Table of Contents


```
% FML_LAB_4
% Author: ARISH

% Q1) McCulloch-Pitts Neuron Simulation for AND, OR, AND-NOT

clear; clc;

% Define all possible binary input combinations
inputs = [0 0; 0 1; 1 0; 1 1];

% Define logic functions with weights and thresholds
logic_gates = {
    'AND',     [1 1], 2;
    'OR',      [1 1], 1;
    'AND-NOT', [1 -1], 1
};

% Activation function
activate = @(x, theta) double(x >= theta);

% Loop through each logic gate
for g = 1:size(logic_gates,1)
    gate_name = logic_gates{g,1};
    weights = logic_gates{g,2};
    theta = logic_gates{g,3};

    fprintf('\nLogic Gate: %s\n', gate_name);
    fprintf(' x1 x2 | y\n');
    fprintf('--------------\n');

    % Test all input combinations
    for i = 1:size(inputs,1)
        x = inputs(i,:);
        net_input = sum(weights .* x);
        y = activate(net_input, theta);
        fprintf('  %d  %d | %d\n', x(1), x(2), y);
    end
end
```

```
Logic Gate: AND
 x1 x2 | y
--------------
  0  0 | 0
  0  1 | 0
```

```
  1   0 | 0
  1   1 | 1

Logic Gate: OR
 x1 x2 | y
--------------
  0   0 | 0
  0   1 | 1
  1   0 | 1
  1   1 | 1

Logic Gate: AND-NOT
 x1 x2 | y
--------------
  0   0 | 0
  0   1 | 0
  1   0 | 1
  1   1 | 0
```

Q2) McCulloch-Pitts Neural Network for XOR

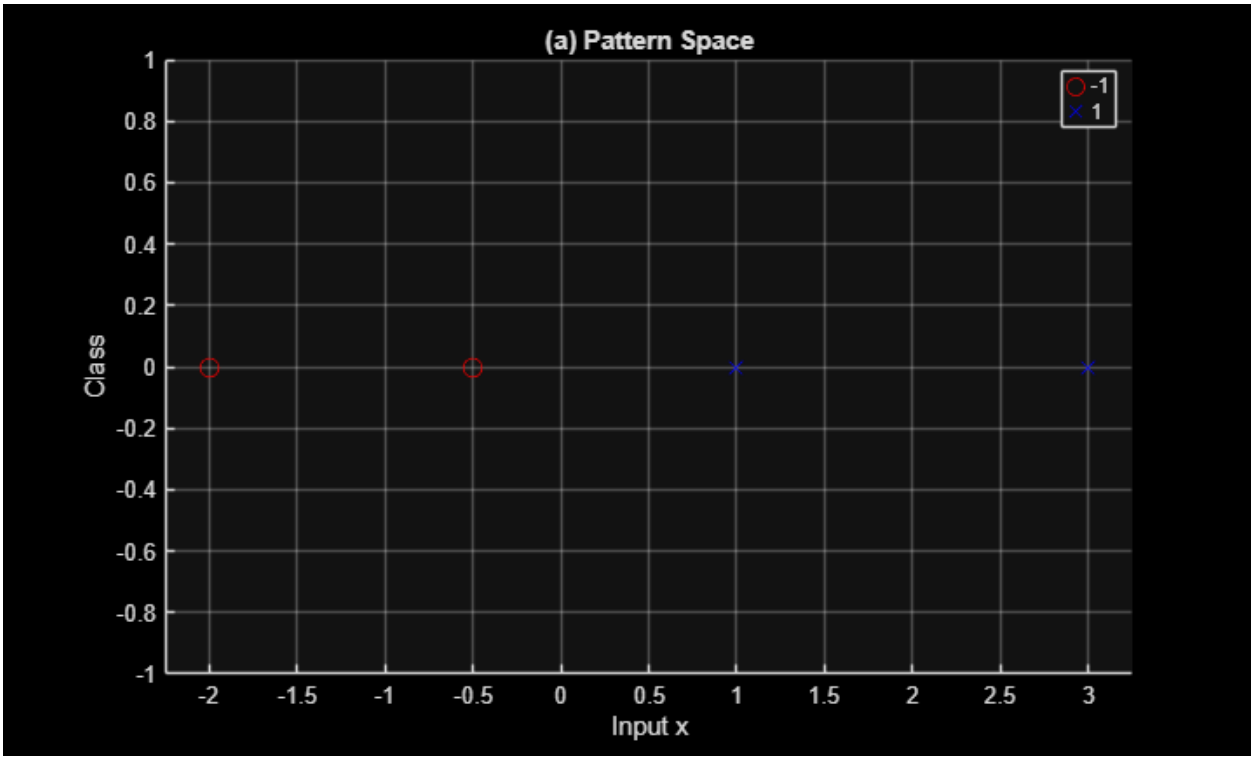
```
%clear; clc;

% Define all input combinations
inputs = [0 0; 0 1; 1 0; 1 1];

% Activation function
activate = @(net, theta) double(net >= theta);

fprintf('XOR via McCulloch-Pitts Network\n');
fprintf(' x1 x2 | z1 z2 | y\n');
fprintf('-----------------------\n');

for i = 1:size(inputs,1)
    x1 = inputs(i,1);
    x2 = inputs(i,2);

    % Hidden neurons
    % z1 = x1 AND NOT x2 → weights: [1 -1], threshold = 1
    z1_net = x1*1 + x2*(-1);
    z1 = activate(z1_net, 1);

    % z2 = x2 AND NOT x1 → weights: [-1 1], threshold = 1
    z2_net = x1*(-1) + x2*1;
    z2 = activate(z2_net, 1);

    % Output neuron: y = z1 OR z2 → weights: [1 1], threshold = 1
    y_net = z1*1 + z2*1;
    y = activate(y_net, 1);

    fprintf('  %d  %d |  %d  %d | %d\n', x1, x2, z1, z2, y);
end
```

