**RAW CODE WITHOUT USING LIBRARIES**

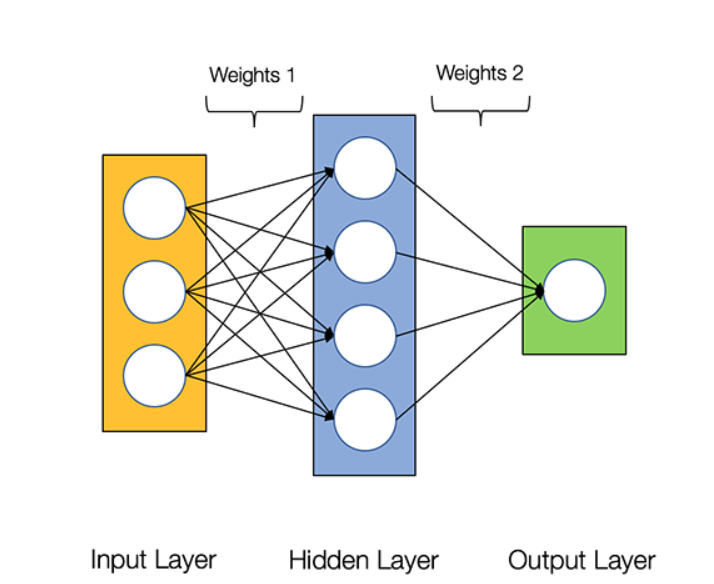
(CLASSIFICATION ALGORITHMS IN ML)

1 - **NEURAL NETWORKS**

**Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of** [**machine learning**](https://www.ibm.com/in-en/cloud/learn/machine-learning) **and are at the heart of** [**deep learning**](https://www.ibm.com/cloud/learn/deep-learning) **algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.**

**IN[1]:**

**> import numpy as np**

****

**IN[2]:**

***> #Helper Functions***

**> import numpy as np**

**> def sigmoid(x):**

**return 1 / (1 + np.exp(-x))**

***> # derivative of sigmoid***

***> # sigmoid(y) \* (1.0 - sigmoid(y))***

***> # the way we use this y is already sigmoided***

**> def sigmoid\_derivative(y):**

**return y \* (1.0 - y)**

**IN[3]:**

**> class NeuralNetwork:**

**> def \_\_init\_\_(self, x, y):**

**> self.input = x**

**> self.weights1 = np.random.rand(self.input.shape[1],4)**

**> self.weights2 = np.random.rand(4,1)**

**> self.y = y**

**> self.output = np.zeros(self.y.shape)**

**> def feedforward(self):**

**> 1)) self.layer1 = sigmoid(np.dot(self.input, self.weights1))**

**> 2)) self.output = sigmoid(np.dot(self.layer1, self.weights2))**

**> def backprop(self):**

***# application of the chain rule to find derivative of the loss #function with respect to weights2 and weights1***

**>d\_weights2 = np.dot(self.layer1.T, (2\*(self.y - self.output) \* sigmoid\_derivative(self.output)))**

**>d\_weights1 = np.dot(self.input.T, (np.dot(2\*(self.y - self.output) \* sigmoid\_derivative(self.output), self.weights2.T) \* sigmoid\_derivative(self.layer1)))**

***# update the weights with the derivative (slope) of the loss function***

**self.weights1 += d\_weights1**

**self.weights2 += d\_weights2**

**In [4]:**

**> X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])**

**> y = np.array([[0],[1],[1],[0]])**

**> nn = NeuralNetwork(X,y)**

**In [5]:**

***#epochs***

**> for i in range(1000):**

**> nn.feedforward()**

**> nn.backprop()**

**In [6]:**

**> nn.weights1**

**OUT[6]:**

**array([[-3.1052015 , 2.7879215 , 4.94178325, 6.45641537],**

**[ 6.29991156, 2.6325145 , 4.76683574, -3.80251242],**

**[ 0.43899981, 1.17461543, -0.62296007, 1.05909485]])**

**In [7]:**

**> nn.weights2**

**Out[7]:**

**> array([[-7.4076306 ],**

**[ 3.96851932],**

**[ 7.40185544],**

**[-7.41474627]])**

**In [8]:**

**> for i in range(y.shape[0]):**

**> print(nn.output[i],y[i])**

**[0.01221399] [0]**

**[0.96513377] [1]**

**[0.96464096] [1]**

**[0.04350959] [0]**