Table of Contents


```
% FML_LAB_3: Single Neuron Training for Logical Functions
% Author: ARISH

clear; clc;
```

Choose logical function

Format: [x1, x2, target] Uncomment the desired function:

```
% (i) AND (binary)
data = [1 1  1;
        1 0 -1;
        0 1 -1;
        0 0 -1];

% % (ii) AND (bipolar)
% data = [1  1   1;
%         1 -1  -1;
%        -1  1  -1;
%        -1 -1  -1];

% % (iii) OR (bipolar)
% data = [1  1   1;
%         1 -1   1;
%        -1  1   1;
%        -1 -1  -1];

% % (iv) AND NOT (bipolar)
% data = [1  1  -1;
%         1 -1   1;
%        -1  1  -1;
%        -1 -1  -1];

% % (v) XOR (bipolar)
% data = [1  1  -1;
%         1 -1   1;
%        -1  1   1;
%        -1 -1  -1];

X = data(:,1:2);              % Inputs
T = data(:,3);                % Targets
X_aug = [X ones(size(X,1),1)];% Add bias term
```

(a) Perceptron Algorithm

```
w_p = randn(1,3);              % Initial weights
eta = 1;                       % Learning rate
max_iter = 20;

for iter = 1:max_iter
    for i = 1:size(X_aug,1)
        y = sign(dot(w_p, X_aug(i,:)));
        if y ~= T(i)
            w_p = w_p + eta * T(i) * X_aug(i,:);
        end
    end
end

fprintf('\nPerceptron Weights: [%.2f %.2f %.2f]\n', w_p);
fprintf('Decision boundary: %.2fx1 + %.2fx2 + %.2f = 0\n', w_p(1), w_p(2),
w_p(3));
```

Perceptron Weights: [0.80 1.88 -2.51]
Decision boundary: 0.80x1 + 1.88x2 + -2.51 = 0

(b) Widrow-Hoff Learning Rule

```
w_wh = randn(1,3);             % Initial weights
eta = 0.1;

for iter = 1:max_iter
    for i = 1:size(X_aug,1)
        y = dot(w_wh, X_aug(i,:));
        e = T(i) - y;
        w_wh = w_wh + eta * e * X_aug(i,:);
    end
end

fprintf('\nWidrow-Hoff Weights: [%.2f %.2f %.2f]\n', w_wh);
fprintf('Decision boundary: %.2fx1 + %.2fx2 + %.2f = 0\n', w_wh(1), w_wh(2),
w_wh(3));
```

Widrow-Hoff Weights: [0.79 0.85 -1.34]
Decision boundary: 0.79x1 + 0.85x2 + -1.34 = 0

Plotting Decision Boundaries

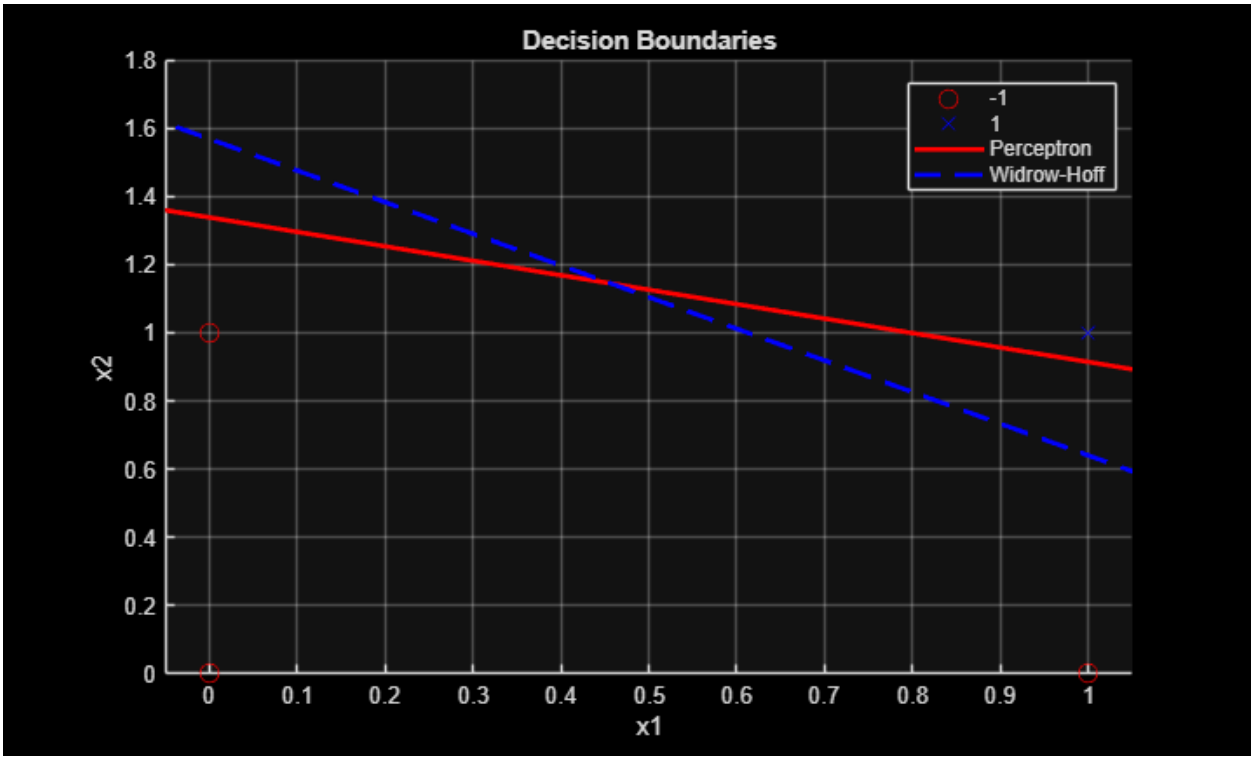
```
figure; hold on;
gscatter(X(:,1), X(:,2), T, 'rb', 'ox', 8);
x_vals = linspace(min(X(:,1))-1, max(X(:,1))+1, 100);

% Perceptron boundary
y_p = -(w_p(1)*x_vals + w_p(3)) / w_p(2);
```

```
plot(x_vals, y_p, 'r-', 'LineWidth', 2, 'DisplayName','Perceptron');

% Widrow-Hoff boundary
y_wh = -(w_wh(1)*x_vals + w_wh(3)) / w_wh(2);
plot(x_vals, y_wh, 'b--', 'LineWidth', 2, 'DisplayName','Widrow-Hoff');

xlabel('x1'); ylabel('x2'); title('Decision Boundaries');
legend; grid on;
```



Published with MATLAB® R2025a