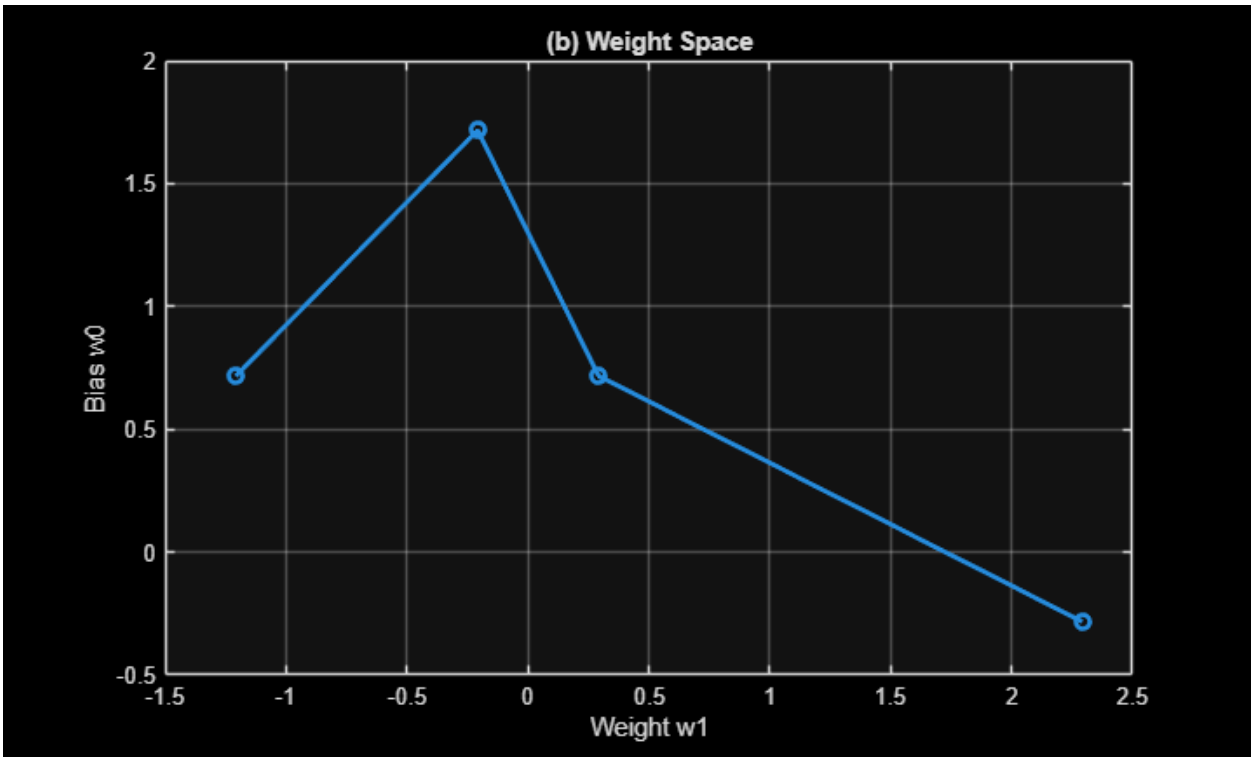
```
XOR via McCulloch-Pitts Network
 x1 x2 | z1 z2 | y
-----------------------
  0  0 |  0  0 | 0
  0  1 |  0  1 | 1
  1  0 |  1  0 | 1
  1  1 |  0  0 | 0
```

