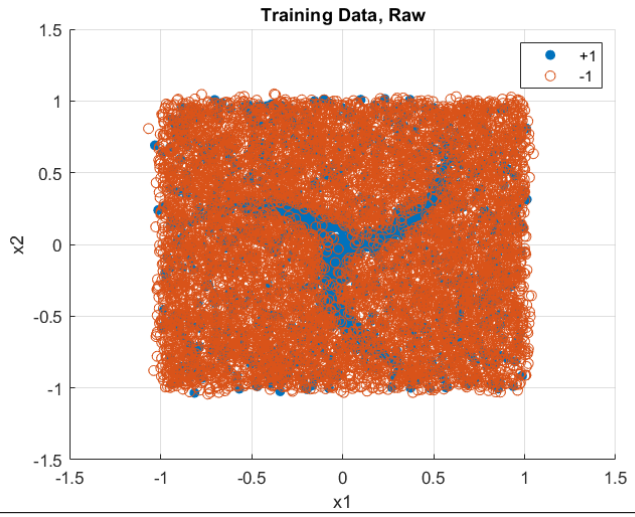
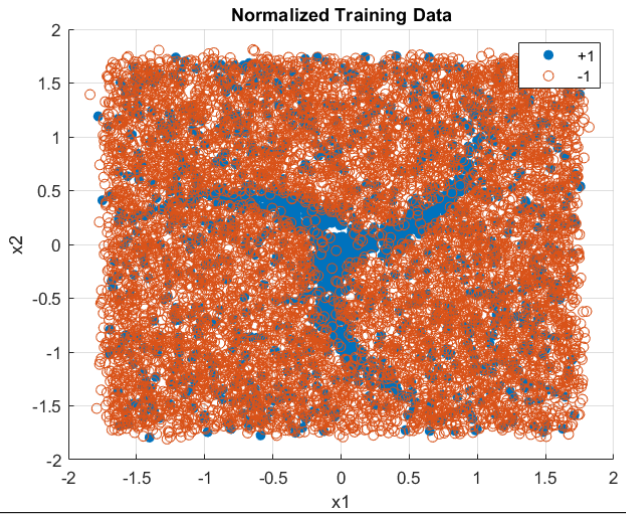
Load, normalize and plot data

```
%% Loading data
clear; clc;

% Load & normalize
trainingSet = readmatrix('training_set.csv');
trainingDataRaw = trainingSet(:, 1:2); % Only for plotting
trainingInput = [normalize(trainingSet(:, 1:2))];
t = trainingSet(:, 3);
validationSet = readmatrix('validation_set.csv');
validationInput = [normalize(validationSet(:, 1:2))];
tVal = validationSet(:, 3);

% Plot raw training data
figure;
hold on;
scatter(trainingDataRaw(t == 1, 1), trainingDataRaw(t == 1, 2), 'o', 'filled
    ', 'displayname', '+1');
scatter(trainingDataRaw(t == -1, 1), trainingDataRaw(t == -1, 2), 'o', '
    displayname', '-1');
title('Training Data, Raw');
xlabel('x1');
ylabel('x2');
legend('Location', 'northeast');
grid on;
hold off;

% Plot normalized training data
figure;
hold on;
scatter(trainingInput(t == 1, 1), trainingInput(t == 1, 2), 'o', 'filled', '
    DisplayName', '+1');
scatter(trainingInput(t == -1, 1), trainingInput(t == -1, 2), 'o', '
    DisplayName', '-1');
title('Normalized Training Data');
xlabel('x1');
ylabel('x2');
legend('Location', 'northeast');
grid on;
hold off;
```






```
%% Parameters
eta = 0.002;
M1 = 30;

w1 = randn(2, M1);
w2 = randn(1, M1);
t1 = zeros(1, M1);
t2 = 0;
```

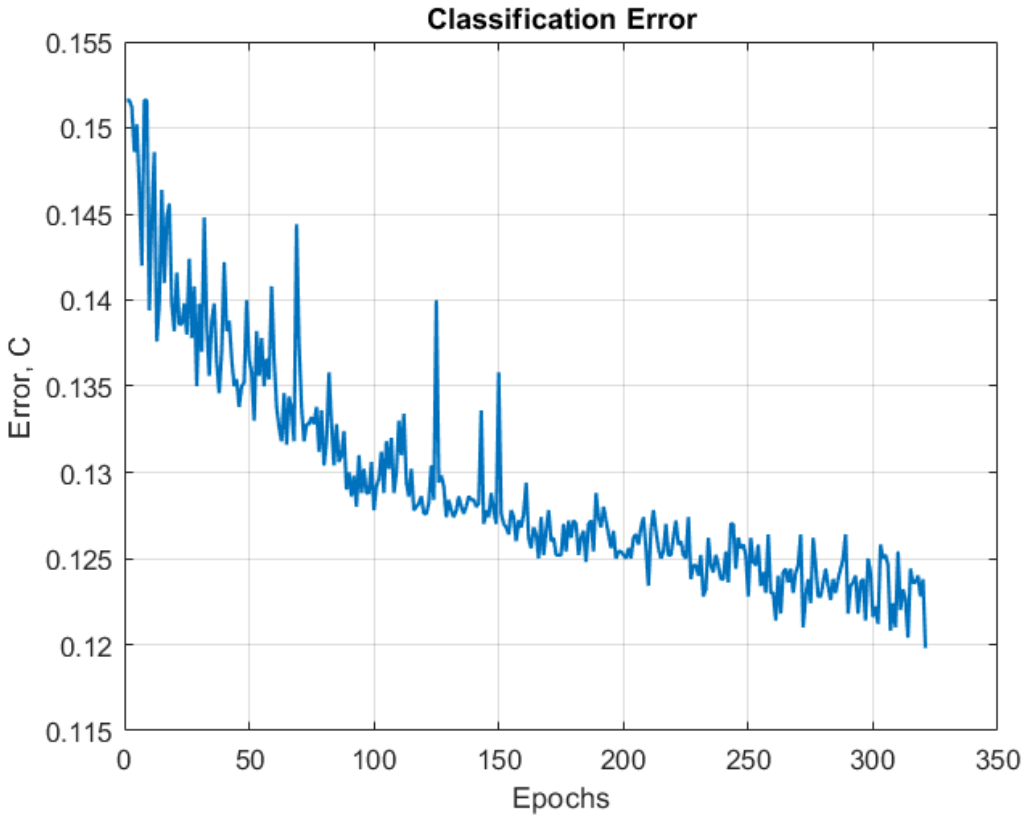
```
%% Plot random decision boundary
figure;
hold on;
[x1_grid, x2_grid] = meshgrid(-1:0.1:1, -1:0.1:1);
decision_inputs = [x1_grid(:), x2_grid(:)];
decision_V = tanh(-t1 + (decision_inputs * w1));
decision_O = tanh(-t2 + sum(w2 .* decision_V, 2));
scatter(trainingInput(t == 1, 1), trainingInput(t == 1, 2), 'o', 'filled', '
    DisplayName', '+1');
scatter(trainingInput(t == -1, 1), trainingInput(t == -1, 2), 'o', '
    DisplayName', '-1');
contour(x1_grid, x2_grid, reshape(decision_O, size(x1_grid)), [0 0], '
    LineWidth', 4, 'LineColor', 'g', 'DisplayName', 'Initial Decision
    Boundary');
title('Initial Random Decision Boundary');
xlabel('x1');
ylabel('x2');
legend('Location', 'northeast');
grid on;
hold off;
```

```
%% Training and validation
```

```
classificationErrors = zeros(1, 5000);
for epoch = 1:5000

    for idx = 1:length(trainingInput)
        % Random training pattern selection
