Neural networks

Artificial neural networks (ANNs) are a family of models inspired by biological neural networks. They are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown.

**Neuron**: A neuron is an information-processing unit that is fundamental to the operation of a neural network, which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

We identify the three basic elements of the neural model:
1. A set of synapses, or connecting links. Specifically a signal $x_j$ at the input of synapse $j$ connected to neuron $k$ is multiplied by the synaptic weight $w_{kj}$, where $j$ denotes the input signal label and $k$ denotes the neuron label.

2. An adder for summing the weighted input signals, together with a bias term $w_{k0}$ (also $b_k$).

3. An activation function for limiting the amplitude of the output of a neuron.

The bias has the effect of increasing or lowering the net input of the activation function.
We may formulate the neuron model in mathematical terms as

\[ v_k = \sum_{j=0}^{m} w_{kj} x_j \]

\[ y_k = \phi(v_k) \]

The activation function, denoted by \( \phi(v) \), defined the output of a neuron in terms of the induced local field \( v \), we identify two basic types of activation functions:
1. Threshold function

\[ \varphi(v) = \begin{cases} 
1 & \text{if } v \geq 0 \\
0 & \text{if } v < 0 
\end{cases} \quad (1) \]

2. Sigmoid function

\[ \varphi(v) = \frac{1}{1 + \exp(-av)} \quad (2) \]

where \( a \) is the sloping parameter of the sigmoid function.

The above functions range from 0 to 1. It is sometimes desirable to have activation function from -1 to 1. \text{sign(.)} function and \text{tanh(.)} are also commonly used.
Multilayer perceptron (MLP) neural networks:

Feedforward networks are arranged in layers, with the first layer taking the inputs and last layer producing outputs. The middle layers are called hidden layers. There is no signal from the output feedback to the input.

In the simplest form, we have an input layer that projects directly to the output layer, then it's called Single layer feedforward neural networks. The capabilities of a single layer perceptron are limited to linear decision boundaries.

Increasing the number of layers of a neural network to a large number is called deep neural networks (DNN).
The mathematical form for a two layer MLP is

\[ y_k(x, w) = \sigma \left( \sum_{j=0}^{M} w_{kj}^{(2)} h \left( \sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right) \]

where \( x \in \mathbb{R}^D \). The superscript \((1)\) and \((2)\) indicate the corresponding weights are in first or second layer. \( h(.) \) and \( \sigma(.) \) are chosen activation functions.
Example: Using two layer MLP for XOR problem. The XOR problem is highly nonlinear which cannot be solved by a single layer perceptron.

<table>
<thead>
<tr>
<th>Input vector $x_n$</th>
<th>Desired response $t_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>0</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>1</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>1</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>0</td>
</tr>
</tbody>
</table>

We use two hidden nodes, and the activation function is the threshold function.

$$y(x, w) = \varphi \left( \sum_{j=0}^{2} w^{(2)}_j \varphi \left( \sum_{i=0}^{2} w^{(1)}_{ji} x_i \right) \right)$$

For the hidden layer, the weights are

$$w^{(1)}_{10} = -1.5, \quad w^{(1)}_{11} = w^{(1)}_{12} = 1 \quad (3)$$

for the first node, and

$$w^{(1)}_{20} = -0.5, \quad w^{(1)}_{21} = w^{(1)}_{22} = 1 \quad (4)$$

for the second node.
For the output layer, the weights are

$$w_0^{(2)} = -0.5, \ w_1^{(2)} = -2, \ w_2^{(2)} = 1 \quad (5)$$

- For the input $(0, 0)$, the first hidden node output $v(-1.5 \times (1) + 1 \times (0) + 1 \times (0)) = 0$
  the second hidden outputs $v(-0.5 \times (1) + 1 \times (0) + 1 \times (0)) = 0$. The output node calculates $v(-0.5 - 2 \times (0) + 1 \times (0)) = 0$

- For the input $(0, 1)$, the first hidden node output $v(-1.5 \times (1) + 1 \times (0) + 1 \times (1)) = 0$
  the second hidden outputs $v(-0.5 \times (1) + 1 \times (0) + 1 \times (1)) = 1$. The output node calculates $v(-0.5 - 2 \times (0) + 1 \times (1)) = 1$
For the input $(1, 0)$, the first hidden node output $v(-1.5 \times 1 + 1 \times 1 + 1 \times 0) = 0$ the second hidden outputs $v(-0.5 \times 1 + 1 \times 1 + 1 \times 0) = 1$. The output node calculates $v(-0.5 - 2 \times 0 + 1 \times 1) = 1$

For the input $(1, 1)$, the first hidden node output $v(-1.5 \times 1 + 1 \times 1 + 1 \times 1) = 1$ the second hidden outputs $v(-0.5 \times 1 + 1 \times 1 + 1 \times 1) = 1$. The output node calculates $v(-0.5 - 2 \times 1 + 1 \times 1) = 0$
Recurrent neural networks: Recurrent neural networks distinguishes itself with feedforward neural networks in that it has at least one feedback loop, which means that the output of a neuron is fed back as its own input.