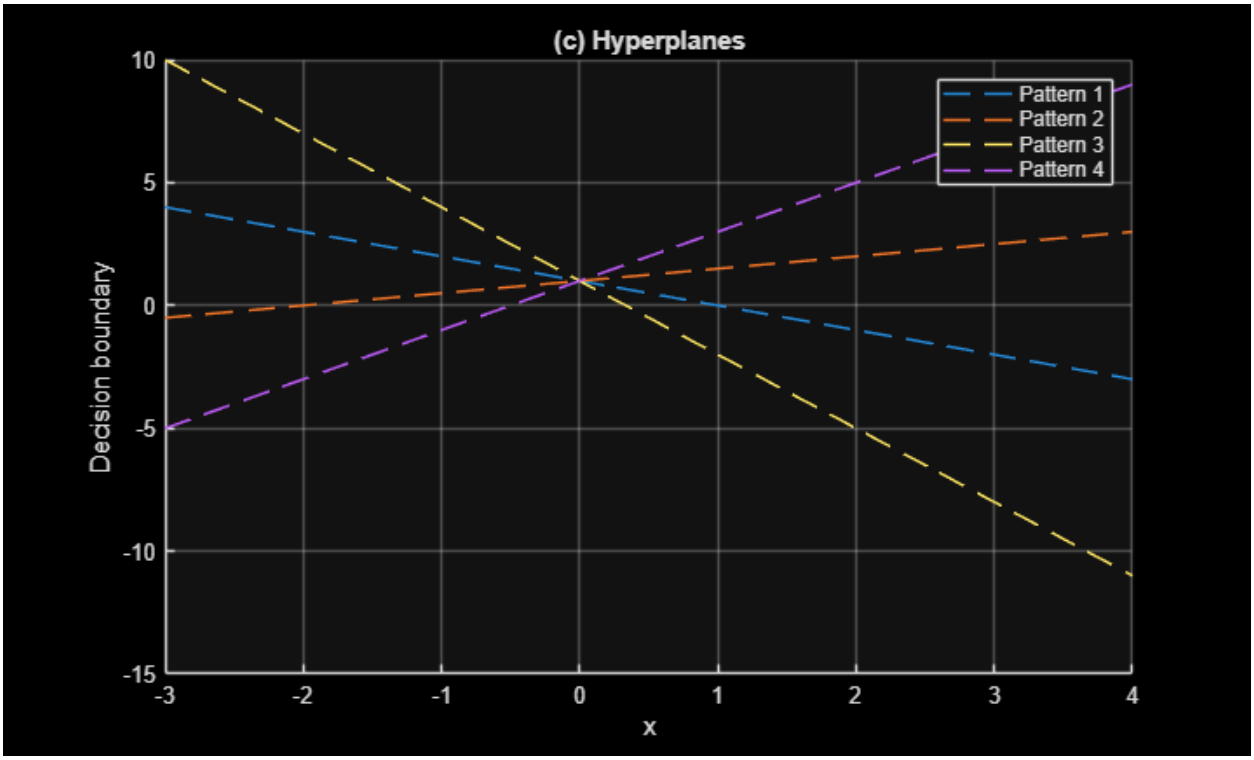
3) Discrete Perceptron Algorithm Visualization

