EXPERIMENT 1.2

Student Name: Samuel **UID: 24MAI10018**

Branch: AIML **Section/Group: 24MAI-1**

Semester: 1 **Date of Performance: / /2025**

Subject Name: SC Lab **Subject Code: 24CSH–668**

AIM: Implementation of Simple Neural Network (McCulloh-Pitts model).

Software Used : Online MATLAB

THEORY: The McCulloch-Pitts (M-P) model, introduced in 1943, is one of the foundational models of artificial neural networks. It simplifies the biological neuron into a mathematical model that takes binary inputs (0 or 1), applies weights to each input, calculates a weighted sum, and produces a binary output based on a threshold. If the weighted sum equals or exceeds the threshold, the neuron "fires" and outputs 1; otherwise, it outputs 0. This model can mimic simple logic gates like AND, OR, and NOT, making it a cornerstone for understanding how neural networks process information. Despite its simplicity, the M-P model demonstrates the fundamental concept of decision-making in neural systems.

Key Components of M-P Model:

1. **Input Layer**: Accepts binary inputs (0 or 1).
2. **Weights**: Each input has an associated weight.
3. **Summation Function**: Computes the weighted sum of inputs.
4. **Activation Function**: Applies a threshold to determine the output (0 or 1).

PSEUDOCODE

AND Gate

1. Input: Two binary values x1 and x2.
2. Parameters:
   - Weight w1 for x1.
   - Weight w2 for x2.
   - Threshold theta.
3. Steps:
   - Compute the weighted sum: yin = x1 * w1 + x2 * w2.
   - Apply the activation function:
     - If yin >= theta, set output = 1.
     - Otherwise, set output = 0.
4. Output: Binary value output.

OR Gate

1. Input: Two binary values x1 and x2.
2. Parameters:
   - Weight w1 for x1.

- Weight w2 for x2.
- Threshold theta.

3. Steps:
   - Compute the weighted sum: yin = x1 * w1 + x2 * w2.
   - Apply the activation function:
     - If yin >= theta, set output = 1.
     - Otherwise, set output = 0.
4. Output: Binary value output.

NOT Gate

1. Input: A single binary value x.
2. Parameters:
   - Weight w for x (negative value).
   - Threshold theta.
3. Steps:
   - Compute the weighted sum: yin = x * w.
   - Apply the activation function:
     - If yin >= theta, set output = 1.
     - Otherwise, set output = 0.
4. Output: Binary value output.

SOURCE CODE:

AND GATE

```
% MATLAB Code for AND Gate using McCulloch-Pitts Model

clc;
clear all;

disp('--- McCulloch-Pitts Model: AND Gate ---');

% Define inputs for AND Gate
x1 = [0 0 1 1]; % First input
x2 = [0 1 0 1]; % Second input

% Ask the user for weights and threshold w1
= input('Enter weight for x1 (w1): '); w2 =
input('Enter weight for x2 (w2): '); theta =
input('Enter threshold value (theta): ');

% Initialize output
y_and = zeros(1, 4);

% Calculate the output
```

```
for i = 1:4
    % Compute weighted sum
    yin = x1(i)*w1 + x2(i)*w2;

    % Apply activation
    function if yin >= theta
        y_and(i) =
    1; else
        y_and(i) =
    0; end
end

% Display inputs and output
disp(['Inputs: X1=', num2str(x1), ', X2=', num2str(x2)]);
disp(['Weights: w1=', num2str(w1), ', w2=', num2str(w2)]);
disp(['Threshold: theta=', num2str(theta)]);
disp(['Output (AND Gate): ', num2str(y_and)]);
```

OR GATE

```
% MATLAB Code for OR Gate using McCulloch-Pitts Model

clc;
clear all;

disp('--- McCulloch-Pitts Model: OR Gate ---');

% Define inputs for OR Gate
x1 = [0 0 1 1]; % First input
x2 = [0 1 0 1]; % Second input

% Ask the user for weights and threshold w1
= input('Enter weight for x1 (w1): '); w2 =
input('Enter weight for x2 (w2): '); theta =
input('Enter threshold value (theta): ');

% Initialize output
y_or = zeros(1, 4);

% Calculate the
output for i = 1:4
    % Compute weighted sum
    yin = x1(i)*w1 + x2(i)*w2;

    % Apply activation function
```

```
    if yin >= theta
        y_or(i) = 1;
    else
        y_or(i) = 0;
    end
end

% Display inputs and output
disp(['Inputs: X1=', num2str(x1), ', X2=', num2str(x2)]);
disp(['Weights: w1=', num2str(w1), ', w2=', num2str(w2)]);
disp(['Threshold: theta=', num2str(theta)]);
disp(['Output (OR Gate): ', num2str(y_or)]);
```

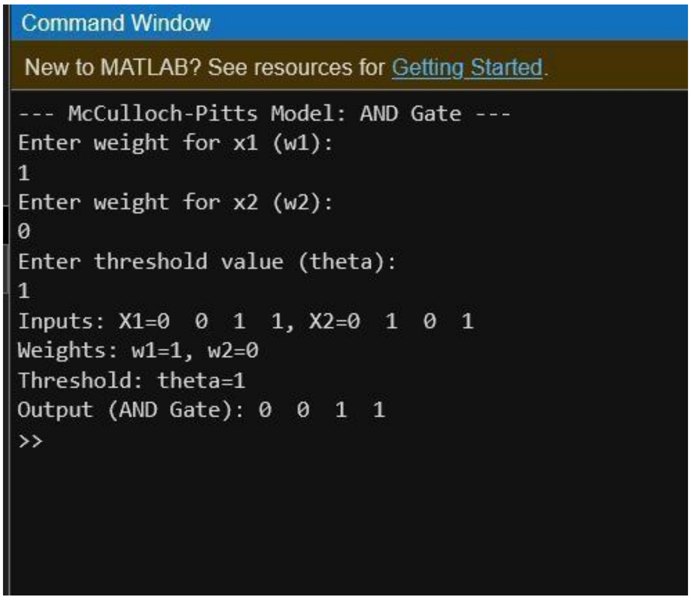
NOT GATE

```
% MATLAB Code for NOT Gate using McCulloch-Pitts Model

clc;
clear all;

disp('--- McCulloch-Pitts Model: NOT Gate ---');

% Define input for NOT Gate
x1 = [0 1]; % Single binary input for NOT Gate

% Ask the user for weight and threshold
w = input('Enter weight for x1 (w): '); % Single weight (negative for NOT)
theta = input('Enter threshold value (theta): ');

% Initialize output
y_not = zeros(1, 2);

% Calculate the
output for i = 1:2
    % Compute weighted
    sum yin = x1(i) * w;

    % Apply activation
    function if yin >= theta
        y_not(i) =
    1; else
        y_not(i) =
    0; end
end

% Display input and output
disp(['Input: X1=', num2str(x1)]);
```

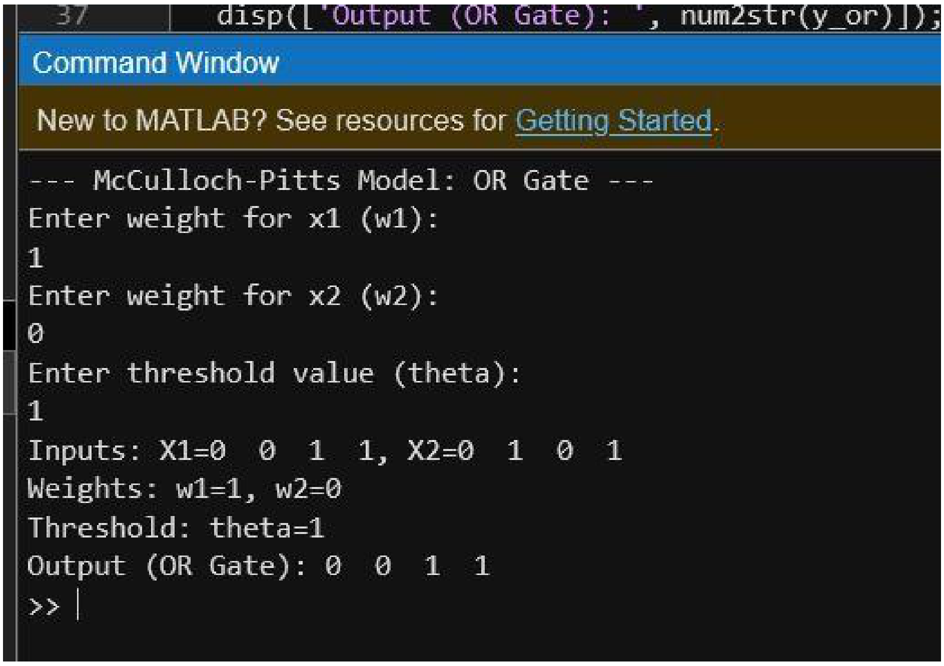
```
disp(['Weight: w=', num2str(w)]);
disp(['Threshold: theta=', num2str(theta)]);
disp(['Output (NOT Gate): ', num2str(y_not)]);
```

SCREENSHOT OF OUTPUT

AND GATE:



OR GATE:



NOT GATE:

```
New to MATLAB? See resources for Getting Started.

--- McCulloch-Pitts Model: NOT Gate ---
Enter weight for x1 (w):
1
Enter threshold value (theta):
1
Input: X1=0  1
Weight: w=1
Threshold: theta=1
Output (NOT Gate): 0  1
>>
```

LEARNING OUTCOMES:

Upon successful completion of this module, learners will be able to:

Understanding Basic Neural Concepts: Learn the foundational principles of how artificial neurons function, inspired by biological neurons.
Binary Input and Output Processing: Understand how binary inputs are processed to produce binary outputs using weights and thresholds.
Implementation of Logic Gates: Gain the ability to simulate basic logical operations (AND, OR, NOT) using the M-P model.
Weight and Threshold Significance: Recognize the role of weights and thresholds in decision-making processes within a neural network.
Simplistic Neural Network Modeling: Build a basic understanding of how simple neural networks are structured and operated.