ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

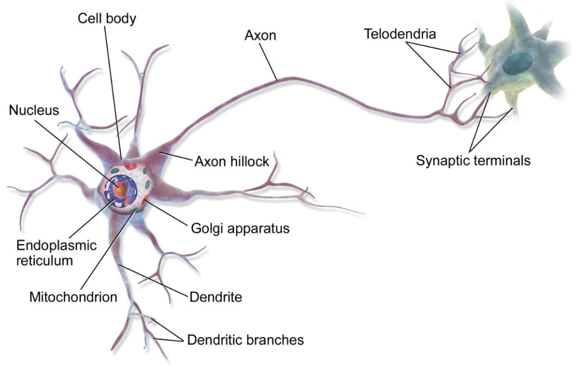
Intelligenza Artificiale

Introduction to Artificial Neural Networks

Ivan Heibi
Dipartimento di Filologia Classica e Italianistica (FICLIT)
Ivan.heibi2@unibo.it

Neurons

- **Biological neurons** receive short electrical impulses called signals from other neurons via these synapses
- When a neuron receives a sufficient number of signals from other neurons within a few milliseconds, it fires its own signals
- Neurons are organized in a vast **network** of billions of neurons, each neuron typically connected to thousands of other neurons.



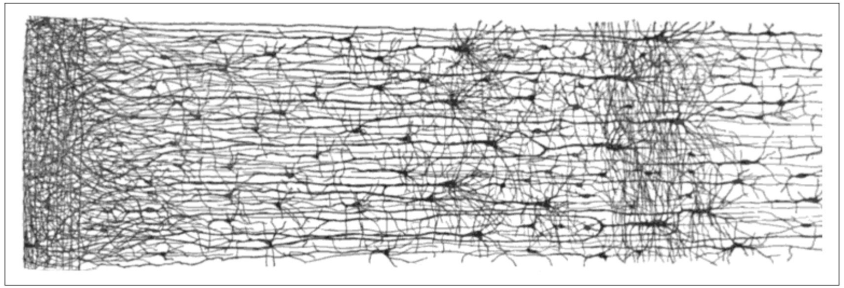


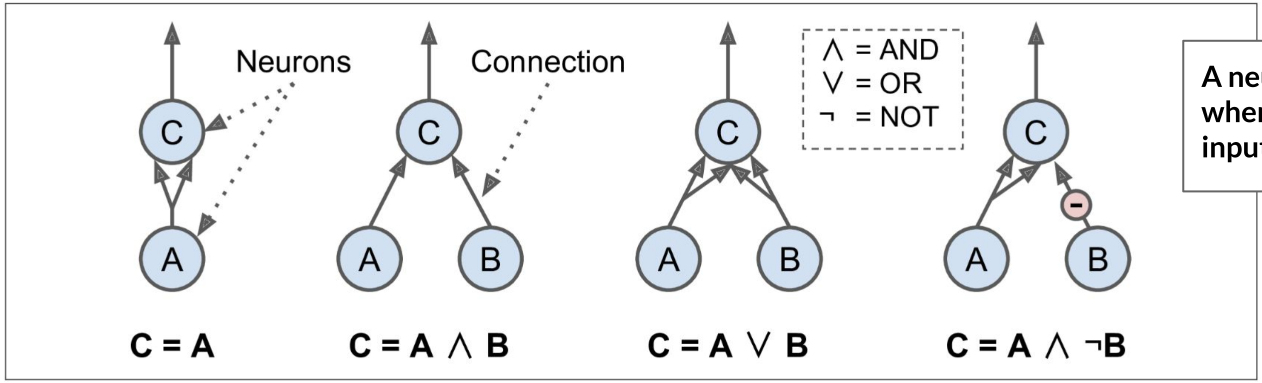
Figure 10-2. Multiple layers in a biological neural network (human cortex)[5]

Logical computations with neurons

McCulloch and Pitts proposed a very simple model of the biological neuron, which later became known as an artificial neuron: it has one or more binary (on/off) inputs and one binary output.

The artificial neuron simply activates its output when more than a certain number of its inputs are active.

McCulloch and Pitts showed that even with such a simplified model it is possible to build a network of artificial neurons that computes any logical proposition you want.



A neuron is <u>activated</u> when <u>at least two</u> of its inputs are active.

