AIM

- **Write a program in Python to implement Back-Proagation Neural Network**

In [14]:

```
import math
import random
import string
class NN:
    def __init__(self, NI, NH, NO):
        # number of nodes in layers
        self.ni = NI + 1 # +1 for bias
        self.nh = NH
        self.no = NO
        self.ai, self.ah, self.ao = [],[], []
        self.ai = [1.0]*self.ni
        self.ah = [1.0]*self.nh
        self.ao = [1.0]*self.no
        self.wi = makeMatrix (self.ni, self.nh)
        self.wo = makeMatrix (self.nh, self.no)

        # initialize node weights to random vals
        randomizeMatrix ( self.wi, -0.2, 0.2 )
        randomizeMatrix ( self.wo, -2.0, 2.0 )
        self.ci = makeMatrix (self.ni, self.nh)
        self.co = makeMatrix (self.nh, self.no)

    def runNN (self, inputs):
        if len(inputs) != self.ni-1:
            print('\nIncorrect number of inputs')
        for i in range(self.ni-1):
            self.ai[i] = inputs[i]
        for j in range(self.nh):
            sum = 0.0
            for i in range(self.ni):
                sum +=( self.ai[i] * self.wi[i][j] )
                self.ah[j] = sigmoid (sum)
        for k in range(self.no):
            sum = 0.0
            for j in range(self.nh):
                sum +=( self.ah[j] * self.wo[j][k] )
                self.ao[k] = sigmoid (sum)
        return self.ao

    def backPropagate (self, targets, N, M):
        output_deltas = [0.0] * self.no
        for k in range(self.no):
            error = targets[k] - self.ao[k]
            output_deltas[k] =  error * dsigmoid(self.ao[k])
        for j in range(self.nh):
            for k in range(self.no):
                change = output_deltas[k] * self.ah[j]
                self.wo[j][k] += N*change + M*self.co[j][k]
                self.co[j][k] = change
                hidden_deltas = [0.0] * self.nh
        for j in range(self.nh):
            error = 0.0
            for k in range(self.no):
                error += output_deltas[k] * self.wo[j][k]
                hidden_deltas[j] = error * dsigmoid(self.ah[j])
        for i in range (self.ni):
            for j in range (self.nh):
                change= hidden_deltas[j] * self.ai[i]
                self.wi[i][j] += N*change + M*self.ci[i][j]
                self.ci[i][j] = change
```

```
                error = 0.0
        for k in range(len(targets)):
            error = 0.5 * (targets[k]-self.ao[k])**2
        return error

    def weights(self):
        print('Input weights:')
        for i in range(self.ni):
            print (self.wi[i])
            print()
            print('Output weights:')
        for j in range(self.nh):
            print (self.wo[j])
            print ('')

    def test(self, patterns):
        for p in patterns:
            inputs = p[0]
            print('Inputs:', p[0], '-->', self.runNN(inputs), '\tTarget', p[1])

    def train (self, patterns, max_iterations = 1000, N=0.5, M=0.1):
        for i in range(max_iterations):
            for p in patterns:
                inputs = p[0]
                targets = p[1]
                self.runNN(inputs)
                error = self.backPropagate(targets, N, M)
            if i % 50 == 0:
                print('\nCombined error', error)
                self.test(patterns)

    def sigmoid (x):
        return math.tanh(x)
    def dsigmoid (y):
        return 1 - y**2

    def makeMatrix ( I, J, fill=0.0):
        m = []
        for i in range(I):
            m.append([fill]*J)
        return m

    def randomizeMatrix ( matrix, a, b):
        for i in range ( len (matrix) ):
            for j in range ( len (matrix[0]) ):
                matrix[i][j] = random.uniform(a,b)

    def main ():
        pat = [
              [[0,0], [1]],
              [[0,1], [1]],
              [[1,0], [1]],
              [[1,1], [0]]
          ]
        myNN = NN ( 2, 2, 1)
        myNN.train(pat)

if __name__ == "__main__":
    main()
```

```
Combined error 0.23844825433975494
Inputs: [0, 0] --> [0.4917183558666849]  Target [1]
Inputs: [0, 1] --> [0.5798753186357646]  Target [1]
Inputs: [1, 0] --> [0.5589327353707154]  Target [1]
Inputs: [1, 1] --> [0.6147745527956241]  Target [0]

Combined error 0.011136335692823098
Inputs: [0, 0] --> [0.9927591503087781]  Target [1]
Inputs: [0, 1] --> [0.9559145764030351]  Target [1]
Inputs: [1, 0] --> [0.9515119763516615]  Target [1]
Inputs: [1, 1] --> [0.20051318954918496]  Target [0]
```

```
