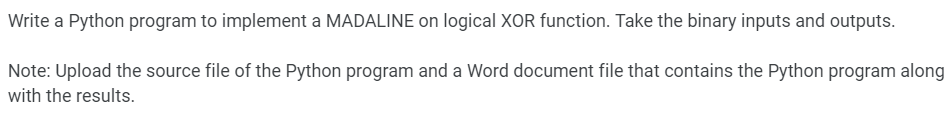
**SOFT COMPUTING**

**ASSIGNMENT -5**

**Perumalla Dharan**

**AP21110010201**



*import* numpy *as* np

class MADALINE:

def \_\_init\_\_(*self*, *learning\_rate*=0.1, *iterations*=10000):

*self*.learning\_rate = *learning\_rate*

*self*.iterations = *iterations*

*self*.w = np.random.uniform(-0.5, 1, (2, 2))

*self*.b1 = np.random.uniform(-0.5, 1, 2)

*self*.v = np.random.uniform(-0.5, 1, (2, 1))

*self*.b2 = np.random.uniform(-0.5, 1, 1)

def fx(*self*, *x*):

*return* np.where(*x* >= 0, 1, 0)

def predict(*self*, *X*):

z\_in = np.dot(*X*, *self*.w) + *self*.b1

z\_out = *self*.fx(z\_in)

y\_in = np.dot(z\_out, *self*.v) + *self*.b2

y\_out = *self*.fx(y\_in)

*return* y\_out

def train(*self*, *X*, *y*):

*for* epoch *in* range(*self*.iterations):

*for* i *in* range(len(*X*)):

z\_in = np.dot(*X*[i], *self*.w) + *self*.b1

z\_out = *self*.fx(z\_in)

y\_in = np.dot(z\_out, *self*.v) + *self*.b2

y\_out = *self*.fx(y\_in)

*if* y\_out != *y*[i]:

*if* *y*[i] == -1:

*for* j *in* range(len(z\_in)):

*if* z\_in[j] > 0:

*self*.w[:, j] += (*self*.learning\_rate \* (*X*[i]) \*(*y*[i]-z\_in[j].item()))

*self*.b1[j] += (*self*.learning\_rate \* (*y*[i]-z\_in[j].item()))

*elif* *y*[i] == 1:

closest\_to\_zero\_index = np.argmin(np.abs(z\_in))

*self*.w[:, closest\_to\_zero\_index] += (*self*.learning\_rate \* *X*[i] \*(*y*[i]-z\_in[j].item()))

*self*.b1[closest\_to\_zero\_index] += (*self*.learning\_rate \*(*y*[i]-z\_in[closest\_to\_zero\_index].item()))

*# Input Data for training (XOR logic)*

X = np.array([[0,0],

[0, 1],

[1, 0],

[1, 1]])

*# Input Data for testing (same as XOR)*

Z =np.array([[0,0],

[0, 1],

[1, 0],

[1, 1]])

y = np.array([[0], [1], [1], [0]])

madaline = MADALINE(*learning\_rate*=0.1, *iterations*=10000)

madaline.train(X, y)

predictions = madaline.predict(Z)

*# Display results*

print("Testing Data:")

print(Z)

print("Predictions after training:")

print(predictions)