Figure 10-3. ANNs performing simple logical computations

The Perceptron – Threshold Logic Unit (TLU)

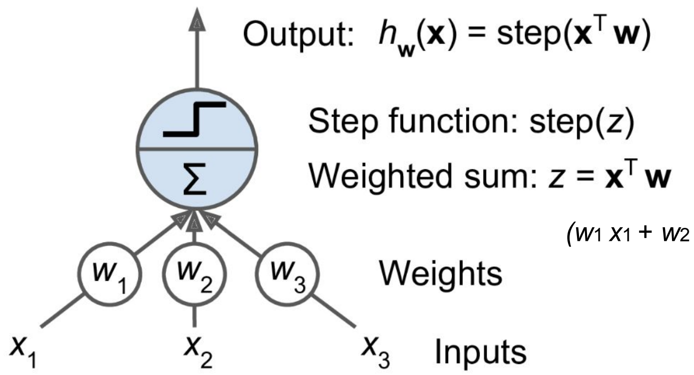
The **Perceptron** is one of the simplest ANN architectures (invented in 1957 by Frank Rosenblatt)

It is based on a slightly different artificial neuron called a **threshold logic unit (TLU)**.

A single TLU can be used for simple linear binary classification.

It computes a linear combination of the inputs and if the result exceeds a threshold, it outputs the positive class or else outputs the negative class.

Training a TLU in this case means finding the right values for w1, w2, and w3 .

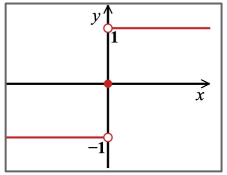


Output: $h_{\mathbf{w}}(\mathbf{x}) = \text{step}(\mathbf{x}^T \mathbf{w})$
Step function: step(z)
Weighted sum: $z = \mathbf{x}^T \mathbf{w}$
Σ
w_1
w_2
w_3
Weights
x_1
x_2
x_3
Inputs

$(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)$

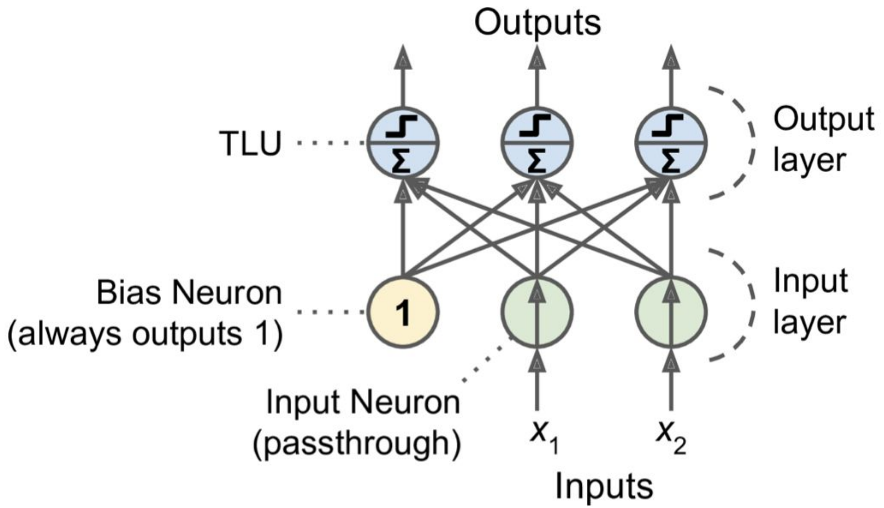
Equation 10-1. Common step functions used in Perceptrons

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$



The Perceptron

A Perceptron is simply composed of a single layer of TLUs, with each TLU connected to all the inputs.



When all the neurons in a layer are connected to every neuron in the previous layer (i.e., its input neurons), it is called a **fully connected layer** or a **dense layer**.

How is a Perceptron trained?

The Perceptron training algorithm proposed by Frank Rosenblatt was largely inspired by Hebb's rule.

Hebb suggested that when a biological neuron often triggers another neuron, the connection between these two neurons grows stronger, i.e. : the connection weight between two neurons is increased whenever they have the same output.

Perceptrons are trained using a variant of this rule that takes into account the error made by the network; it reinforces connections that help reduce the error.

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta\left(y_j - \hat{y}_j\right)x_i$$

- $w_{i,j}$ is the connection between the ith input neuron and jth output neuron
- x_i is the ith value of the current training instance
- y_i (hat) is the output of the jth output neuron
- y_i is the target output of the jth output neuron
- (eta) is the learning rate

Perceptron example

Let's assume a Perceptron with two inputs is classifying images of cats and dogs.

The output of the perceptron $\widehat{\mathrm{y}}_j = 0.7$ (the probability that the image is a dog).

