Table of Contents


```
% FML_LAB_2
% Author: ARISH
% Q1) Discrete Perceptron Training Algorithm for 3D Patterns
clear; clc;

% Input Patterns
X = [ 0.8  0.5  0.0;
      0.9  0.7  0.3;
      1.0  0.8  0.5;
      0.0  0.2  0.3;
      0.2  0.1  1.3;
      0.2  0.7  0.8 ];

D = [1; 1; 1; -1; -1; -1];  % Desired outputs (class labels)

% Initialize
w = zeros(1,3);             % Initial weight vector
eta = 1;                    % Learning rate
max_iter = 100;

fprintf('\nQ1)');

% Training Loop
for iter = 1:max_iter
    error_count = 0;
    for i = 1:size(X,1)
        x_i = X(i,:);
        y = sign(dot(w, x_i));
        if y ~= D(i)
            w = w + eta * D(i) * x_i;
            error_count = error_count + 1;
        end
    end
    if error_count == 0
        break;
    end
end

% Output
fprintf('\nFinal weight vector:\n');
disp(w);
fprintf('Number of iterations: %d\n', iter);
```

Q1)

```
Final weight vector:
    1.5000    0.3000   -0.8000

Number of iterations: 3
```

Q2) Delta Rule (Widrow-Hoff) for Continuous Perceptron

```
%clear; clc;

% Input Patterns
X = [ 0.8  0.5  0.0;
      0.9  0.7  0.3;
      1.0  0.8  0.5;
      0.0  0.2  0.3;
      0.2  0.1  1.3;
      0.2  0.7  0.8 ];

D = [1; 1; 1; -1; -1; -1];  % Desired outputs

% Parameters
etas = [0.01, 0.1, 0.5, 1];  % Try different learning rates
max_iter = 100;
tolerance = 1e-3;

fprintf('\nQ2)');

% Training Loop for Each η
for e = 1:length(etas)
    eta = etas(e);
    w = zeros(1,3);          % Initial weights
    iter = 0;
    prev_w = w;

    for k = 1:max_iter
        iter = iter + 1;
        for i = 1:size(X,1)
            x_i = X(i,:);
            y = dot(w, x_i);          % Continuous output
            e_i = D(i) - y;           % Error
            w = w + eta * e_i * x_i; % Weight update
        end

        % Check convergence
        if norm(w - prev_w) < tolerance
            break;
        end
        prev_w = w;
    end

    % Output
    fprintf('\nLearning rate η = %.2f\n', eta);
```

```
    fprintf('Final weight vector: [%.4f %.4f %.4f]\n', w);
    fprintf('Iterations to converge: %d\n', iter);
end
```

Q2)
Learning rate η = 0.01
Final weight vector: [1.0298 0.1898 -0.9740]
Iterations to converge: 100

Learning rate η = 0.10
Final weight vector: [2.0457 -0.8109 -1.0555]
Iterations to converge: 100

Learning rate η = 0.50
Final weight vector: [2.2699 -1.0105 -0.9901]
Iterations to converge: 45

Learning rate η = 1.00
Final weight vector: [2.1809 -0.9900 -0.8978]
Iterations to converge: 13

Q3) Continuous Perceptron: Output = x1 + x2

Trained using Widrow-Hoff (Delta) Rule

```
%clear; clc;

% Training Data
% Inputs: [x1, x2]
X = [0 0;
     0 1;
     1 0;
     1 1;
     0.5 0.5;
     0.2 0.8];

% Targets: sum of inputs
T = sum(X, 2);  % T = x1 + x2

% Parameters
X_aug = [X ones(size(X,1),1)];  % Add bias term
w = randn(1,3);                 % Initial weights [w1 w2 bias]
eta = 0.1;                      % Learning rate
max_iter = 100;
tolerance = 1e-4;

fprintf('\nQ3');

% Training Loop
prev_w = w;
for iter = 1:max_iter
    for i = 1:size(X_aug,1)
        x_i = X_aug(i,:);
```

```
        y = dot(w, x_i);            % Continuous output
        e = T(i) - y;               % Error
        w = w + eta * e * x_i;      % Weight update
    end
    if norm(w - prev_w) < tolerance
        break;
    end
    prev_w = w;
end

% Output
fprintf('\nFinal weights: [%.4f %.4f %.4f]\n', w);
fprintf('Equation: y = %.4f·x1 + %.4f·x2 + %.4f\n', w(1), w(2), w(3));
fprintf('Iterations: %d\n', iter);

% Test the model
fprintf('\nTesting on training data:\n');
for i = 1:size(X,1)
    x_i = [X(i,:) 1];
    y = dot(w, x_i);
    fprintf('Input: [%.2f %.2f] → Output: %.4f (Target: %.2f)\n', X(i,1),
X(i,2), y, T(i));
end
```

Q3

```
Final weights: [0.9994 0.9992 0.0008]
Equation: y = 0.9994·x1 + 0.9992·x2 + 0.0008
Iterations: 98

Testing on training data:
Input: [0.00 0.00] → Output: 0.0008 (Target: 0.00)
Input: [0.00 1.00] → Output: 1.0001 (Target: 1.00)
Input: [1.00 0.00] → Output: 1.0002 (Target: 1.00)
Input: [1.00 1.00] → Output: 1.9994 (Target: 2.00)
Input: [0.50 0.50] → Output: 1.0001 (Target: 1.00)
Input: [0.20 0.80] → Output: 1.0001 (Target: 1.00)
```

Published with MATLAB® R2025a