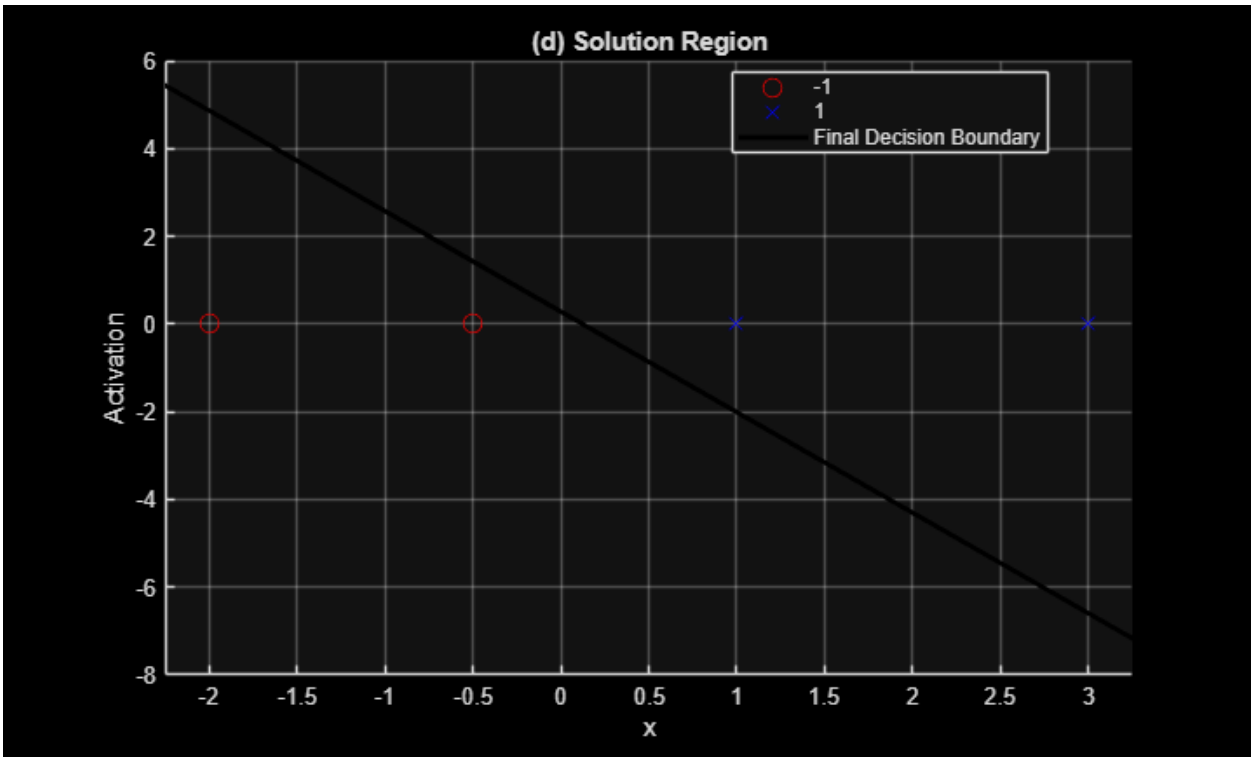
```
%clear; clc;

% Training Data
X = [1; -0.5; 3; -2];          % Input patterns (1D)
D = [1; -1; 1; -1];            % Desired outputs (class labels)

% Convert to 2D input vectors with bias term
X_aug = [X, ones(size(X))];  % Each row: [x_i, bias]

% Initialize weights
w = randn(1,2);                % Random initial weights [w1, bias]
eta = 1;                       % Learning rate
max_iter = 20;

% Store weight updates
W_history = w;                 % Track weight evolution

% Perceptron Learning
for iter = 1:max_iter
    error_count = 0;
    for i = 1:length(X)
        x_i = X_aug(i,:);      % [x_i, 1]
        y = sign(dot(w, x_i));
        if y ~= D(i)
            w = w + eta * D(i) * x_i;
            W_history = [W_history; w];
            error_count = error_count + 1;
        end
    end
    if error_count == 0
        break;
    end
end

% (a) Pattern Space
figure;
gscatter(X, zeros(size(X)), D, 'rb', 'ox', 8);
xlabel('Input x'); ylabel('Class'); title('(a) Pattern Space');
ylim([-1 1]); grid on;

% (b) Weight Space
figure;
plot(W_history(:,1), W_history(:,2), '-o', 'LineWidth', 2);
```

