Contents

1. Neural Networks

1.1. Biological foundations

Neurons are first and foremost studied by neurobiology and neurophyisiology. The interest of artificial intelligence is to mimic the way biological neurons work, so that the same model can be applied to non-living beings. In particular, the interest is to study the way living beings collect information through senses, the way they process this collected information and the way they learn from experience.

Neurons have a a core in the form of the nucleus that receives information from other neurons collected information. When the nucleus receives a sufficient aumount of stimulation, it releases back information on nearby neurons. The connection between the simulated neuron and the stimulating one is called **synapsis**; an excited neuron induces the synapsis to release chemicals called **neurotransmitters**, received from the **dendrites** of the receiving neuron.

If a neuron receives enough stimulation from its dendrites, it decides to send in turn a signal to other neurons through an electric signal. The **axon** propagate the electric stimulus from the dendrites to the nucleus. When a neuron sends an electric signal, we say that the neuron *fired*.

A real computer cannot, as is, completely capture the complexity of a real brain.

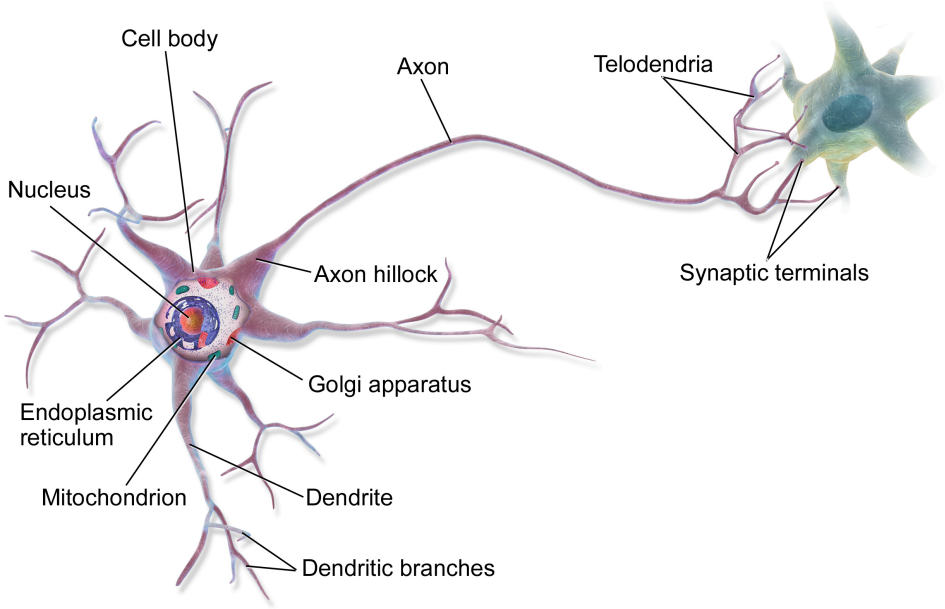


Figure 1: Schematic image of a neuron. By BruceBlaus, CC BY 3.0, via Wikimedia Commons. original image

Advantages of neural networks:

- High parallelism, which entails speedup;
- Fault tolerance, even if a large part of the network is failing the overall network might still work (not always, but close to);
- If some neurons get degraded, we slowly loose our capabilities, but never abruptly. Failing nodes can be phased slowly.

In first approximation, any living being has an input facility (smell, touch, taste), which deliver information to a neuron pool connected to an output. The idea is to have a model that approximates this structure but without the "living being" part.

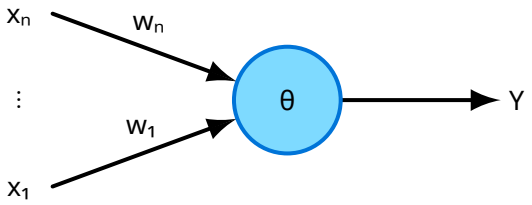
1.2. Threshold Logic Units

A **Threshold Logic Unit** (**TLU**), also known as **perceptron**[1] or **McCulloch-Pitts neuron** is a mathematical structure that models, in a very simplified way, how neurons operate.