My desire is $\mathrm{y}_j = 1$

If we have:

$$\mathrm{x}_1 = average\_pixel\_intensity$$

$$\mathrm{x}_2 = texture\_consistency$$

$$\eta = 0.1$$

Then the weights are updated as follow:

$$\mathrm{w}_{1,1}^{(next\_step)} = \mathrm{w}_{1,1} + 0.1 * (1 - 0.7) * \mathrm{x}_1$$

$$\mathrm{w}_{1,2}^{(next\_step)} = \mathrm{w}_{1,2} + 0.1 * (1 - 0.7) * \mathrm{x}_2$$

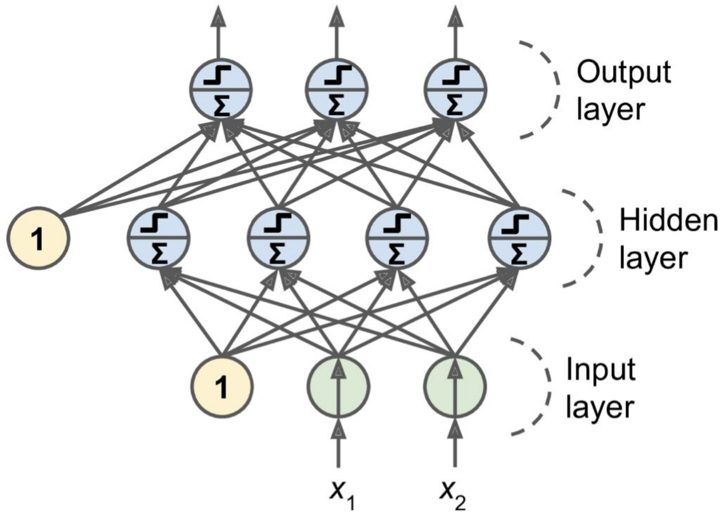
> The two weights would be <u>increased</u> if both x_1 and x_2 are positive (i.e., the image has typical dog features) and <u>decreased</u> if both x_1 and x_2 are both negative.

Multi-Layer Perceptrons

An MLP is composed by:

- one (passthrough) **input layer,**
- one or more layers of TLUs, **called hidden layers**,
- and one final layer of TLUs called the **output layer**.

For many years researchers struggled to find a way to train MLPs, without success. But in 1986, Rumelhart, Hinton and Williams published a groundbreaking paper introducing the backpropagation training algorithm, which is still used today [1].

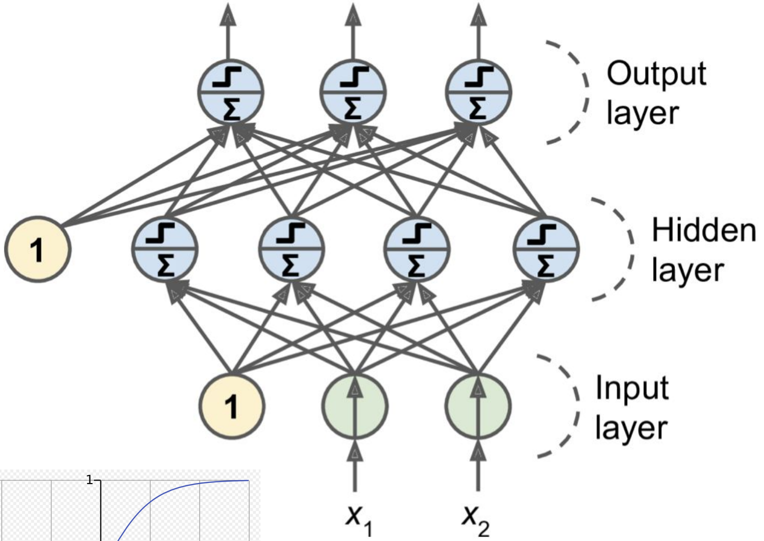


[1] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

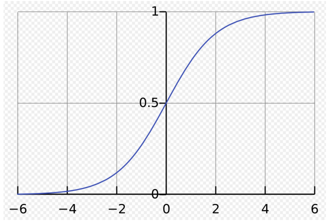
Backpropagation

For each training instance:

- **Forward pass:** feed the network and pass through all the layers until we get the output layer
- **Measure the error** (i.e. how much the output differs from the expected output)
- **Backward pass:** go through each layer in reverse to measure the error contribution of each connection
- **Gradient Descent step:** tweak all the connection weights in the network, using the error gradients



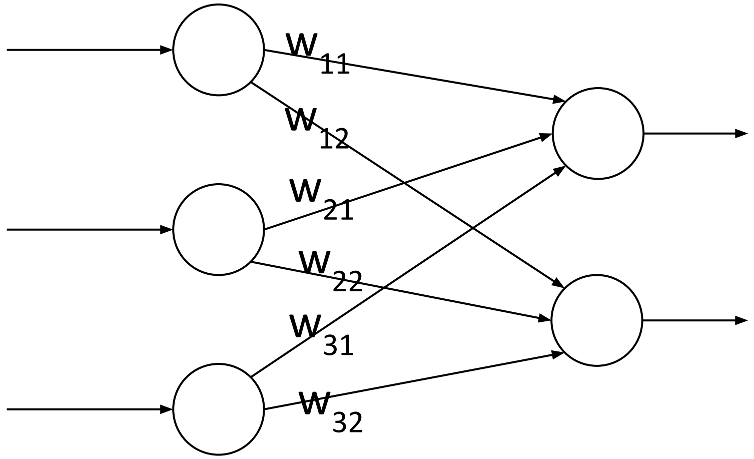
Rumelhart et al. replaced the
step function with the **logistic function (sigmoid)**
σ(x) = 1 / (1 + exp(–x)).



