

Exercise-6a

Aim: Build a perceptron model to simulate AND and OR logic gates.

Google Account
 23501A0537 DUDEKULA CHINNA
 NASARVALI
 23501a0537@pvpsit.ac.in

```
import numpy as np

# Step 1: Take number of training samples
n = int(input("Enter number of training samples (rows in truth table): "))

X = []
Y = []

print("\nEnter input features and output (space separated):")
print("Format: x1 x2 ... y (where y is output as 1 or -1)")

for i in range(n):
    data = list(map(int, input(f"Sample {i+1}: ").split()))
    X.append(data[:-1] + [1]) # add bias term = 1
    Y.append(data[-1])

X = np.array(X)
Y = np.array(Y)

# Automatically detect number of features
m = X.shape[1] - 1 # (excluding bias)

# Step 2: Initialize weights
w = np.zeros(m+1, dtype=int) # +1 for bias
print("\nInitial weights:", w)

# Step 3: Perceptron Learning
epochs = 10
for epoch in range(epochs):
    error_count = 0
    for i in range(n):
        # Calculate the dot product
        dot_product = np.dot(w, X[i])
        # Predict the output (1 if dot product >= 0, -1 otherwise)
        y_pred = 1 if dot_product >= 0 else -1

        if y_pred != Y[i]: # misclassified
            w = w + Y[i] * X[i]
            error_count += 1
            print(f"Update in epoch {epoch+1}: w = {w}")
    if error_count == 0:
        print("\nTraining complete in epoch:", epoch+1)
        break

print("\nFinal Weights:", w)

# Step 4: Test user input
print("\n--- Testing ---")
while True:
    test = list(map(int, input("Enter test input features (space separated, or -1 to stop): ").split()))
    if test[0] == -1:
        break
    test.append(1) # bias
    # Calculate the dot product for the test input
    test_dot_product = np.dot(w, test)
    # Predict the output (1 if dot product >= 0, -1 otherwise)
    result = 1 if test_dot_product >= 0 else -1
    print("Output:", result)
```

Enter number of training samples (rows in truth table): 4

Enter input features and output (space separated):
 Format: x1 x2 ... y (where y is output as 1 or -1)
 Sample 1: 0 0 1
 Sample 2: 0 1 1
 Sample 3: 1 0 1
 Sample 4: 1 1 -1

```
Initial weights: [0 0 0]
Update in epoch 1: w = [-1 -1 -1]
Update in epoch 2: w = [-1 -1 0]
Update in epoch 2: w = [-1 0 1]
Update in epoch 2: w = [-2 -1 0]
Update in epoch 3: w = [-2 0 1]
Update in epoch 3: w = [-1 0 2]
Update in epoch 3: w = [-2 -1 1]
Update in epoch 4: w = [-1 -1 2]
Update in epoch 4: w = [-2 -2 1]
Update in epoch 5: w = [-2 -1 2]
```

Training complete in epoch: 6

Final Weights: [-2 -1 2]

--- Testing ---

Enter test input features (space separated, or -1 to stop): 0 0

Output: 1

Enter test input features (space separated, or -1 to stop): 1 0

Output: 1

Enter test input features (space separated, or -1 to stop): 0 1

Output: 1

Enter test input features (space separated, or -1 to stop): 1 1

Output: -1