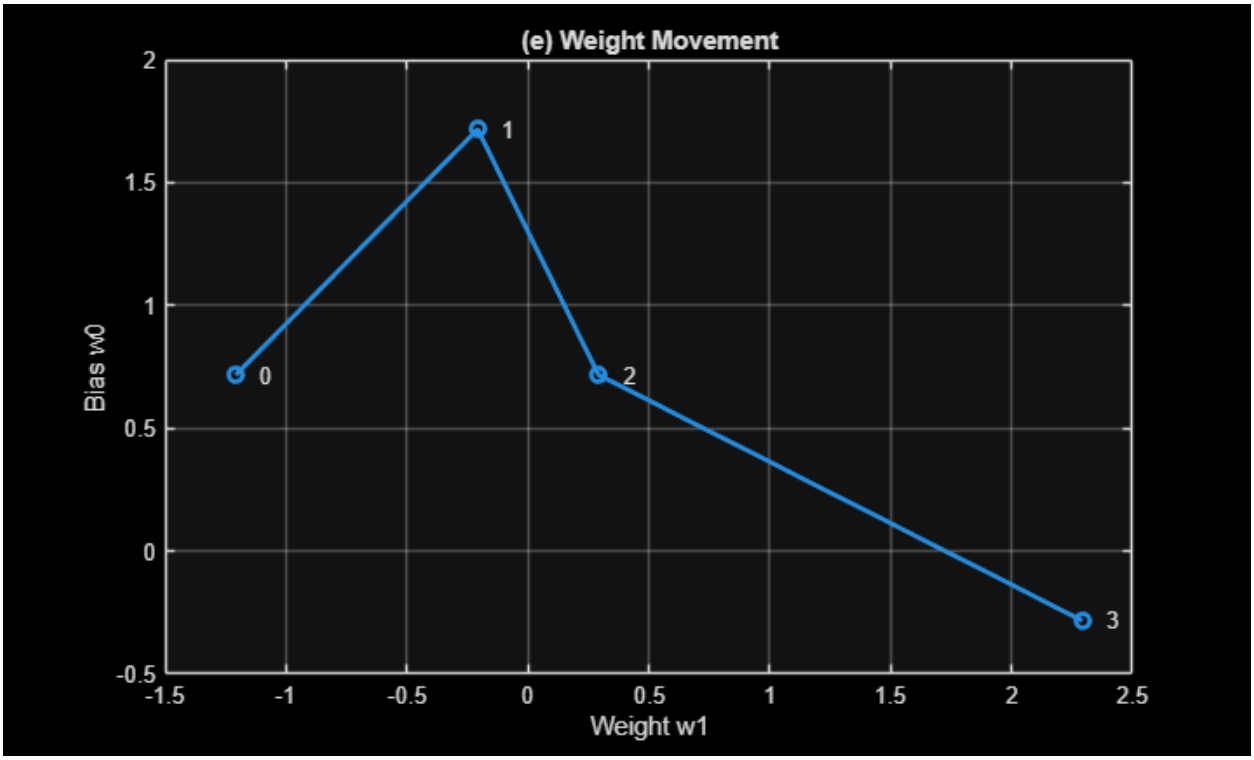
```
xlabel('Weight w1'); ylabel('Bias w0'); title('(b) Weight Space');
grid on;

% (c) Hyperplanes for Each Pattern
figure; hold on;
x_vals = linspace(-3, 4, 100);
for i = 1:length(X)
    % Hyperplane: w1*x + w0 = 0 → w0 = -w1*x
    w1 = X(i); w0 = 1;  % Assume unit weight for visualization
    y_vals = -w1 * x_vals + w0;
    plot(x_vals, y_vals, '--', 'DisplayName', sprintf('Pattern %d', i));
end
xlabel('x'); ylabel('Decision boundary'); title('(c) Hyperplanes');
legend; grid on;

% (d) Solution Region
% Plot final decision boundary
figure; hold on;
gscatter(X, zeros(size(X)), D, 'rb', 'ox', 8);
x_vals = linspace(-3, 4, 100);
y_vals = -(w(1) * x_vals + w(2));
plot(x_vals, y_vals, 'k-', 'LineWidth', 2, 'DisplayName','Final Decision
Boundary');
xlabel('x'); ylabel('Activation'); title('(d) Solution Region');
legend; grid on;

% (e) Weight Movement
figure;
plot(W_history(:,1), W_history(:,2), '-o', 'LineWidth', 2);
text(W_history(:,1)+0.1, W_history(:,2), string(0:size(W_history,1)-1));
xlabel('Weight w1'); ylabel('Bias w0'); title('(e) Weight Movement');
grid on;
```


(a) Pattern Space
-1
1
Class
Input x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3


(b) Weight Space
Bias w0
Weight w1
2
1.5
1
0.5
0
-0.5
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5


(c) Hyperplanes
Pattern 1
Pattern 2
Pattern 3
Pattern 4
Decision boundary
x
10
5
0
-5
-10
-15
-3
-2
-1
0
1
2
3
4


(d) Solution Region
-1
1
Final Decision Boundary
Activation
x
6
4
2
0
-2
-4
-6
-8
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3



Published with MATLAB® R2025a