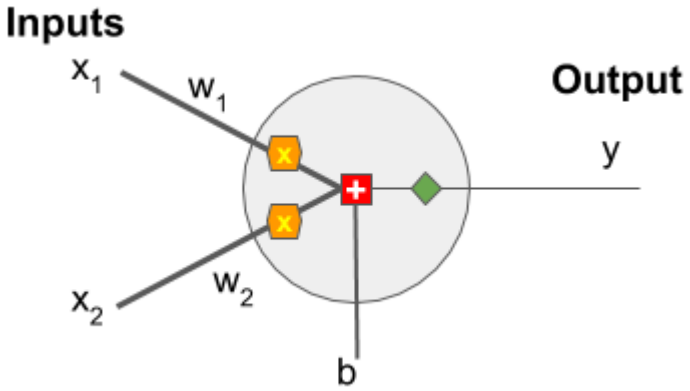
$$x_1 \rightarrow x_1 * w_1$$

$$x_2 \rightarrow x_2 * w_2$$

  2. All weighted sums are added with a bias b (***feedforward***) $(x_1 * w_1) + (x_2 * w_2) + b$

  3. The sum is passed through an activation function

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

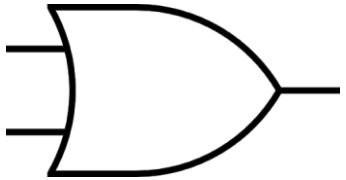


Code: [basic-neuron.py]

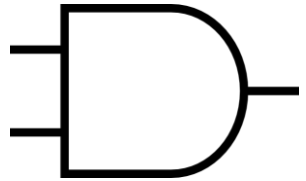
The output of the network depends on the **weights** and the **activation function**

3. Artificial Neuron. What can we do?

- Try to implement a logic function with the two-input neuron.



x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1



x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

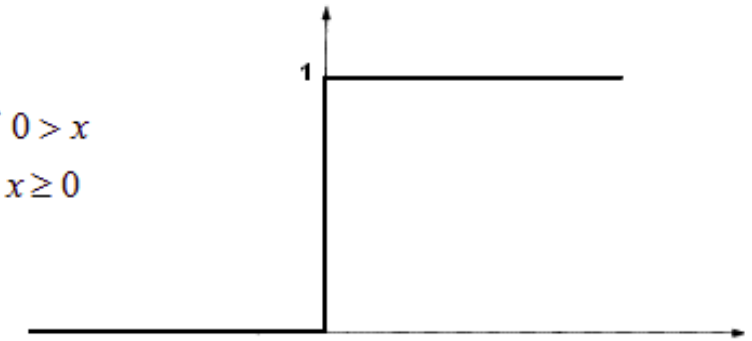
Code: *basic-neuron.py*

4. Activation Function

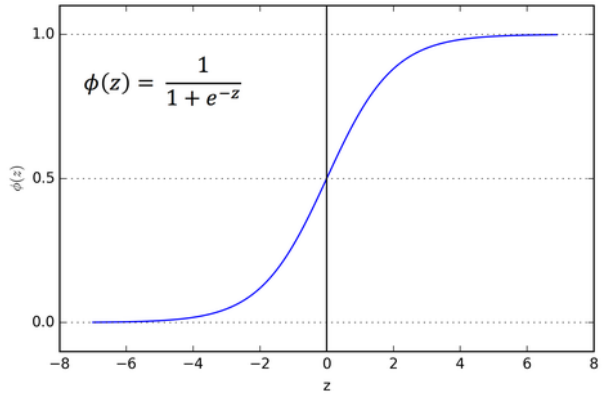
- Activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The motive is to **introduce non-linearity** into the output of a neuron.
- If we do not apply activation function then the output signal would be **simply linear function** (one-degree polynomial).
- Linear functions are limited in their complexity, have **less power**. Without activation function, our model cannot learn and model complicated data such as images, videos, audio, speech, etc.

