Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

AI EXPERIMENT 6 - Perceptron Learning

Theory:

Jigar Siddhpura
60004210155

AI - Experiment 6

Aim : Program on perceptron learning to design a pattern classifier.

Theory : 1. Perceptron learning algo is one of fundamental supervised learning algo technique used for binary classification tasks.

2. It is based on a simplified model of a biological neuron.
3. It learns a ~~tes~~ linear decision boundary that separates classes in a future space.
4. Basic concepts of perceptron Learning :
   Inputs $(x_1, x_2, \dots x_n)$
   Output (1 or 0)

Perceptron computes weighted sum of inputs & compares it to a threshold (θ)

$$\text{output} = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i x_i + b > \theta \\ 0, & \text{otherwise} \end{cases}$$

Training :

1. Goal of this algorithm is that it adjusts the weights & biases such that perceptron classifies.
2. Start with random weights & biases.
3. For each sample $(x_1, x_2, \dots x_n)$, with class label 'y', compute $\hat{y}$ (predicted output)
4. Update weights & biases using perceptron model

$$w_i \leftarrow w_i + \alpha (y - \hat{y}) \cdot x_i$$
$$b \leftarrow b + \alpha (y - \hat{y})$$

where, α is the learning rate, controlling weight convergence.

(5) Algo converges if the training data is linearly separable. In other if there exists a hyperplane that perfectly separates the classes; perceptron will find a set of weights & bias, that achieve this separation.

Conclusions:

Implementing perceptron learning algo offers a foundational understanding of how neural networks work.
While it has limitations, understanding perceptron's principle is essential for developing more advanced neural networks architecture & their applications in solving complex real world problems.

Code :

```python
import numpy as np

# Input features
X = np.array([
    [1, 0, 1, 0, 0, 0, 1, 1, 1],
    [1, 0, 0, 1, 0, 0, 1, 1, 1],
    [1, 1, 0, 1, 0, 0, 1, 1, 1],
    [1, 0, 0, 1, 0, 0, 1, 1, 1],
    [1, 0, 0, 1, 0, 1, 1, 1, 1],
    [1, 0, 1, 0, 1, 1, 1, 0, 1],
    [1, 0, 1, 1, 1, 0, 1, 0, 1],
    [1, 0, 1, 1, 0, 1, 1, 0, 1],
    [1, 0, 0, 1, 1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1, 1, 1, 0, 1]
])

# Initial weights
W = np.array([1, -1, 0, 0.5, 1, 1, 1, 0.5, 1])

# Desired outputs
d = np.array([0, 0, 0, 1, 0, 1, 1, 1, 1, 1])

# Learning rate
c = 1

# Number of epochs
epochs = 6

# Training the perceptron
for epoch in range(epochs):
    print("Iteration ", epoch + 1)

    for i in range(len(X)):
        # Compute the net input
        net = np.dot(X[i], W)

        # Apply the step function
        op = 1 if net > 0 else 0

        # Compute the error
        error = d[i] - op

        # Update weights
        dW = c * error * X[i]
        W += dW

        print("W", i, W)

    print("\nW after ", epoch + 1, " epochs ", W)

print("Final W after ", epochs, "epochs:")
print(W)

# Testing the perceptron with a new input
test_input = [1, 0, 0, 1, 0, 0, 1, 1, 0]
net = np.dot(test_input, W)

# Apply the step function to get the output
output = 1 if net > 0 else 0

print("Output for test input:", output)
```

