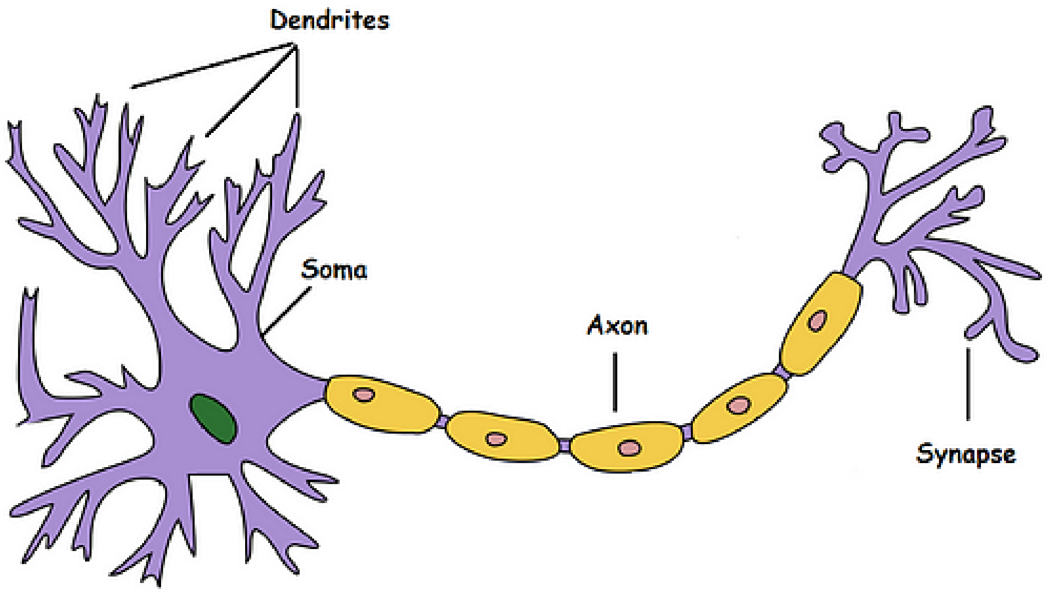
Fundamentals of Neural Network

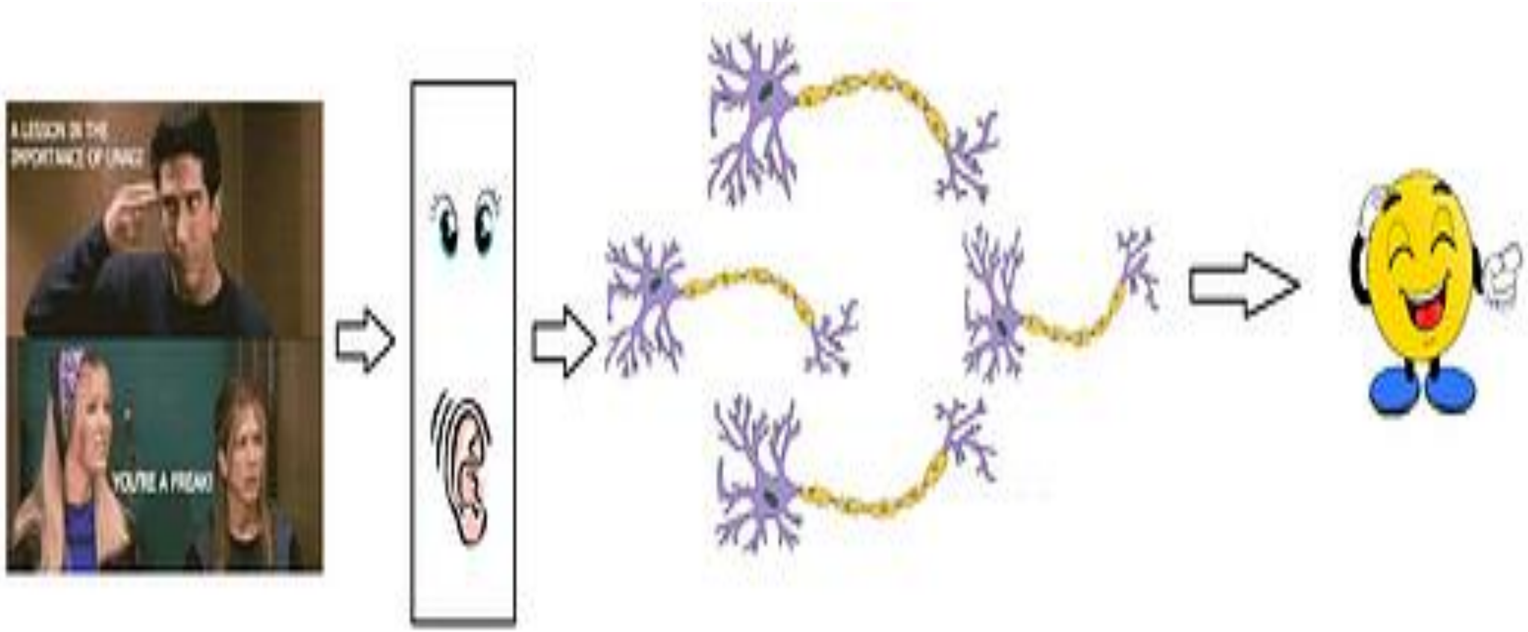
Dr. Kavita P Shirsat

Model Of A Biological Neuron

Dendrite: Receives signals from other neurons
Soma: Processes the information
Axon: Transmits the output of this neuron
Synapse: Point of connection to other neurons



Model Of A Biological Neuron



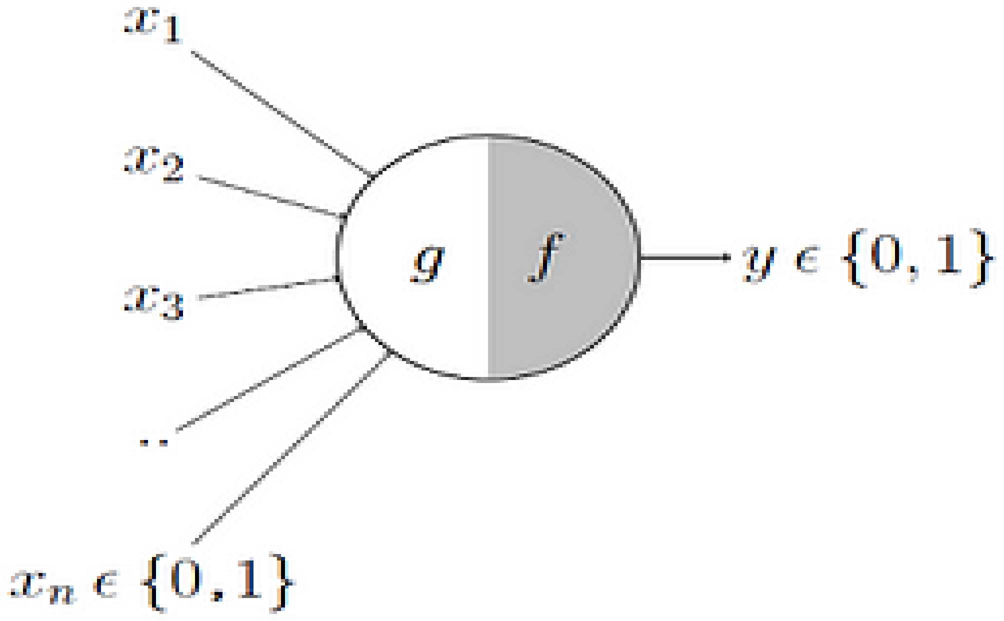
McCulloch-Pitts Neuron

It may be divided into 2 parts.

- The first part, ***g*** takes an input (ahem dendrite ahem), performs an aggregation and
- based on the aggregated value the second part, ***f*** makes a decision.

$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$



Threshold (θ\thetaθ):

•A fixed value that the weighted sum must exceed for the neuron to "fire" or activate

McCulloch-Pitts Neuron

- Input: $x1=1, x2=1$
- Weights: $w1=0.7$ (excitatory), $w2=-0.5$ (inhibitory)
- Threshold (θ): 0.2
- The net input is:
- Net Input $=(x1\cdot w1)+(x2\cdot w2)=(1\cdot 0.7)+(1\cdot -0.5)=0.7$
- If the threshold is 0.2, the neuron just meets the threshold and "fires."