[1]The original definition of perceptron was more refined than a TLU, but the two terms are often used interchangeably.

A TLU has n binary inputs $x_1, x_2, ..., x_n$, each weighted by a weight $w_1, w_2, ..., w_n$, that generates a single binary output y. If the sum of all the inputs multiplied by their weights is a value greater or equal than a given threshold θ, the output y is equal to 1, otherwise is equal to 0.

The analogy between TLUs and biological neurons is straightforward. The output of a TLU is analogous to the firing of a neuron: an output equal to 1 corresponds to the firing of a neuron, whereas an output equal to 0 corresponds to a neuron insufficiently stimulated to be firing.



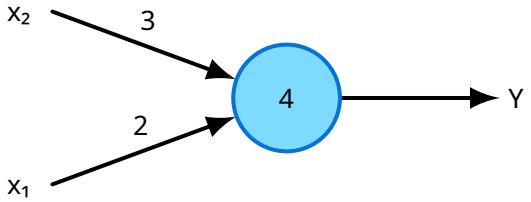
$$y = \begin{cases} 1 \text{ if } \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0 \text{ otherwise} \end{cases}$$

Figure 2: A common way of representing a TLU. The processing unit is drawn as a circle, with the threshold in its center. Inputs are drawn as arrows entering the TLU from the left, with their respective weights above. The output is an arrow exiting the TLU from the right.

The inputs can be collected into a single input vector $\boldsymbol{x} = (x_1, ..., x_n)$ and the weights into a weight vector $\boldsymbol{w} = (w_1, ..., w_n)$. With this formalism, the output y is equal to 1 if $\langle \boldsymbol{w}, \boldsymbol{x} \rangle \geq \theta$, where $\langle\rangle$ denotes the scalar product.

Exercise 1.2.1: Construct a TLU with two inputs whose threshold is 4 and whose weights are $w_1 = 3$ and $w_2 = 2$.

Solution:



x_1	x_2	$3x_1 + 2x_2$	y
0	0	0	0
1	0	3	0
0	1	2	0
1	1	5	1

□

Intuitively, a negative weight corresponds to an inhibitory synapse: if the corresponding input becomes active (that is, equal to 1), it gives a negative contribution to the overall excitation. On the other hand, a positive weight corresponds to an excitatory synapse: if the corresponding input becomes active (that is, equal to 1), it gives a positive contribution to the overall excitation.

Note how the weighted summation that discriminates whether the output of a TLU is 1 or 0 is very similar to an n-dimensional linear function. That is, by substituting the $\geq$ sign with a $=$ sign, it effectively turns into an n dimensional straight line:

$$\sum_{i=1}^{n} w_i x_i = \theta \Rightarrow \sum_{i=1}^{n} w_i x_i - \theta = 0 \Rightarrow w_1 x_1 + w_2 x_2 + ... + w_n x_n - \theta = 0$$

As a matter of fact, the line $\sum_{i=1}^{n} w_i x_i - \theta = 0$ acts as a **decision border**, partitioning the n-dimensional Euclidean hyperplane into two half-planes: one containing n-dimensional points that give an output of 1 when fed the TLU and the other containing points that give an output of 0.

To deduce which of the two regions of space is which, it suffices to inspect the coefficients of the line equation. Indeed, the coefficients $x_1, ..., x_n$ are the elements of a normal vector of the line: the half-plane that contains points that give the TLU an output of 1 is the one to which this vector points to.

Unfortunately, not all linear functions can be represented by a TLU. More formally, two sets of points are called **linearly separable** if there exists a linear function, called **decision function**, that partitions the Euclidean hyperplane into two half-planes, each containing one of the two sets.