Output :

```
PS D:\SEM 5\AI\EXPERIMENTS> python -u "d:\SEM 5\AI\EXPERIMENTS\perceptron.py"
Iteration  1
W 0 [ 0.  -1.  -1.   0.5  1.   1.   0.  -0.5  0. ]
W 1 [ 0.  -1.  -1.   0.5  1.   1.   0.  -0.5  0. ]
W 2 [ 0.  -1.  -1.   0.5  1.   1.   0.  -0.5  0. ]
W 3 [ 1.  -1.  -1.   1.5  1.   1.   1.   0.5  1. ]
W 4 [ 0.  -1.  -1.   0.5  1.   0.   0.  -0.5  0. ]
W 5 [ 1.  -1.   0.   0.5  2.   1.   1.  -0.5  1. ]
W 6 [ 1.  -1.   0.   0.5  2.   1.   1.  -0.5  1. ]
W 7 [ 1.  -1.   0.   0.5  2.   1.   1.  -0.5  1. ]
W 8 [ 1.  -1.   0.   0.5  2.   1.   1.  -0.5  1. ]
W 9 [ 1.  -1.   0.   0.5  2.   1.   1.  -0.5  1. ]

W after  1  epochs  [ 1.  -1.   0.   0.5  2.   1.   1.  -0.5  1. ]
Iteration  2
W 0 [ 0.  -1.  -1.   0.5  2.   1.   0.  -1.5  0. ]
W 1 [ 0.  -1.  -1.   0.5  2.   1.   0.  -1.5  0. ]
W 2 [ 0.  -1.  -1.   0.5  2.   1.   0.  -1.5  0. ]
W 3 [ 1.  -1.  -1.   1.5  2.   1.   1.  -0.5  1. ]
W 4 [ 0.  -1.  -1.   0.5  2.   0.   0.  -1.5  0. ]
W 5 [ 0.  -1.  -1.   0.5  2.   0.   0.  -1.5  0. ]
W 6 [ 0.  -1.  -1.   0.5  2.   0.   0.  -1.5  0. ]
W 7 [ 1.  -1.   0.   1.5  2.   1.   1.  -1.5  1. ]
W 8 [ 1.  -1.   0.   1.5  2.   1.   1.  -1.5  1. ]
W 9 [ 1.  -1.   0.   1.5  2.   1.   1.  -1.5  1. ]

W after  2  epochs  [ 1.  -1.   0.   1.5  2.   1.   1.  -1.5  1. ]
Iteration  3
W 0 [ 0.  -1.  -1.   1.5  2.   1.   0.  -2.5  0. ]
W 1 [ 0.  -1.  -1.   1.5  2.   1.   0.  -2.5  0. ]
W 2 [ 0.  -1.  -1.   1.5  2.   1.   0.  -2.5  0. ]
W 3 [ 1.  -1.  -1.   2.5  2.   1.   1.  -1.5  1. ]
W 4 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
W 5 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
W 6 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
W 7 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
W 8 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
W 9 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
```

```
W after  3  epochs  [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
Iteration  4
W 0 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
W 1 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
W 2 [ 0.  -1.  -1.   1.5  2.   0.   0.  -2.5  0. ]
W 3 [ 1.  -1.  -1.   2.5  2.   0.   1.  -1.5  1. ]
W 4 [ 0.  -1.  -1.   1.5  2.  -1.   0.  -2.5  0. ]
W 5 [ 1.  -1.   0.   1.5  3.   0.   1.  -2.5  1. ]
W 6 [ 1.  -1.   0.   1.5  3.   0.   1.  -2.5  1. ]
W 7 [ 1.  -1.   0.   1.5  3.   0.   1.  -2.5  1. ]
W 8 [ 1.  -1.   0.   1.5  3.   0.   1.  -2.5  1. ]
W 9 [ 1.  -1.   0.   1.5  3.   0.   1.  -2.5  1. ]

W after  4  epochs  [ 1.  -1.   0.   1.5  3.   0.   1.  -2.5  1. ]
Iteration  5
W 0 [ 0.  -1.  -1.   1.5  3.   0.   0.  -3.5  0. ]
W 1 [ 0.  -1.  -1.   1.5  3.   0.   0.  -3.5  0. ]
W 2 [ 0.  -1.  -1.   1.5  3.   0.   0.  -3.5  0. ]
W 3 [ 1.  -1.  -1.   2.5  3.   0.   1.  -2.5  1. ]
W 4 [ 0.  -1.  -1.   1.5  3.  -1.   0.  -3.5  0. ]
W 5 [ 0.  -1.  -1.   1.5  3.  -1.   0.  -3.5  0. ]
W 6 [ 0.  -1.  -1.   1.5  3.  -1.   0.  -3.5  0. ]
W 7 [ 1.  -1.   0.   2.5  3.   0.   1.  -3.5  1. ]
W 8 [ 1.  -1.   0.   2.5  3.   0.   1.  -3.5  1. ]
W 9 [ 1.  -1.   0.   2.5  3.   0.   1.  -3.5  1. ]

W after  5  epochs  [ 1.  -1.   0.   2.5  3.   0.   1.  -3.5  1. ]
Iteration  6
W 0 [ 1.  -1.   0.   2.5  3.   0.   1.  -3.5  1. ]
W 1 [ 0.  -1.   0.   1.5  3.   0.   0.  -4.5  0. ]
W 2 [ 0.  -1.   0.   1.5  3.   0.   0.  -4.5  0. ]
W 3 [ 1.  -1.   0.   2.5  3.   0.   1.  -3.5  1. ]
W 4 [ 0.  -1.   0.   1.5  3.  -1.   0.  -4.5  0. ]
W 5 [ 0.  -1.   0.   1.5  3.  -1.   0.  -4.5  0. ]
W 6 [ 0.  -1.   0.   1.5  3.  -1.   0.  -4.5  0. ]
W 7 [ 0.  -1.   0.   1.5  3.  -1.   0.  -4.5  0. ]
W 8 [ 1.  -1.   0.   2.5  4.   0.   1.  -3.5  1. ]
W 9 [ 1.  -1.   0.   2.5  4.   0.   1.  -3.5  1. ]

W after  6  epochs  [ 1.  -1.   0.   2.5  4.   0.   1.  -3.5  1. ]
Final W after  6 epochs:
[ 1.  -1.   0.   2.5  4.   0.   1.  -3.5  1. ]
Output for test input: 1
PS D:\SEM 5\AI\EXPERIMENTS>
```