Feed forward neural networks

A feed-forward network has connections only in one direction.

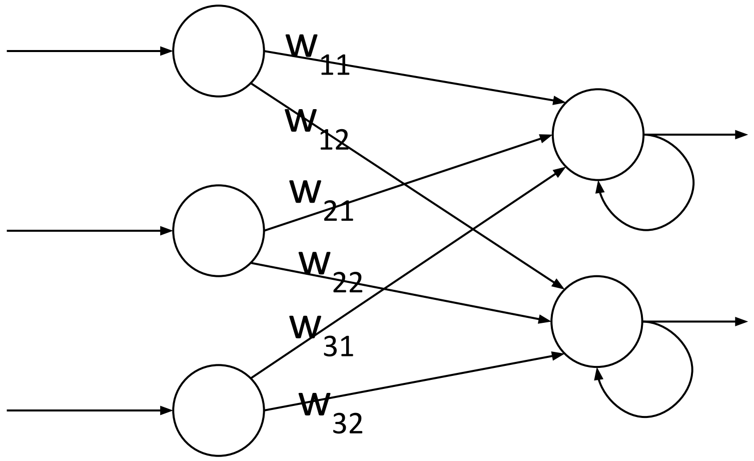
A feed-forward neural network represents a function of its current input <u>(no internal state)</u>.



Recurrent neural networks

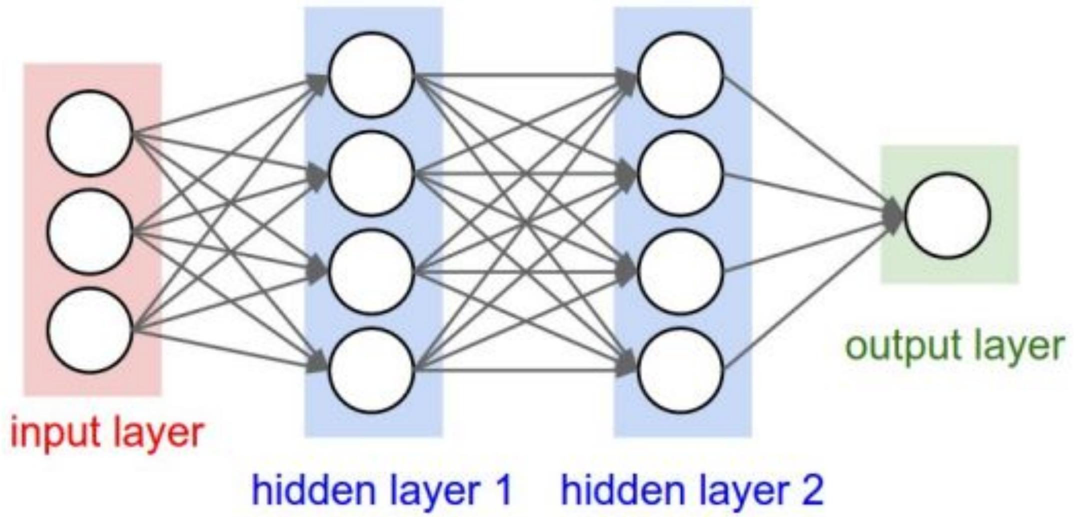
A recurrent neural network feeds its outputs back into its inputs.

This means that the response of the network to a given input depends on its initial state, which may depend on previous inputs. Hence they can support short term memory.



Deep Neural Network

A deep neural network is an artificial neural network with multiple layers between the input and output layers.



Deep Neural Network usage

The choice of the neural network type depends on various factors

- **Task:**
  DNNs are well-suited for complex tasks such as image recognition, automatic translation, and text generation. However, for simpler tasks like binary classification or linear regression, simpler neural networks like perceptrons might be sufficient.

- **Dataset size**
  DNNs typically require large amounts of data to learn effectively. A simpler neural network might be more efficient with smaller datasets.

- **Availability of computational resources**
  Training a DNN can be time-consuming and computationally demanding. It might be necessary to use a simpler neural network, when the computational resources are limited,

Deep Neural Network usage

https://playground.tensorflow.org/#activation=sigmoid&batchSize=16&dataset=xor&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=1&seed=0.31628&showTestData=false&discretize=false&percTrainData=30&x=true&y=true&xTimesY=false&xSquared=true&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false