4. Activation Function. Types

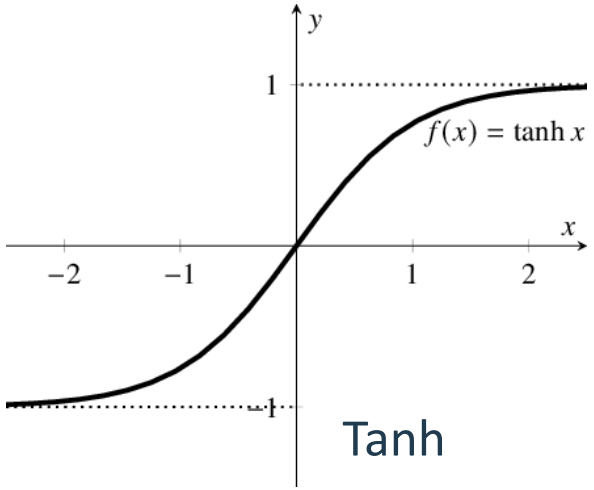
$$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



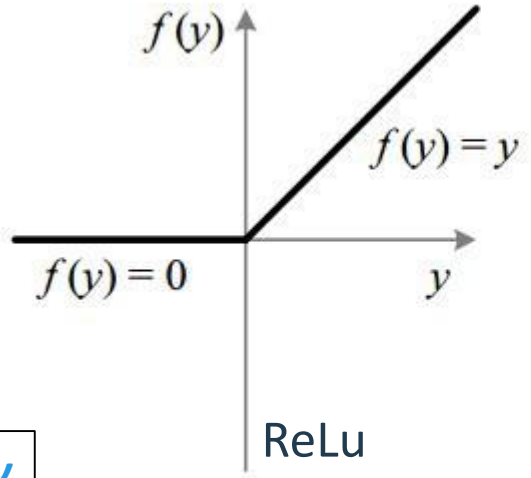
Step



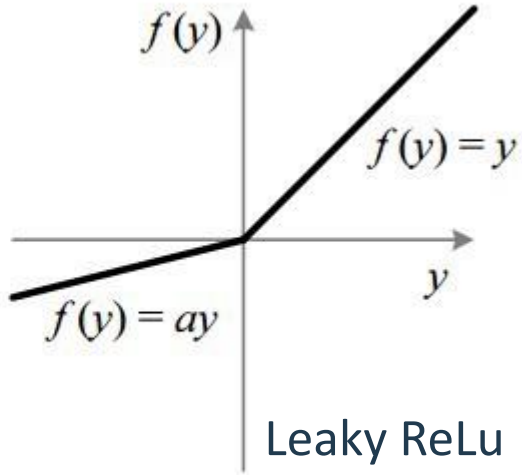
Sigmoid



Tanh



ReLu



Leaky ReLu

Code: *neuron.py*

5. Artificial Neural Networks

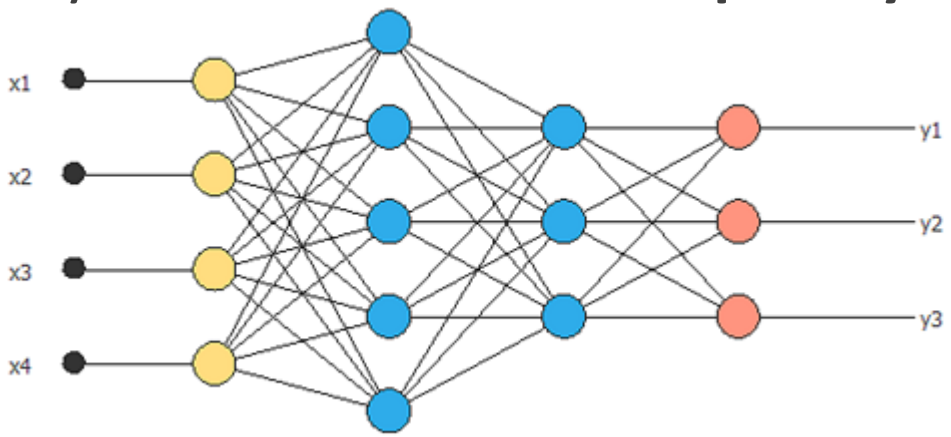
A neural network is a bunch of neurons connected together.