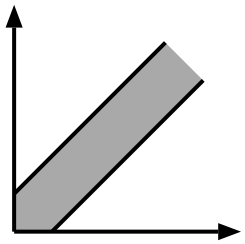
A set of points in the plane is called **convex** if connecting each point of the set with straight lines does not require to go outside of the set. The **convex hull** of a set of points is its the smallest superset that is convex. If two sets of points are both convex and disjoint, they are linearly separable.

A TLU is capable of representing only functions such as these, but for two sets of points a decision function might not exist, and therefore not all sets of points are linearly separable.

Exercise 1.2.2: Is the function $A \leftrightarrow B$ linearly separable?

Solution: No, and it can be proven.

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	1

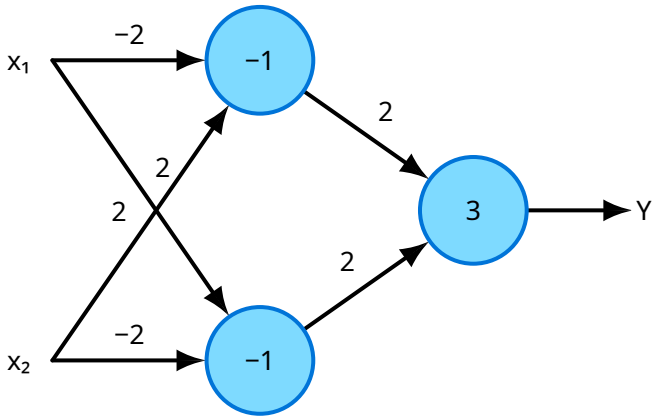


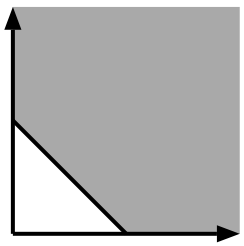
□

Even though single TLPs are fairly limited, it is possible to chain more TLPs together, creating a *network* of threshold logic units. This can be done by breaking down a complicated boolean function into approachable functions, each representable by a TLU, and using the outputs of TLUs as inputs of other TLUs. Since both the inputs and the outputs of a TLP are binary values, this doesn't pose a problem. By applying a coordinate transformation, moving from the original domain to the image domain, the set of points become linearly separable.

Exercise 1.2.3: Is it possible to construct a network of threshold logic units that can represent $A \leftrightarrow B$?

Solution: Yes. Note how $A \leftrightarrow B$ can be rewritten as $(A \to B) \wedge (B \to A)$. Each of the three functions (two single implications and one logical conjunction) is linearly separable.





□

It can be shown that all Boolean functions with an arbitrary number of inputs can be computed by networks of TLUs, since any Boolean function can be rearranged in the disjunctive normal form (or conjunctive normal form). A Boolean function in disjunctive normal form is only constituted by a streak of `or` each constituted by `and` (potentially negated), which are all linearly separable.

In particular, a TLU network of two layers will suffice: let $y = f(x_1, ..., x_n)$ be a Boolean function of n variables. It is possible to construct a network of threshold logic units that represents y applying this algorithm:

1. Rewrite the function y in disjunctive normal form:

$$D_f = K_1 \vee ... \vee K_m = \left(l_{1,1} \wedge ... \wedge l_{1,n}\right) \vee ... \vee \left(l_{m,1} \wedge ... \wedge l_{m,n}\right) = \bigvee_{j=1}^{m} \left(\bigwedge_{i=1}^{n} l_{j,i} \right)$$

   Where each $l_{j,i}$ can be either non-negated (positive literal) or negated (negative literal)
2. For each K_j construct a TLU having n inputs (one input for each variable) and the following weights and threshold:

$$w_{j,i} = \begin{cases} 2 \text{ if } l_{i,j} \text{ is a positive literal} \\ -2 \text{ if } l_{i,j} \text{ is a negative literal} \end{cases} \qquad \theta_j = n - 1 + \frac{1}{2} \sum_{i=1}^{n} w_{j,i}$$

3. Create one output neuron, having m inputs (equal to the number of TLUs created in the previous steps), threshold equal to 1 and all weights equal to 2.