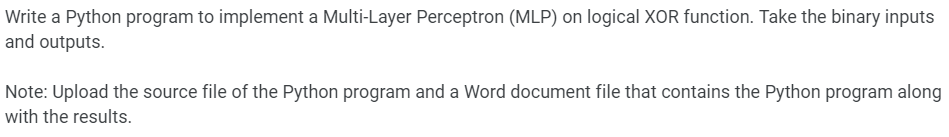
**SOFT COMPUTING**

**ASSIGNMENT -4**

**Perumalla Dharan**

**AP21110010201**



*import* numpy *as* np

class MLP:

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

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

*self*.iterations = *iterations*

*self*.weights\_input\_hidden = np.random.uniform(-0.5, 0.5, (2, 2))

*self*.bias\_hidden = np.zeros(2)

*self*.weights\_hidden\_output = np.random.uniform(-0.5, 0.5, (2, 1))

*self*.bias\_output = np.zeros(1)

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

*return* 1 / (1 + np.exp(-*x*))

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

hidden\_input = np.dot(*X*, *self*.weights\_input\_hidden) + *self*.bias\_hidden

hidden\_output = *self*.sigmoid(hidden\_input)

final\_input = np.dot(hidden\_output, *self*.weights\_hidden\_output) + *self*.bias\_output

final\_output = *self*.sigmoid(final\_input)

*return* np.round(final\_output)

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

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

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

hidden\_input = np.dot(*X*[i], *self*.weights\_input\_hidden) + *self*.bias\_hidden

hidden\_output = *self*.sigmoid(hidden\_input)

final\_input = np.dot(hidden\_output, *self*.weights\_hidden\_output) + *self*.bias\_output

final\_output = *self*.sigmoid(final\_input)

y\_pred = np.round(final\_output)

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

output\_error = *y*[i] - final\_output

hidden\_error = output\_error.dot(*self*.weights\_hidden\_output.T)

*self*.weights\_hidden\_output += *self*.learning\_rate \* np.outer(hidden\_output, output\_error)

*self*.bias\_output += *self*.learning\_rate \* output\_error

*self*.weights\_input\_hidden += *self*.learning\_rate \* np.outer(*X*[i], hidden\_error)

*self*.bias\_hidden += *self*.learning\_rate \* hidden\_error

def print\_weights(*self*):

print("Input-Hidden Weights:\n", *self*.weights\_input\_hidden)

print("Hidden-Output Weights:\n", *self*.weights\_hidden\_output)

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

[0, 1],

[1, 0],

[1, 1]])

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

mlp = MLP(*learning\_rate*=0.1, *iterations*=10000)

mlp.train(X, y)

mlp.print\_weights()

predictions = mlp.predict(X)

print("Predictions after training:")

print(np.round(predictions))

Output -