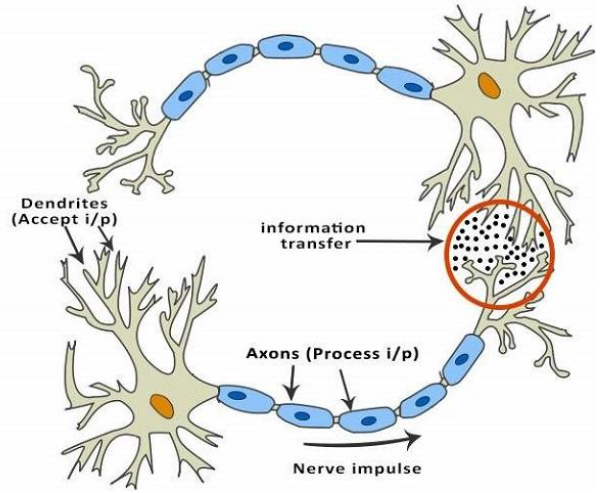
Neural networks are typically **organized in layers**.

Layers are made up of a number of **interconnected neurons.**

Inputs are presented to the network via the **input layer**, which communicates to **one or more hidden layers** through **weighted connections**.

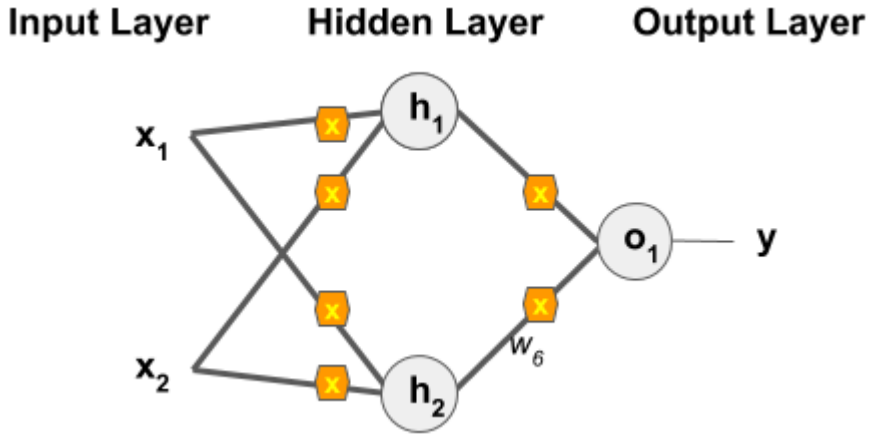
The hidden layers then link to an **output layer**.