Combined error 0.0021638918119382 74
Inputs: [0, 0] --> [0.9964957253179815]  Target [1]
Inputs: [0, 1] --> [0.9652692285748277]  Target [1]
Inputs: [1, 0] --> [0.9632754730923068]  Target [1]
Inputs: [1, 1] --> [0.042571971651788 78]  Target [0]

Combined error 0.0017299532369826197
Inputs: [0, 0] --> [0.997366500299601]  Target [1]
Inputs: [0, 1] --> [0.9725162546473564]  Target [1]
Inputs: [1, 0] --> [0.9717636898092352]  Target [1]
Inputs: [1, 1] --> [0.037696816764428945]  Target [0]

Combined error 0.0012738889081292128
Inputs: [0, 0] --> [0.9978163001814991]  Target [1]
Inputs: [0, 1] --> [0.9764552796069919]  Target [1]
Inputs: [1, 0] --> [0.9760490022048941]  Target [1]
Inputs: [1, 1] --> [0.03224823790870259]  Target [0]

Combined error 0.0009749414749159763
Inputs: [0, 0] --> [0.9981011803883064]  Target [1]
Inputs: [0, 1] --> [0.9791126763687251]  Target [1]
Inputs: [1, 0] --> [0.9788547923783184]  Target [1]
Inputs: [1, 1] --> [0.028273674378764577]  Target [0]

Combined error 0.0007899328948533726
Inputs: [0, 0] --> [0.9983011648817425]  Target [1]
Inputs: [0, 1] --> [0.9810558385942697]  Target [1]
Inputs: [1, 0] --> [0.9808759652368086]  Target [1]
Inputs: [1, 1] --> [0.025481512623331534]  Target [0]

Combined error 0.0006808660417503443
Inputs: [0, 0] --> [0.998451139863196]  Target [1]
Inputs: [0, 1] --> [0.982568581578017]  Target [1]
Inputs: [1, 0] --> [0.9824353117996251]  Target [1]
Inputs: [1, 1] --> [0.023667001659958713]  Target [0]

Combined error 0.0005975597957300863
Inputs: [0, 0] --> [0.9985688495387567]  Target [1]
Inputs: [0, 1] --> [0.983791615069146]  Target [1]
Inputs: [1, 0] --> [0.9836886145046992]  Target [1]
Inputs: [1, 1] --> [0.022221908476757996]  Target [0]

Combined error 0.000517968866385747
Inputs: [0, 0] --> [0.9986640363420732]  Target [1]
Inputs: [0, 1] --> [0.9847844733891827]  Target [1]
Inputs: [1, 0] --> [0.9847021716411649]  Target [1]
Inputs: [1, 1] --> [0.02073695481687303]  Target [0]

Combined error 0.0004667346850809813 7
Inputs: [0, 0] --> [0.9987430371742934]  Target [1]
Inputs: [0, 1] --> [0.985621614599378]  Target [1]
Inputs: [1, 0] --> [0.9855542362821736]  Target [1]
Inputs: [1, 1] --> [0.01969675086824124]  Target [0]

Combined error 0.0004219087272785436
Inputs: [0, 0] --> [0.9988100186854467]  Target [1]
Inputs: [0, 1] --> [0.9863438269910825]  Target [1]
Inputs: [1, 0] --> [0.9862876181456784]  Target [1]
Inputs: [1, 1] --> [0.018760339545514728]  Target [0]

Combined error 0.0003831727464173185
Inputs: [0, 0] --> [0.9988675685311446]  Target [1]
Inputs: [0, 1] --> [0.9869642839305436]  Target [1]
Inputs: [1, 0] --> [0.9869166145061561]  Target [1]
Inputs: [1, 1] --> [0.017893760558738574]  Target [0]

Combined error 0.0003539717753108405
Inputs: [0, 0] --> [0.9989178053125445]  Target [1]
Inputs: [0, 1] --> [0.9875156066532436]  Target [1]
Inputs: [1, 0] --> [0.9874746856429638]  Target [1]
Inputs: [1, 1] --> [0.017218133161965243]  Target [0]
```

```
Combined error 0.00032566038067847705
Inputs: [0, 0] --> [0.9989620433432006]  Target [1]
Inputs: [0, 1] --> [0.9880002332814282]  Target [1]
Inputs: [1, 0] --> [0.9879646997528563]  Target [1]
Inputs: [1, 1] --> [0.01652927252806915]  Target [0]

Combined error 0.0003041266897930005
Inputs: [0, 0] --> [0.9990014465968147]  Target [1]
Inputs: [0, 1] --> [0.9884380216389181]  Target [1]
Inputs: [1, 0] --> [0.9884069003780152]  Target [1]
Inputs: [1, 1] --> [0.01598801968752317]  Target [0]

Combined error 0.000283251539691603
Inputs: [0, 0] --> [0.9990367660460395]  Target [1]
Inputs: [0, 1] --> [0.9888298026541817]  Target [1]
Inputs: [1, 0] --> [0.9888023171200414]  Target [1]
Inputs: [1, 1] --> [0.015440131587628109]  Target [0]

Combined error 0.00026620599046863234
Inputs: [0, 0] --> [0.9990687028846492]  Target [1]
Inputs: [0, 1] --> [0.9891880284567527]  Target [1]
Inputs: [1, 0] --> [0.9891635987546132]  Target [1]
Inputs: [1, 1] --> [0.01498080789204278]  Target [0]

Combined error 0.00025065734664740376
Inputs: [0, 0] --> [0.9990977164260472]  Target [1]
Inputs: [0, 1] --> [0.9895132574496751]  Target [1]
Inputs: [1, 0] --> [0.9894914076286793]  Target [1]
Inputs: [1, 1] --> [0.014544271690338931]  Target [0]

Combined error 0.00023647081134114465
Inputs: [0, 0] --> [0.9991242540703081]  Target [1]
Inputs: [0, 1] --> [0.9898129131997737]  Target [1]
Inputs: [1, 0] --> [0.9897932729057921]  Target [1]
Inputs: [1, 1] --> [0.014137435272012618]  Target [0]
```

In [ ]: