Aim: Write a Python program using Perceptron Neural Network to recognize even and odd numbers. Given numbers are in ASCII form from 0 to 9.

Software for Python: Jupyter Notebook

Theory:

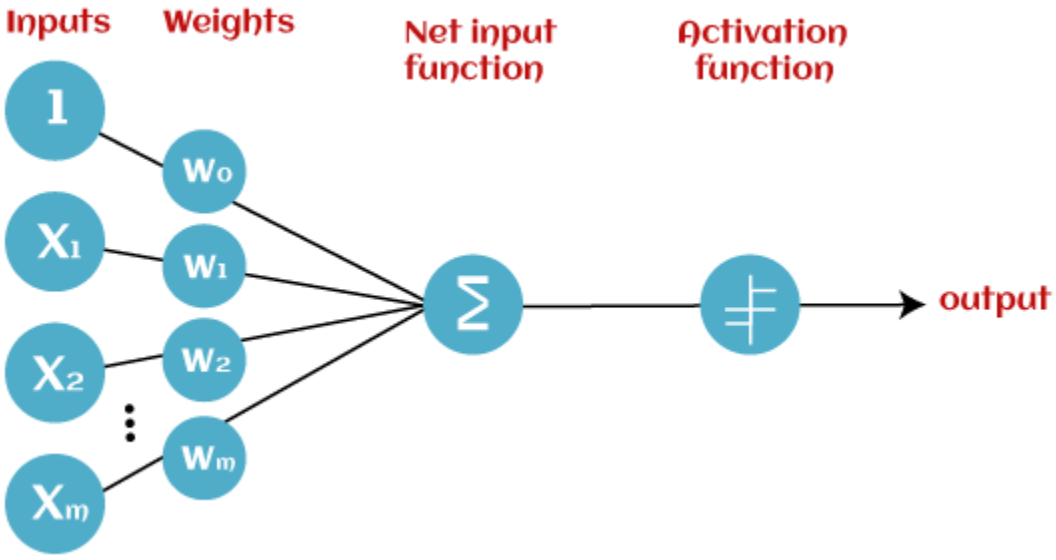
Perceptron Neural Network

Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks. Further, Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence.

Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks. However, it is a supervised learning algorithm of binary classifiers. Hence, we can consider it as a single-layer neural network with four main parameters, i.e., input values, weights and Bias, net sum, and an activation function.

Basic Components of Perceptron

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components. These are as follows:



Input Nodes or Input Layer:

This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

Weight and Bias:

Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the

associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

Activation Function:

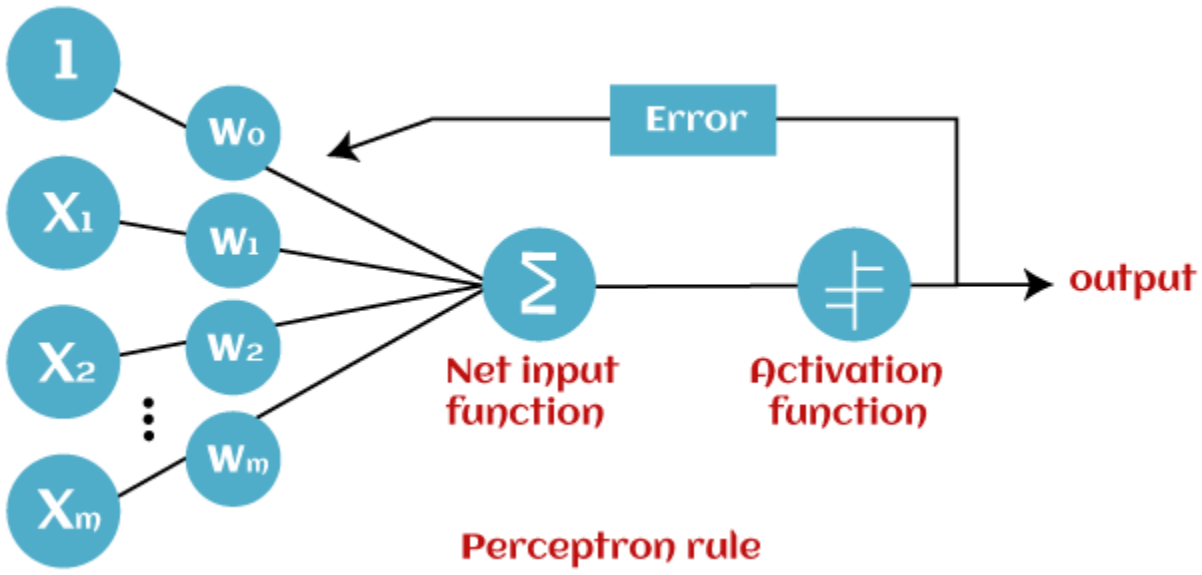
These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

Types of Activation functions:

1. Sign function
2. Step function, and
3. Sigmoid function

How does Perceptron work?

