***Practical 8***

**Aim:** *Implement feed forward back propagation neural network learning algorithm for the restaurant waiting problem.*

***Theory:***

***What is Backpropagation?***

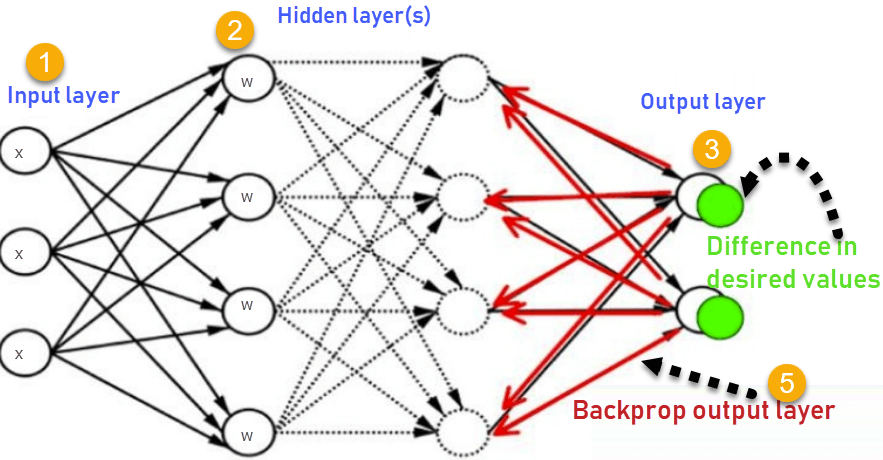
*Backpropagation is the essence of neural network training. It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization.*

*Backpropagation in neural network is a short form for “backward propagation of errors.” It is a standard method of training artificial neural networks. This method helps calculate the gradient of a loss function with respect to all the weights in the network.*

***How Backpropagation Algorithm Works***

*The Back propagation algorithm in neural network computes the gradient of the loss function for a single weight by the chain rule. It efficiently computes one layer at a time, unlike a native direct computation. It computes the gradient, but it does not define how the gradient is used. It generalizes the computation in the delta rule.*

*Consider the following Back propagation neural network example diagram to understand:*

******

1. *Inputs X, arrive through the preconnected path*
2. *Input is modelled using real weights W. The weights are usually randomly selected.*
3. *Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.*
4. *Calculate the error in the outputs.*
5. *Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.*

*Keep repeating the process until the desired output is achieved*

***Why We Need Backpropagation?***

*Most prominent advantages of Backpropagation are:*

* *Backpropagation is fast, simple and easy to program*
* *It has no parameters to tune apart from the numbers of input*
* *It is a flexible method as it does not require prior knowledge about the network*
* *It is a standard method that generally works well*
* *It does not need any special mention of the features of the function to be learned.*

***What is a Feed Forward Network?***

*A feedforward neural network is an artificial neural network where the nodes never form a cycle. This kind of neural network has an input layer, hidden layers, and an output layer. It is the first and simplest type of artificial neural network.*

***Types of Backpropagation Networks***

*Two Types of Backpropagation Networks are:*

* *Static Back-propagation*
* *Recurrent Backpropagation*

**Static back-propagation:**

It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.

**Recurrent Backpropagation:**

Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.

The main difference between both of these methods is: that the mapping is rapid in static back-propagation while it is non-static in recurrent backpropagation.

**Backpropagation Key Points**

* Simplifies the network structure by elements weighted links that have the least effect on the trained network
* You need to study a group of input and activation values to develop the relationship between the input and hidden unit layers.
* It helps to assess the impact that a given input variable has on a network output. The knowledge gained from this analysis should be represented in rules.
* Backpropagation is especially useful for deep neural networks working on error-prone projects, such as image or speech recognition.
* Backpropagation takes advantage of the chain and power rules allows backpropagation to function with any number of outputs.

**Best practice Backpropagation**

Backpropagation in neural network can be explained with the help of “Shoe Lace” analogy

**Too little tension =**

* Not enough constraining and very loose

**Too much tension =**

* Too much constraint (overtraining)
* Taking too much time (relatively slow process)
* Higher likelihood of breaking

**Pulling one lace more than other =**

* Discomfort (bias)

**Disadvantages of using Backpropagation**

* The actual performance of backpropagation on a specific problem is dependent on the input data.
* Back propagation algorithm in data mining can be quite sensitive to noisy data
* You need to use the matrix-based approach for backpropagation instead of mini-batch.

***Code:***

import numpy as np

class NeuralNetwork():

    def \_\_init\_\_(self):

        #seeding for random number generation

        np.random.seed()

        #converting weights to a 3 by 1 matrix

        self.synaptic\_weights=2\*np.random.random((3,1))-1

    #x is output variable

    def sigmoid(self, x):

        #applying the sigmoid function

        return 1/(1+np.exp(-x))

    def sigmoid\_derivative(self,x):

        #computing derivative to the sigmoid function

        return x\*(1-x)

    def train(self,training\_inputs,training\_outputs,training\_iterations):

        #training the model to make accurate predictions while adjusting

        for iteration in range(training\_iterations):

            #siphon the training data via the neuron

            output=self.think(training\_inputs)

            error=training\_outputs-output

            #performing weight adjustments

            adjustments=np.dot(training\_inputs.T,error\*self.sigmoid\_derivative(output))

            self.synaptic\_weights+=adjustments

    def think(self,inputs):

        #passing the inputs via the neuron to get output

        #converting values to floats

        inputs=inputs.astype(float)

        output=self.sigmoid(np.dot(inputs,self.synaptic\_weights))

        return output

if \_\_name\_\_=="\_\_main\_\_":

    #initializing the neuron class

    neural\_network=NeuralNetwork()

    print("Beginning randomly generated weights: ")

    print(neural\_network.synaptic\_weights)

    #training data consisting of 4 examples--3 inputs & 1 output

    training\_inputs=np.array([[0,0,1],[1,1,1],[1,0,1],[0,1,1]])

    training\_outputs=np.array([[0,1,1,0]]).T

    #training taking place

    neural\_network.train(training\_inputs,training\_outputs,15000)

    print("Ending weights after training: ")

    print(neural\_network.synaptic\_weights)

    user\_input\_one=str(input("User Input One: "))

    user\_input\_two=str(input("User Input Two: "))

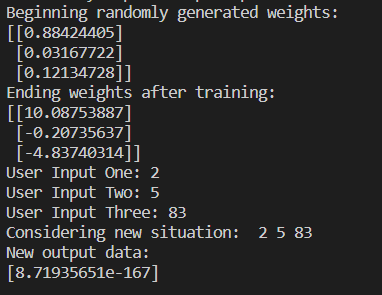
    user\_input\_three=str(input("User Input Three: "))

    print("Considering new situation: ",user\_input\_one,user\_input\_two,user\_input\_three)

    print("New output data: ")

    print(neural\_network.think(np.array([user\_input\_one,user\_input\_two,user\_input\_three])))

***Output:***

******

***Conclusion:***

*Implemented feed forward back propagation neural network learning algorithm for the restaurant waiting problem.*