5. Artificial Neural Networks. Code

- **Example 1**: Two inputs, two neurons in a hidden layer, and one output



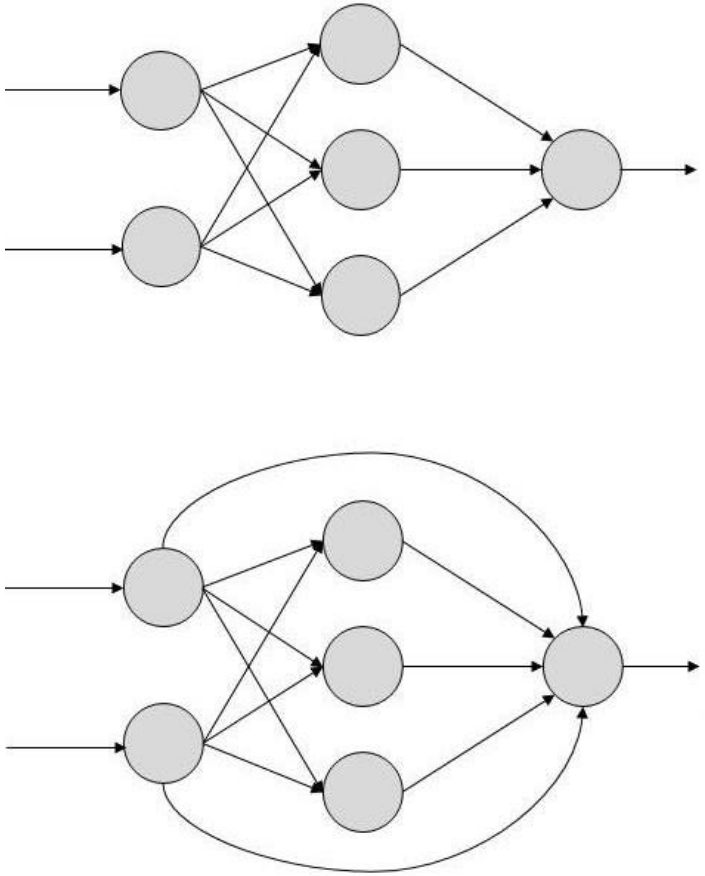
Code: ***basic-two-layer-neural-network.py***

