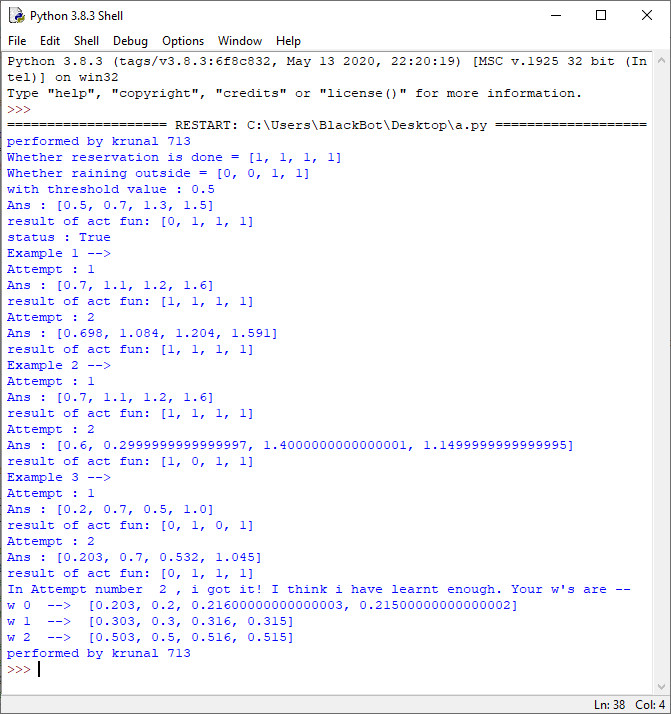
**Practical no 6**

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

**CODE:**

|  |
| --- |
| class Perceptron : # With 2 inputs and 1 output  def \_\_init\_\_(self, a,b, c, tval):  self.x = a # input vector  self.result = b # activation result  self.cresult = c # summation result  self.threshold = tval # threshold value used by activation function  self.w = []  def h(self, tw): # calculating summation(hypothesis function)  hresult= []  for i in range(0 , len(self.result)):  hresult.append(0)  #print("index - ", i, ";", hresult)  for j in range(0,len(tw)):  #print("i=",i, ",j=",j)  hresult[i] = hresult[i] + ( tw[j][i]\*self.x[j][i] )  return hresult  def checkthreshold(self, hresult): # applying activation function on summation result using threshold value  #flag = True  actfun =[]  for i in range(0 , len(self.result)) :  if (hresult[i] <= self.threshold ):  actfun.append(0)  else :  actfun.append( 1)  print("Ans :", hresult)  print("result of act fun:", actfun)  for i in range(0 , len(self.x)) :  if (actfun[i] != self.result[i]) :  return False  return True    def training(self, tw, alpha): #passing w vector and alpha value  i=1  while i<=2 : # Max 100 attempts  print("Attempt :", i)  hresult = self.h(tw)  if(self.checkthreshold(hresult)) : #if training result matches the test result  self.w = tw  print("In Attempt number ", i, ", i got it! I think i have learnt enough. Your w's are --" )  for x in range(0,len(self.w)):  print("w", x, " --> ", self.w[x])  break  i = i +1  # Changing values of w to reduce error/loss using batch gradient descent learning rule given on page 721 eqn 18.6  for j in range(0,len(self.result)) :  for k in range(0, len(tw)):  sum = 0  for n in range(0, len(tw)):  sum = sum + (self.cresult[j] - hresult[j]) \*self.x[n][j]  tw[k][j] = tw[k][j] + alpha\*sum  if(i>=100):  print("I am exhausted, tried 100 iterations! plz change something else...")  a = [ [1,1,1,1], [0,0,1,1] , [0,1,0,1] ] # x vector, x0 is dummy  b = [0,1,1,1] # result of activation function  c = [0.5, 0.7, 1.3, 1.5] # sample h values  print(“performed by krunal 713”)  p = Perceptron(a,b,c, 0.5) # threshold = 0.5  print("Whether reservation is done =", p.x[0])  print("Whether raining outside =", p.x[1])  print("with threshold value :", p.threshold)  r = p.h([ [0.5,0.5,0.5,0.5], [0.8, 0.8, 0.8, 0.8], [0.2, 0.2, 0.2, 0.2]])  print("status :", p.checkthreshold(r))  print("Example 1 -->") #with alpha as 0.01, you will not get result  p.training( [ [0.7,0.7,0.7,0.7], [0.5, 0.5, 0.5, 0.5], [0.4, 0.4, 0.4, 0.4]], 0.01)  print("Example 2 -->") #with alpha as 0.5, you will not get result  p.training( [ [0.7,0.7,0.7,0.7], [0.5, 0.5, 0.5, 0.5], [0.4, 0.4, 0.4, 0.4]], 0.5)  print("Example 3 -->")  p.training( [ [0.2,0.2,0.2,0.2], [0.3, 0.3, 0.3, 0.3], [0.5, 0.5, 0.5, 0.5]], 0.01)  print(“performed by krunal 713”) |

**OUTPUT :**

****