**Academic Year: 2024-2025 Name of Student:Diya Thakkar**

**Semester: 6 Student ID:22107040**

**Class / Branch: DS Date Of Performance:1-4-25**

**Subject: ML Date Of Submission:1-4-25 Name of Instructor:Prof. Ujawala Pagare**

# Experiment No. 8

**Aim:**- To implement the Single Layer Perceptron Learning algorithm.

**Output:-**

# Initialize weights and bias

w1, w2, b = 0, 0, 0 # Start with zero

learning\_rate = 1 # Learning rate

# Bipolar activation function

def activate(y\_net):

return 1 if y\_net > 0 else -1

# Train the perceptron

def train\_perceptron(inputs, targets, learning\_rate, max\_epochs):

global w1, w2, b

print("\nTraining Perceptron Step-by-Step:")

for epoch in range(max\_epochs):

weights\_changed = False # Track if weights change

print(f"\nEpoch {epoch + 1}:")

print("x1 x2 Target Net Input Output Δw1 Δw2 Δb w1 w2 b")

for i in range(len(inputs)):

x1, x2 = inputs[i] # Get input values

target = targets[i] # Expected output

# Compute net input and apply activation function

y\_net = w1 \* x1 + w2 \* x2 + b

output = activate(y\_net)

# Compute weight updates using (y\_true - y\_predicted)

error = target - output

delta\_w1 = learning\_rate \* error \* x1

delta\_w2 = learning\_rate \* error \* x2

delta\_b = learning\_rate \* error

# Update weights and bias if there is an error

if error != 0:

w1 += delta\_w1

w2 += delta\_w2

b += delta\_b

weights\_changed = True # Track change

else:

delta\_w1, delta\_w2, delta\_b = 0, 0, 0 # No update if output == target

# Print step-by-step updates

print(f"{x1:2} {x2:2} {target:2} {y\_net:3} {output:3} {delta\_w1:3} {delta\_w2:3} {delta\_b:3} {w1:3} {w2:3} {b:3}")

# Stop early if no weights changed

if not weights\_changed:

print("\nConverged! Stopping early.")

break

# Define inputs and targets for the AND operation

inputs = [(1, 1), (1, -1), (-1, 1), (-1, -1)]

targets = [1, -1, -1, -1] # Expected AND output for bipolar values

# Train the perceptron

train\_perceptron(inputs, targets, learning\_rate=1, max\_epochs=10)

# Final Model Outputs

print("\nFinal Model Outputs:")

for x1, x2 in inputs:

output = activate(w1 \* x1 + w2 \* x2 + b)

print(f"Input: ({x1}, {x2}) → Output: {output}")