        mu = randi(length(trainingInput));
        x = trainingInput(mu, :);

        % Feedforward
        V = tanh(-t1 + (x * w1));
        O = tanh(-t2 + sum(w2 * V'));

        % Backpropagation:
        % Networks output vs desired outputs deviation (Delta_m):
        outputError = (t(mu) - O) * (1 - tanh(-t2 + dot(w2, V))^2);
        % How much each neuron contributed to the deviation (delta_m):
        hiddenError = (w2' * outputError) .* (1 - tanh(-t1 + x * w1).^2)';

        % Update weights and thresholds
        w2 = w2 + eta * outputError * V;
        t2 = t2 - eta * outputError;
        w1 = w1 + eta * (hiddenError * x)';
        t1 = t1 - eta * hiddenError';
    end

    % Validation
    errorSum = 0;
    pVal = length(validationSet);
    for mu = 1:pVal
        x = validationInput(mu, :);
        V = tanh(-t1 + (x * w1));
        O = tanh(-t2 + sum(w2 * V'));
        errorSum = errorSum + abs(sign(O) - tVal(mu));
    end

    C = (1/(2*pVal)) * errorSum;
    classificationErrors(epoch) = C;

    % Stop if classification error < 12%
    if C < 0.12
        break;
    end
end
disp(['Last epoch: ' num2str(epoch) ', Classification error: C = ' num2str(C
   *100) '%']);
```

```
%% Saving files
% Saving in desired format
csvwrite('w1.csv', w1');
csvwrite('w2.csv', w2');
csvwrite('t1.csv', t1');
csvwrite('t2.csv', t2);
```

```
%% Plot error(epochs)
figure;
plot(classificationErrors(1:epoch), 'LineWidth', 1.3);
title('Classification Error');
xlabel('Epochs');
ylabel('Error, C');
grid on;
```



```
%% Plot decision boundary after training
figure;
hold on;
% Latest decision boundary after training
decision_V = tanh(-t1 + (decision_inputs * w1));
decision_O = tanh(-t2 + sum(w2 .* decision_V, 2));
scatter(trainingInput(t == 1, 1), trainingInput(t == 1, 2), 'o', 'filled', '
    DisplayName', '+1');
scatter(trainingInput(t == -1, 1), trainingInput(t == -1, 2), 'o', '
    DisplayName', '-1');
contour(x1_grid, x2_grid, reshape(decision_O, size(x1_grid)), [0 0], '
    LineWidth', 4, 'LineColor', 'g', 'DisplayName', 'Decision Boundary (After
     Training)');
title('Decision Boundary After Training');
xlabel('x1');
ylabel('x2');
legend('Location', 'northeast');
grid on;
hold off;
```