In Machine Learning, Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function. The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum. Then this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the step function and is represented by 'f'.



This step function or Activation function plays a vital role in ensuring that output is mapped between required values (0,1) or (-1,1). It is important to note that the weight of input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.

Perceptron model works in two important steps as follows:

Step-1

In the first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:

$wi*xi = x1*w1 + x2*w2 +…wn*xn$

Add a special term called bias 'b' to this weighted sum to improve the model's performance.

$\sum wi*xi + b$

Step-2

In the second step, an activation function is applied with the above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

$Y = f(\sum wi*xi + b)$

Procedure:

1. This code uses the Perceptron Neural Network to identify even and odd numbers.
2. The problem is solved by converting ASCII representation of numbers from 0 to 9 into binary representation and then training the Perceptron model on these inputs.
3. The sigmoid function calculates the sigmoid activation function for a given input. The sigmoid_derivative function calculates the derivative of the sigmoid function.
4. The predict function takes in the inputs and weights, computes the weighted sum, passes it through the sigmoid activation function and returns the output.
5. The train function trains the Perceptron model. It takes in the inputs, targets, initial weights, learning rate and the number of epochs as inputs.
6. It iterates over the number of epochs and updates the weights based on the error between the target and the output.
7. The even_or_odd function takes in a number and returns 0 if it is even and 1 if it is odd.
8. The ascii_to_binary function takes in a number and converts it into binary representation.
9. Finally, the inputs, targets, and weights are prepared.
10. The Perceptron model is trained using the train function and the final weights are obtained.
11. The Perceptron model is then tested on a number of test cases, and the result is printed indicating whether the ASCII representation of the number is even or odd.

Conclusion: Thus we have studied Perceptron Neural Network to recognize even and odd numbers using python programming.

Program code and Output

In [1]:
```
import numpy as np

def sigmoid(x):
  return 1.0 / (1 + np.exp(-x))

def sigmoid_derivative(x):
  return x * (1.0 - x)

def predict(inputs, weights):
  weighted_sum = np.dot(inputs, weights)
  output = sigmoid(weighted_sum)
  return output

def train(inputs, targets, weights, learning_rate, epochs):
  for i in range(epochs):
    outputs = []
    for j in range(len(inputs)):
      output = predict(inputs[j], weights)
      error = targets[j] - output
      weights += learning_rate * error * sigmoid_derivative(output) * inputs[j]
      outputs.append(output)
    print("Epoch: {}/{}".format(i+1, epochs))
    for j in range(len(inputs)):
      print("Input: {} Output: {} Target: {}".format(inputs[j], outputs[j], targets[j]))
  return weights

def even_or_odd(number):
  return 0 if int(number) % 2 == 0 else 1

def ascii_to_binary(number):
  binary = bin(int(number))[2:]
  binary = [int(digit) for digit in binary.zfill(7)]
  return binary

inputs = []
targets = []

for i in range(10):
  binary = ascii_to_binary(str(i))
  inputs.append(binary)
  targets.append(even_or_odd(i))

inputs = np.array(inputs)
targets = np.array(targets)
weights = np.random.rand(7)
```

```
Input: [0 0 0 0 0 1 0] Output: 0.374048650667305 Target: 0
Input: [0 0 0 0 0 1 1] Output: 0.8471099196871616 Target: 1
Input: [0 0 0 0 1 0 0] Output: 0.3391334988080453 Target: 0
Input: [0 0 0 0 1 0 1] Output: 0.8267801532326151 Target: 1
Input: [0 0 0 0 1 1 0] Output: 0.23255457327696114 Target: 0
Input: [0 0 0 0 1 1 1] Output: 0.7384587767589553 Target: 1
Input: [0 0 0 1 0 0 0] Output: 0.38748885380793996 Target: 0
Input: [0 0 0 1 0 0 1] Output: 0.8554733884361188 Target: 1
The ASCII representation of 0 is odd
The ASCII representation of 1 is odd
The ASCII representation of 2 is even
The ASCII representation of 3 is odd
The ASCII representation of 4 is even
The ASCII representation of 5 is odd
The ASCII representation of 6 is even
The ASCII representation of 7 is odd
The ASCII representation of 8 is even
The ASCII representation of 9 is odd
```