How do NNs and ANNs work?

- NNs are able to **learn** by **adapting their connectivity patterns** so that the organism improves its behavior in terms of reaching certain (evolutionary) goals.
- The strength of a connection, or whether it is excitatory or inhibitory, depends on the state of a receiving neuron's **synapses**.
- The NN achieves **learning** by appropriately adapting the states of its synapses.

The Net Input Signal

The **net input signal** is the sum of all inputs after passing the synapses:

$$\text{net}_i(t) = \sum_{j=1}^{n} w_{i,j}(t)x_j(t)$$

This can be viewed as computing the **inner product** of the vectors $w_i$ and $x$:

$$\text{net}_i(t) = \| w_i(t) \| \cdot \| x(t) \| \cdot \cos \alpha,$$

where $\alpha$ is the **angle** between the two vectors.

The Activation Function

One possible choice is a **threshold function**:

$$f_i(\text{net}_i(t)) = \begin{cases} 1, & \text{if } \text{net}_i(t) \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

The graph of this function looks like this:
**_binary analogy: threshold logic units**
Example:

\[ x_1 w_1 = 1 \]
\[ x_2 w_2 = 1 \]
\[ x_3 w_3 = -1 \]
\[ \theta = 1.5 \]
\[ x_1, x_2, x_3 \]

**networks**

Yet another example:

\[ x_1 \]
\[ w_1 = \]
\[ x_2 \]
\[ w_2 = \]
\[ \theta = \]

\[ x_1 \oplus x_2 \]

XOR

Impossible! TLUs can only realize **linearly separable** functions.

---

**linear separability**

A function \( f: \{0, 1\}^n \rightarrow \{0, 1\} \) is linearly separable if the space of input vectors yielding 1 can be separated from those yielding 0 by a **linear surface** (hyperplane) in \( n \) dimensions. Examples (two dimensions):

![Linear Separability - Two Dimensions](image)

linearly separable

linearly inseparable

To explain linear separability, let us consider the function \( f: \mathbb{R}^n \rightarrow \{0, 1\} \) with

\[
f(x_1, x_2, \ldots, x_n) = 1, \quad \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq \theta
\]

\[
= 0, \quad \text{otherwise}
\]

where \( x_1, x_2, \ldots, x_n \) represent real numbers.

This will also be useful for understanding the computations of **artificial neural networks**.

---

**linear separability**

So by varying the weights and the threshold, we can realize any **linear separation** of the input space into a region that yields output 1, and another region that yields output 0.

As we have seen, a **two-dimensional** input space can be divided by any straight line. A **three-dimensional** input space can be divided by any two-dimensional plane.

In general, an **n-dimensional** input space can be divided by an \((n-1)\)-dimensional plane or hyperplane.

Of course, for \( n > 3 \) this is hard to visualize.
Linear Separability
Of course, the same applies to our original function $f$ using binary input values. The only difference is the restriction in the input values. Obviously, we cannot find a straight line to realize the XOR function:

$$
\begin{array}{ccc}
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{array}
$$

In order to realize XOR with TLUs, we need to combine multiple TLUs into a network.

Multi-Layered XOR Network

Capabilities of Threshold Neurons
What can threshold neurons do for us? To keep things simple, let us consider such a neuron with two inputs:

The computation of this neuron can be described as the inner product of the two-dimensional vectors $x$ and $w_i$, followed by a threshold operation.

Capabilities of Threshold Neurons
Let us assume that the threshold $\theta = 0$ and illustrate the function computed by the neuron for sample vectors $w_i$ and $x$:

Since the inner product is positive for $-90^\circ < \alpha < 90^\circ$, in this example the neuron's output is 1 for any input vector $x$ to the right of or on the dotted line, and 0 for any other input vector.

Capabilities of Threshold Neurons
By choosing appropriate weights $w_i$ and threshold $\theta$ we can place the line dividing the input space into regions of output 0 and output 1 in any position and orientation. Therefore, our threshold neuron can realize any linearly separable function $R^n \rightarrow \{0, 1\}$. Although we only looked at two-dimensional input, our findings apply to any dimensionality $n$. For example, for $n = 3$, our neuron can realize any function that divides the three-dimensional input space along a two-dimension plane.

Capabilities of Threshold Neurons
What do we do if we need a more complex function? Just like Threshold Logic Units, we can also combine multiple artificial neurons to form networks with increased capabilities. For example, we can build a two-layer network with any number of neurons in the first layer giving input to a single neuron in the second layer. The neuron in the second layer could, for example, implement an AND function.
Capabilities of Threshold Neurons

What kind of function can such a network realize?

Capabilities of Threshold Neurons

Assume that the dotted lines in the diagram represent the input-dividing lines implemented by the neurons in the first layer:

Then, for example, the second-layer neuron could output 1 if the input is within a polygon, and 0 otherwise.

Capabilities of Threshold Neurons

Assume that the polygons in the diagram indicate the input regions for which each of the second-layer neurons yields output 1:

Then, for example, the third-layer neuron could output 1 if the input is within any of the polygons, and 0 otherwise.

Capabilities of Threshold Neurons

However, we still may want to implement functions that are more complex than that.
An obvious idea is to extend our network even further.
Let us build a network that has three layers, with arbitrary numbers of neurons in the first and second layers and one neuron in the third layer.
The first and second layers are completely connected, that is, each neuron in the first layer sends its output to every neuron in the second layer.

What type of function can a three-layer network realize?

The more neurons there are in the first layer, the more vertices can the polygons have.
With a sufficient number of first-layer neurons, the polygons can approximate any given shape.
The more neurons there are in the second layer, the more of these polygons can be combined to form the output function of the network.
With a sufficient number of neurons and appropriate weight vectors \( \mathbf{w} \), a three-layer network of threshold neurons can realize any (I) function \( \mathbb{R}^n \rightarrow \{0, 1\} \).