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 \textit{linearly separable} function $\mathbb{R}^n \rightarrow \{0, 1\}$.

Although we only looked at two-dimensional input, our findings apply to any \textit{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 \textbf{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?

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 \textit{polygon}, 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 \textbf{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 \textbf{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?
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

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 $w_i$, a three-layer network of threshold neurons can realize any (!) function $\mathbb{R}^n \to \{0, 1\}$.

Terminology

Usually, we draw neural networks in such a way that the input enters at the bottom and the output is generated at the top.
Arrows indicate the direction of data flow.
The first layer, termed input layer, just contains the input vector and does not perform any computations.
The second layer, termed hidden layer, receives input from the input layer and sends its output to the output layer.
After applying their activation function, the neurons in the output layer contain the output vector.

Terminology

Example: Network function $f: \mathbb{R}^3 \to \{0, 1\}^2$

Linear Neurons

Obviously, the fact that threshold units can only output the values 0 and 1 restricts their applicability to certain problems.
We can overcome this limitation by eliminating the threshold and simply turning $f_i$ into the identity function so that we get:

$$o_i(t) = \text{net}_i(t)$$

With this kind of neuron, we can build networks with $m$ input neurons and $n$ output neurons that compute a function $f: \mathbb{R}^m \to \mathbb{R}^n$.

Linear Neurons

Linear neurons are quite popular and useful for applications such as interpolation.
However, they have a serious limitation: Each neuron computes a linear function, and therefore the overall network function $f: \mathbb{R}^m \to \mathbb{R}^n$ is also linear.
This means that if an input vector $x$ results in an output vector $y$, then for any factor $\phi$ the input $\phi x$ will result in the output $\phi y$.
Obviously, many interesting functions cannot be realized by networks of linear neurons.
Gaussian Neurons

Another type of neurons overcomes this problem by using a Gaussian activation function:

\[ f_i(\text{net}(t)) = e^{\frac{\text{net}(t) - 1}{\sigma^2}} \]

The drawback of Gaussian neurons is that we have to make sure that their net input does not exceed 1. This adds some difficulty to the learning in Gaussian networks.

Gaussian Neurons

Gaussian neurons are able to realize non-linear functions. Therefore, networks of Gaussian units are in principle unrestricted with regard to the functions that they can realize.

The drawback of Gaussian neurons is that we have to make sure that their net input does not exceed 1.

Sigmoidal Neurons

Sigmoidal neurons accept any vectors of real numbers as input, and they output a real number between 0 and 1.

Sigmoidal neurons are the most common type of artificial neuron, especially in learning networks.

A network of sigmoidal units with m input neurons and n output neurons realizes a network function \( f : \mathbb{R}^m \rightarrow (0,1)^n \)

The parameter \( \tau \) controls the slope of the sigmoid function, while the parameter \( \theta \) controls the horizontal offset of the function in a way similar to the threshold neurons.