McCulloch-Pitts Neuron

The model is inspired by biological neurons:

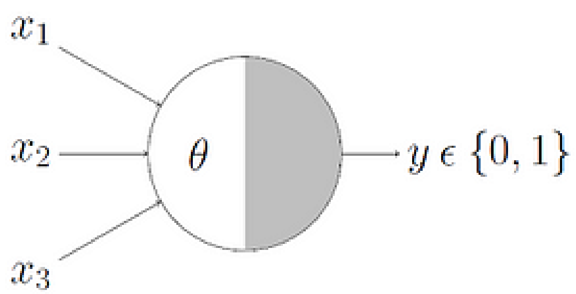
1. **Inputs**: Represent dendrites receiving signals.
2. **Weights**: Correspond to the strength of synaptic connections.
3. **Summation**: Mimics how a neuron integrates inputs.
4. **Threshold**: Reflects the firing threshold of a neuron.
5. **Output**: Indicates whether the neuron fires (sends a signal via its axon).

Artificial Neurons (McCulloch-Pitts or Modern Models)

In artificial neural networks, excitatory and inhibitory signals are modeled using **weights**:

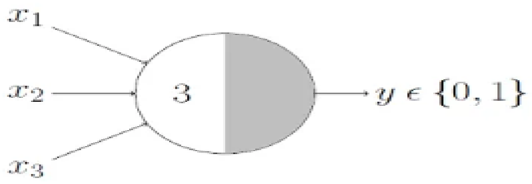
1. **Excitatory Weights**:
    1. Positive weights ($w>0$) amplify the input signal, increasing the likelihood of neuron activation.
    2. Example: If an input is 1 and the weight is +0.5, the contribution to the summation is +0.5, which adds positively to the net input.
2. **Inhibitory Weights**:
    1. Negative weights ($w<0$) suppress the input signal, reducing the likelihood of neuron activation.
    2. Example: If an input is 1 and the weight is −0.5, the contribution to the summation is −0.5, which subtracts from the net input.

Boolean Functions Using M-P Neuron



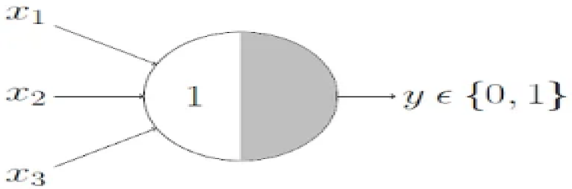
This representation just denotes that, for the boolean inputs ***x_1***, ***x_2*** and ***x_3*** if the ***g*(x)** i.e., **sum ≥ theta**, the neuron will fire otherwise, it won't.

AND Function

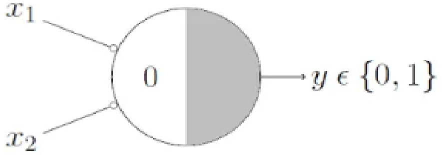


An AND function neuron would only fire when ALL the inputs are ON i.e., g(x) ≥ 3 here.

OR Function

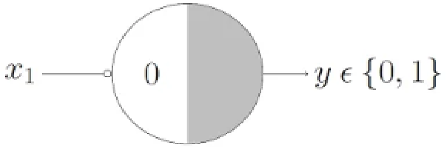


NOR Function



For a NOR neuron to fire, we want ALL the inputs to be 0 so the thresholding parameter should also be 0 and we take them all as inhibitory input.

NOT Function



NOT and NOR Gate

1. **NOT Gate**

Logic: The output is 1 if the input is 0; otherwise, it is 0.

- **Inputs:** x_1
- **Threshold:** $\theta = 0.5$
- **Net input:** $\text{net} = x_1$
- **Output:**

$$y = \begin{cases} 1 & \text{if net} < \theta \\ 0 & \text{if net} \geq \theta \end{cases}$$

x_1	net	y
0	0	1
1	1	0

---

2. **NOR Gate**

Logic: The output is 1 if both inputs are 0; otherwise, it is 0. NOR is the negation of OR.

