Artificial neural networks are modeled after biological neural networks and how they work.

The dendrites of a neuron receive input signals from neighbors (or sensors, such as eyes and ears). If the potential build-up on the cell body from the dendrite inputs becomes sufficient, the neuron will "fire" sending a depolarizing electrical impulse down the axon. The axon, which can connect to as many 10000 other nerve cells, can then contribute to the electrical inputs of many other neurons. This cascading can continue on for many layers of neurons.

The fundamental models for artificial neural networks are based on this process, (More advanced models have since been developed.)
\[ \phi_j = \text{Activation Value} \]

\[ f(\phi_j) = \text{Activation Function} \leftarrow \text{Several Types Used} \]

**Linear:** \[ f(x) = \alpha x \]

**Step:** \[ f(x) = \begin{cases} \beta & \text{if } x \geq \Theta \\ -\beta & \text{if } x < \Theta \end{cases} \quad \Theta = \text{Threshold Value} \]

**Ramp:** (Symmetric Saturating Linear)

\[ f(x) = \begin{cases} \gamma & \text{if } x \geq \gamma \\ x & \text{if } |x| < \gamma \\ -\gamma & \text{if } x \leq -\gamma \end{cases} \]

**Sigmoid:**

\[ f(x) = \begin{cases} \frac{1}{1 + e^{-\alpha x}} & (\text{Unipolar}) \\ 2 \left( \frac{1}{1 + e^{-\alpha x}} \right) - 1 & (\text{Bipolar}) \end{cases} \]

with \( \alpha > 0 \)

\[ f(x) \rightarrow 0 \text{ to } 1 \quad f(x) \rightarrow -1 \text{ to } +1 \]

**Gaussian:** \[ f(x) = e^{-\frac{x^2}{2\nu}} \quad \text{with } \nu > 0 \]
These individual neurons are then connected together in various arrangements. For instance, a fully-connected, feedforward network:

\[ \phi_j = \sum_{i=0}^{n} x_i w_i \]

**Perceptron - A simple neuron**

(Bias) \( x_0 = 1 \)

\[ f(\phi_j) = \begin{cases} +1 & \text{if } \phi_j > 0 \\ -1 & \text{if } \phi_j \leq 0 \end{cases} \]

The threshold of the neuron is 0.
Let’s look at a small example with no bias.

\[ \phi_j = \sum_{i=1}^{z} x_i w_i \]

\[ f(\phi_j) \rightarrow y_j \]

So the output is a function of \( \phi_j = x_1 w_1 + x_2 w_2 \)

The decision of the perception is based on this, so the line \( x_1 w_1 + x_2 w_2 = 0 \) represents a "Decision Boundary".

For example, let \( w_1 = +1 \) and \( w_2 = -1 \)

Then the decision line is \( x_1 - x_2 = 0 \)

\[ \begin{array}{|c|c|c|}
   \hline
   x_1 & x_2 & \phi_j & f(\phi_j) \\
   \hline
   1 & 3 & -2 & -1 \\
   5 & 2 & +3 & +1 \\
   -3 & -2 & -1 & -1 \\
   4 & -6 & +10 & +1 \\
   \hline
\end{array} \]

* So the perception is a linear discriminator.

For more inputs and weights the decision boundary becomes a hyperplane (multidimensional), but it is still linear.
\[ x_0w_0 + x_1w_1 + x_2w_2 + \ldots + x_nw_n = 0 \]

So each perceptron divides the input space into two regions. These two regions can be used to categorize/classify the input space/coordinates.

Now consider two perceptrons (with no bias)

\[ \phi_1 = \sum_{i=1}^{2} x_i w_{i1} f(\phi_i) \rightarrow y_1 \]

\[ \phi_2 = \sum_{i=1}^{2} x_i w_{i2} f(\phi_i) \rightarrow y_2 \]

\[ w_{ij} \text{ weight from } i \text{ to } j \]

Let \[ w_{11} = +1 \]
\[ w_{21} = -1 \]
\[ w_{12} = -1 \]
\[ w_{22} = +3 \]

Then \( y_1 \) has decision boundary \( x_1 - x_2 = 0 \)
\( y_2 \) has decision boundary \( -x_1 + 3x_2 = 0 \)
So the output combinations of $y_1 \& y_2$ form four distinct regions.

For

<table>
<thead>
<tr>
<th>$y_1$</th>
<th>$y_2$</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>III</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>IV</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>II</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>I</td>
</tr>
</tbody>
</table>

For example $X_1 = 0$ $X_2 = 2$

Then $y_1 = -1$ $y_2 = +1$

Class or Category IV

So if we look at a layer of perceptions we have

\[ \Phi_1 = x_1w_{11} + x_2w_{21} \Rightarrow \Phi_1[n] = \overrightarrow{w}_1[n]\overrightarrow{x}[n] \]

\[ \Phi_2 = x_1w_{12} + x_2w_{22} \]

and $y_1 = f(\Phi_1)$

$y_2 = f(\Phi_2)$
If \( f(\phi_j) \) is linear then

\[
\begin{align*}
    y_1 &= x_1 w_{11} + x_2 w_{21} \\
    y_2 &= x_1 w_{12} + x_2 w_{22}
\end{align*}
\]

\[
\begin{bmatrix}
    y_1[n] \\
    y_2[n]
\end{bmatrix} =
\begin{bmatrix}
    w_{11} & w_{21} \\
    w_{12} & w_{22}
\end{bmatrix}
\begin{bmatrix}
    x_1[n] \\
    x_2[n]
\end{bmatrix}
\]

\[
\vec{Y}[n] = \vec{W}[n] \vec{X}[n]
\]

In general

\[
\vec{Y}[n] = \vec{W}[n] \vec{X}[n] + \vec{B}[n]
\]

This \( \phi_j \) format is similar to what shows up in many signal processing applications.

Discrete Convolution \( \Rightarrow f_1[k] * f_2[k] = \sum_{m=0}^{K} f_1[m] f_2[k-m] \)

DFT/FFT \( \Rightarrow X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \) with \( W_N = e^{-j\frac{2\pi}{N}} \)

A layer of perceptrons can then create several regions on a plane (or hyperplane) by using the linear discriminator property.

However there is a simple problem that cannot be solved by this arrangement.
The XOR Problem

No single line can separate the two categories.

More than one perceptron in a layer can split the region into multiple groups, but then we need to combine groups into categories.

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>

Logic "0" (False)
Logic "1" (True)
Logic "1" (True)
Logic "0" (False)