- **Example 2**: X inputs, H neurons in the hidden layer, and one output

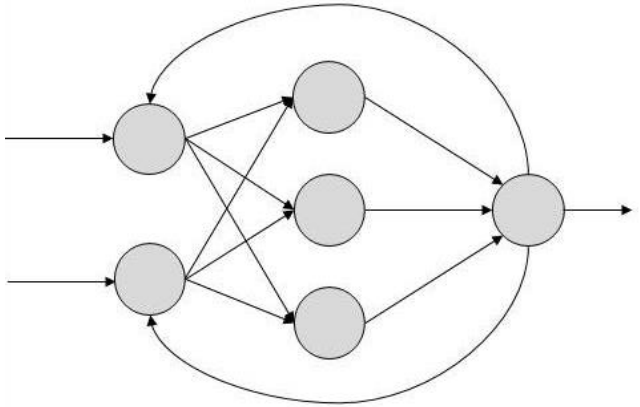
Code: ***two-layer-neural-network.py***

5. Artificial Neural Networks. Types

FeedForward ANN



Recurrent ANN



6. How do ANNs Learn?

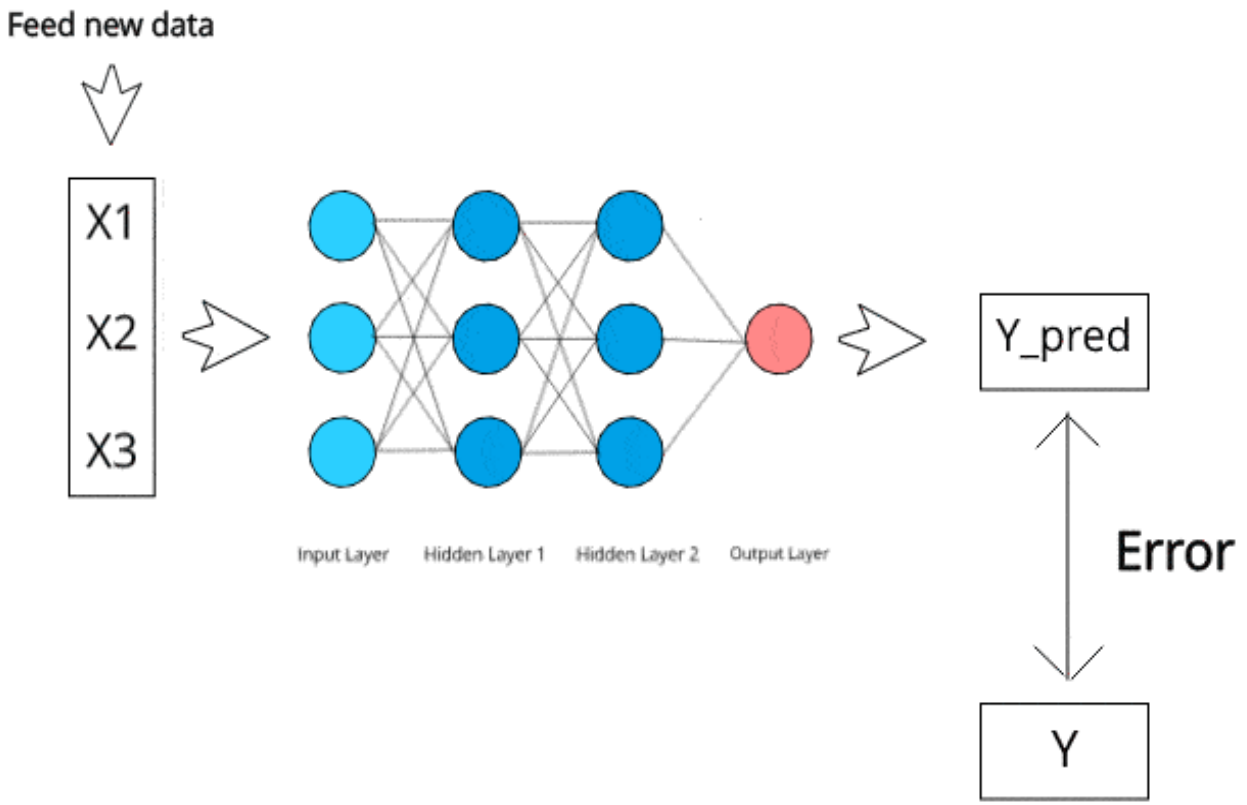
ANNs are capable of learning and they need to be **trained**.

Main types of learning:

- **Supervised Learning:** It involves a ***teacher*** that is scholar than the ANN itself. For example, the teacher feeds some example data about which the teacher already knows the answers.
- **Unsupervised Learning:** It is required when there is no example data set with known answers. For example, searching for a hidden pattern.
- **Reinforcement Learning:** This strategy built on observation. The ANN makes a decision by observing its environment. If the observation is negative, the network adjusts its weights to be able to make a different required decision the next time.

6. How do ANNs Learn?

ANNs learn by solving the **optimization problem** of reducing the error in terms of ***loss or cost function***.



Back Propagation Algorithm:
It learns by example. If you submit to the algorithm the example of what you want the network to do, it changes the network's weights so that it can produce desired output for a particular input on finishing the training.

[A really nice explanation here: https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a](https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a)

6. How do ANNs Learn? Example/Code

Example: We want to develop an AI system based on ANNs to automatically hire or discard candidates to be data scientist according two metrics: their experience programming and their GitHub projects. The training data are represented in the following table:

Name	Experience programming (in months)	GitHub projects	Hired
Michael	12	6	**No**
Shash	32	4	**Yes**
Hannah	11	3	**No**
Lisa	29	11	**Yes**

What is the prediction for **Jamal** that has **14 months** of experience and **6 GitHub** projects and **Mina** that has **31 months** of experience and **0 projects**?

6. How do ANNs Learn? Example/Code

We can start applying a given transformation to the data to ease the training process. For example, we subtract from each value the average and make the output binary (1=yes, 0=no)

Name	Experience programming	GitHub projects	Hired
Michael	12	6	**No**
Shash	32	4	**Yes**
Hannah	11	3	**No**
Lisa	29	11	**Yes**
Jamal	14	6	**?**
Mina	31	0	**?**

Average experience programming → 21

Average GitHub projects → 6

Name	Experience programming	GitHub projects	Hired
Michael	-9	0	**0**
Shash	11	-2	**1**
Hannah	-10	-3	**0**
Lisa	8	5	**1**
Jamal	-7	0	**?**
Mina	10	-6	**?**

