Date 02/12/20

Expt. No. 04 Page No. 08

Milind -Bijukumar 4MT17CS058

* Build an Attribute Neural Network by implementing the Back propogation algorithm and test the same using appropriate dataset

* Program

```
from math import exp
 from random import seed
 from random import random.
def initialize network(n-input,n_hidden,n_outputs):
     network -list()
    hidden_layer = [{'weights':[random() for i in range
                      (n_inputs+1)]} for i in range (n_hidden)]
    network.append (hidden_layer)
    output_layer =[{'weights': [random() for i in range
                      (n_hidden+1)]} for i in range(n_outputs]
     network.append(output_layer)
     return network.
def activate(weights,inputs):
     activation = weights[-1]
     for i in range(len(weights) -1):
            activation +=weights[i] *inputs[i]
      return activation
def transfer(activation):
        return 1.0 / (1.0+exp (-activation))
def forward propogate(network,row):
        input -row
        for layer in network :
             new_input =[ ]
```

Teacher's Signature

Milind Bijukumar - 4MT17CS058

```
            for neuron in layer :
                  activation=activation(neuron [weights], inputs)
                  neuron['output'] = transfer(activation)
                  new_inputs.append(neuron['output'])
            inputs=new.inputs
    return inputs
def transfer derivative(output):
    return output *(1.0-output)
def backward_propagate_error(network, expected):
    for j in range(len(layer)):
        error=0.0
        for neuron in network[i+1]:
              error +=(neuron['weights'][i] * neuron['delta'])
        error append (error)
    else:
        for j in range (len(layer)):
              neuron =layer[i]
              errors =append (expected[j].neuron['output']
        for j in range (len(layer)):
            neuron =layer(j]
            neuron delta] =error[j] * transfer_derivative
                              (neuron['output'])
def update_weights(network, row, l_rate):
    for i in range(len(network)):
        input=row[:-1]
        if i!=0 :
            input=[neuron['output'] for neuron in network[i-1]]
        for neuron in network[i]:
            for j in range(len(input)):
```

```
            neuron ['weight'][j]+=l_rate *neuron ['delta']* inputs [j]
        neuron ['weight'][-1]+ = l-rate * neuron ['delta']
def train_network (network ,train, l-rate, n_epoch, n_outputs):
    for epoch in range (n_epouch):
        sum_error=0
        for row in train:
            output = forward_propogate (network ,row)
            expected [row [-1]] =1
            sum-error +=sum [expected [i].-output [i]) * *
                                  2 for i in range (len (expected))])
            backward_propogate_error (network, expected)
            update_weight (network, row, l_rate)
        print( '>epocn = %d, lrate = %.3f, error = %.3f %3(epoch, l_rate, sum
                                                            -error))
seed (1)
dataset =[
    [2.7810836, 2.550537003,0], [1.465489372, 2. 362125076,0],
    [3.396561688, 4.400293529,0], [1. 38807019, 1.850220317,0],
    [3.064072232, 3.005305973,0], [7.627531214, 2.7599262235,1]
    [5.332441248, 2.088626775, 1], [6.922596716, 1. 77106367, 2],
    [8.675418651, -0 .242068655,1] [7.673756466, 3.50856301,1]]
n_input =len(dataset [0]) -1
n_outputs=len(set([row[-1] for row in dataset ]])

network =initialize _network (n_input, 2, n-outputs)
train_network (network ,dataset 0.5, 20, n_outputs)
for layer in network:
    print(layer)
```

Teacher's Signature

* Output :=

```
>epoch =0   lrate =0.500 , error =6.350
>epoch =1   lrate =0.500   error = 5.531
>epoch= 2   lrate =0.500   error = 5.221
>epoch=3    lrate=0.500    error= 4.951
>epoch=4    lrate = 0.500  error= 4.519
>epoch=5    lrate = 0.500  error= 4.173
>epoch=6    lrate = 0.500  error = 3.835
>epoch=7    lrate = 0.500  error= 3.506
>epoch=8    lrate=0.500    error= 3.192
>epoch=9    lrate=0.500    error = 2.898
>epoch=10   lrate =0.500   error = 2.626
>epoch =11  lrate =0.500   error = 2.377
>epoch =12  lrate = 0.500  error = 2.153
>epoch =13  lrate =0.500   error = 1.953
>epoch =14  lrate= 0.500   error= 1.714
>epoch =15  lrate =0.500   error= 1.614
>epoch =16  lrate=0.500    error = 1.472
>epoch = 17 lrate= 0.500   error= 1.346
>epoch =18  lrate =0.500   error= 1.293
>epoch =19  lrate =0.500   error = 1.132

[{'weights' : [-1.4688375095432327, 1.850887325439514,
      1.08858178629550297], 'output': 0.029980305640185
      'delta': -0.005956604162325625 },
 {'weights' : [0.37711098142462157, -0.0625909804552087,
      0.2765123702642716], 'output' : 0.94562229002113
      'delta': 0.0026279652850863837 }]
```

```
[{'weights': [2.51139449397849, -0.3391929502445985,
        -0.9671565426390275], 'output': 0.236487942025
        7587, 'delta': 0.04270059278364587},

    ['weights': [-2.5584149848484263, 1.0036422106209202,
        0.42383086467582715], 'output': 0.77905352024
        3836, 'delta': 0.03803132596437354}]
```