- **Inputs:** x_1, x_2
- **Threshold:** $\theta = 1$
- **Net input:** $\text{net} = x_1 + x_2$
- **Output:**

$$y = \begin{cases} 1 & \text{if net} < \theta \\ 0 & \text{if net} \geq \theta \end{cases}$$

x_1	x_2	net	y
0	0	$0 + 0 = 0$	1
0	1	$0 + 1 = 1$	0
1	0	$1 + 0 = 1$	0
1	1	$1 + 1 = 2$	0

Nand Gate

3. NAND Gate

Logic: The output is 1 if at least one input is 0; otherwise, it is 0. NAND is the negation of AND.

- **Inputs:** x_1, x_2
- **Threshold:** $\theta = 2$
- **Net input:** $\mathrm{net} = x_1 + x_2$
- **Output:**

$$y = \begin{cases} 1 & \text{if net} < \theta \\ 0 & \text{if net} \geq \theta \end{cases}$$

x_1	x_2	net	y
0	0	$0+0=0$	1
0	1	$0+1=1$	1
1	0	$1+0=1$	1
1	1	$1+1=2$	0

Perceptron

- **Definition of Perceptron**
- The **Perceptron** is a type of artificial neuron introduced by Frank Rosenblatt in 1958. It is a simple supervised learning algorithm for binary classification tasks. A perceptron takes multiple input values, computes their weighted sum, applies an activation function (typically a step function), and produces an output.

Mathematical Representation

A perceptron computes the following:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Where:

- x_i: Inputs to the perceptron.
- w_i: Weights associated with each input.
- b: Bias term.
- f: Activation function (commonly a step function).
- y: Output of the perceptron (binary, e.g., 0 or 1).

The **step function** is defined as:

$$f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Working of a perceptron

Working of a Perceptron

1. **Initialization:** Assign small random values to weights (w_i) and bias (b).
2. **Input:** Receive input data $x_1, x_2, \ldots, x_n$.
3. **Weighted Sum:** Compute the weighted sum $z = \sum w_i x_i + b$.
4. **Activation Function:** Apply the step function to decide the output.
5. **Learning Rule:** If the output is incorrect, adjust weights and bias using the **Perceptron Learning Rule**:

$$\Delta w_i = \eta \cdot (t - y) \cdot x_i$$

$$\Delta b = \eta \cdot (t - y)$$

- t: Target output.
- y: Predicted output.
- η: Learning rate.

Example of AND Gate

Given inputs and target outputs for an AND gate:

- Inputs (x_1, x_2): $(0,0), (0,1), (1,0), (1,1)$
- Target outputs (t): $0, 0, 0, 1$

Step 1: Initialize Weights and Bias

- Start with $w_1 = 0.5, w_2 = 0.5, b = -0.7$.

Step 2: Compute Output for Each Input

$$z = w_1 x_1 + w_2 x_2 + b$$

Apply the step function to determine y.

x_1	x_2	z	Output y	Target t	Correct?
0	0	$(0.5 \cdot 0) + (0.5 \cdot 0) - 0.7 = -0.7$	0	0	Yes
0	1	$(0.5 \cdot 0) + (0.5 \cdot 1) - 0.7 = -0.2$	0	0	Yes
1	0	$(0.5 \cdot 1) + (0.5 \cdot 0) - 0.7 = -0.2$	0	0	Yes
1	1	$(0.5 \cdot 1) + (0.5 \cdot 1) - 0.7 = 0.3$	1	1	Yes

Weights are correct; no updates needed.

Learning with Weight Adjustments

Suppose the perceptron misclassifies a sample. Update the weights.

Input: $(x_1, x_2) = (1, 0), t = 1$

- Current $w_1 = 0.5, w_2 = 0.5, b = -0.7$.
- Compute $z = (0.5 \cdot 1) + (0.5 \cdot 0) - 0.7 = -0.2$.
- Step function gives $y = 0$ (incorrect).