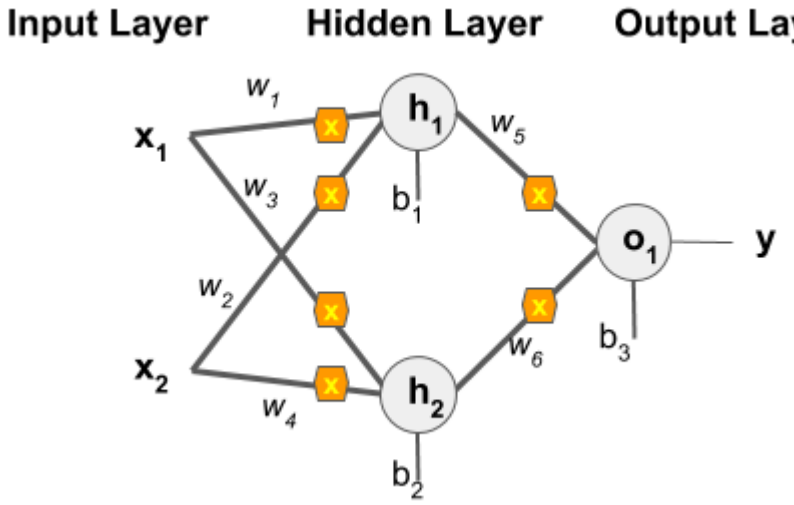
Training data

Test data

Code: *train_two-layer-neural-network.py*

6. How do ANNs Learn? Example/Code

We will use a two layer NN



Code: *train_two-layer-neural-network.py*

Configuration:

- Activation function: sigmoid
- Loss function: MSE
- Learning rate: 0.1

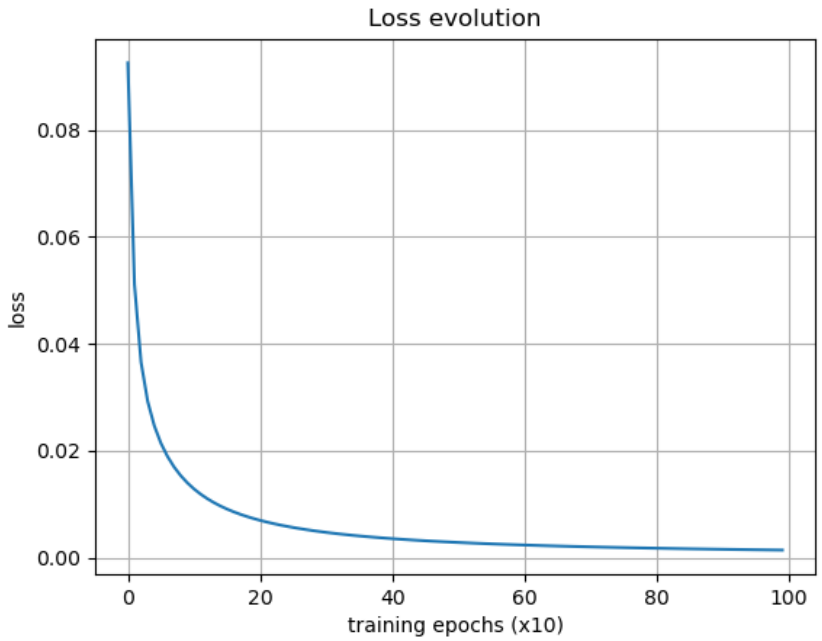
```
activation = ActivationFunctions().sigmoid
activation_derivative = ActivationFunctions.derivative_sigmoid
loss = NetoworksLoss.mse_loss
learning_rate = 0.1
```

Change the configuration and see the different results

6. How do ANNs Learn? Example/Code

Results (they may vary):

- Training:



Prediction:

- Jamal will not be hired → output=0.037
- Mina will be hired → output=0.797

Massachusetts Institute of Technology

ALFA
ANYSCALE LEARNING FOR ALL

Thanks!

Comments?

JAMAL TOUTOUH

toutouh@mit.edu
www.jamal.es
www.necol.net
@jamtou