**Roll No:04,37**

**Practical No:7**

**Aim: Implement Adaboost ensemble learning algorithm for the restaurant waiting problem.**

**Code:**

class Perceptron :

def AdaBoost(self, examples, K):

w = []

N = len(examples[1])

y = examples[1]

for i in range (0,N):

w.append(1/N)

print("original w:", w)

h = []

for k in range (0,K) :

print("K=", k+1)

h.append([])

h = self.L(examples,w)

error = 0

for j in range(0, N) :

#print("pair :", h[j], ":" ,y[j])

#there is huge difference in the h result and actual answer

if( h[j] != y[j] ):

error = error + w[j]

print("error :", error)

for j in range(0, N) :

if( h[j] == y[j] ):# this condition never becomes true in current example

w[j] = w[j] \* error / (1-error) # never gets executed

self.normalize(w)

print(w)

print("result using final w :")

h = self.L(examples,w)

print(h)

def normalize(self, w): # not sure how to normalize

for t in range (0,len(w)) :

normalizer = 1 / float( sum(w) )

w = [x \* normalizer for x in w]

def L(self, ex, w) : # Learning function

hresult= []

for i in range(0 , len(ex[1])):

hresult.append(0)

hresult[i] = hresult[i] + ( w[i]\*ex[0][i] )

return hresult

corresponding queue number

ex = [ [1,2,3,4,5,6] , [15,20,30,40,45,60] ]

k = 30

p = Perceptron()

p.AdaBoost(ex,k)

**Output:**

original w: [0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 1

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 2

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 3

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 4

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 5

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 6

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 7

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 8

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

K= 9

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]………

K= 30

error : 0.9999999999999999

[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]

result using final w :

[0.16666666666666666, 0.3333333333333333, 0.5, 0.6666666666666666, 0.8333333333333333, 1.0]