Update Weights and Bias:

$$\Delta w_1 = \eta \cdot (t - y) \cdot x_1 = 0.1 \cdot (1 - 0) \cdot 1 = 0.1$$

$$\Delta w_2 = \eta \cdot (t - y) \cdot x_2 = 0.1 \cdot (1 - 0) \cdot 0 = 0$$

$$\Delta b = \eta \cdot (t - y) = 0.1 \cdot (1 - 0) = 0.1$$

Updated weights and bias:

$$w_1 = 0.5 + 0.1 = 0.6, \quad w_2 = 0.5 + 0 = 0.5, \quad b = -0.7 + 0.1 = -0.6$$

Learning with Weight Adjustments

Substitute the values:

$z = (0.6 * 1) + (0.5 * 0) + (-0.6)$

$z = 0.6 - 0.6 = 0$

Step 2: Apply the Step Function

The step function determines the perceptron output (y):

$f(z) = 1$ if $z >= 0$

0 if $z < 0$

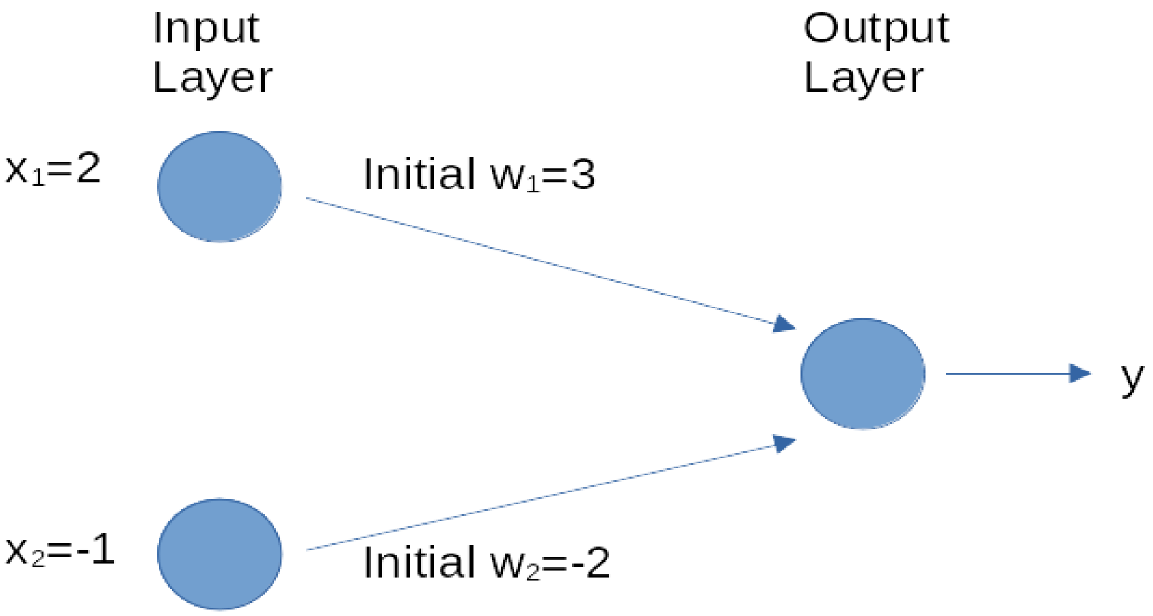
Since $z = 0$, the output is:

$y = 1$

Step 3: Compare with the Target Output

Predicted output (y): 1 - Target output (t): 1

Learning with Weight Adjustments Solve



the target output y=0

$$g'(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases}$$

Solution

We assume the task is a binary classification problem involving two features and that the neural network uses the rectified linear (ReLU) unit as activation function and has zero bias. In addition, the learning rate is 0.01 and the current weight (which is a vector) is $(3, -2)$ while the next training batch consists of just one input-output pair of $x = (2, -1)$ and the target output $y = 0$. Running the neural network on sample x gives a predicted output $y = 1$. The network weights will be updated as follows:
Mathematically, ReLU is defined as $g(x) = max(x, 0)$ and its derivative

$$g'(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta w_{ij} = 0.01 * (0 - 1) * (1, 0).\,(2, -1) = (-0.02, 0)$$

New weights = $(3, -2) - (-0.02, 0) = (2.98, -2)$.