Homework 2: One-layer perceptron

$\underline{x}^{(\mu)} = [x_1^{(\mu)}, x_2^{(\mu)}]^T$

$t^{\mu} = \pm 1$

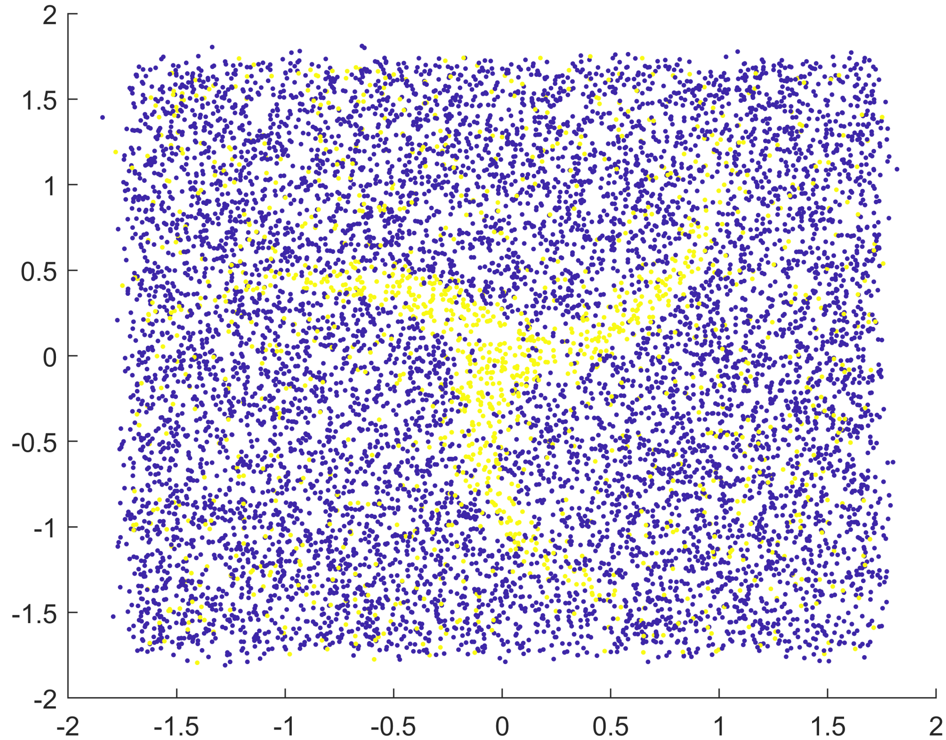
Start by normalizing the data by centering each of the two input components and normalizing their respective variances to 1.

Your task is to train a perceptron with one fully-connected hidden layer, two input terminals and one output unit. Use the following network layout:

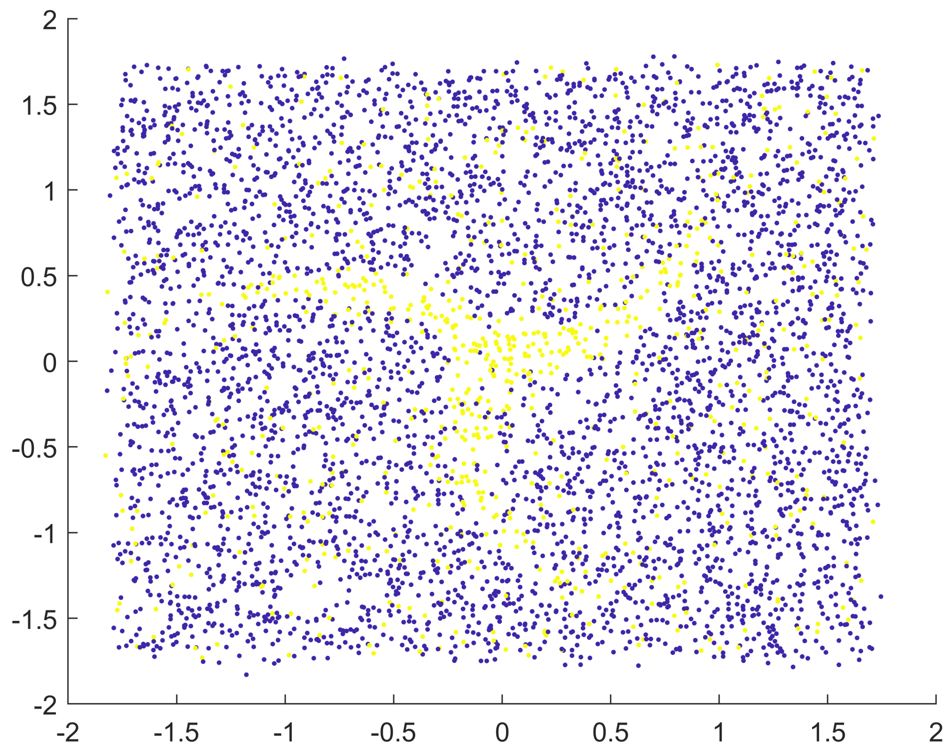
```
clc
clear
training_set = readmatrix("training_set.csv");
validation_set = readmatrix("validation_set.csv");
eta = 0.006;
N = 2;
epoch_n = 2000;
error_criteria = 0.12;
```

Preprocessing

```
train_input_raw = training_set(:,1:2);
train_input = normalize(train_input_raw)';      % [2x10000]
target = training_set(:,3);                      % [10000x1]
figure
scatter(train_input(1,:),train_input(2,:),[],target,'.');
```




```
p = size(train_input,2);

figure
validation_input_raw = validation_set(:,1:2);
validation_input = normalize(validation_input_raw)';
% validation_input = validation_set(:,1:2)';      % [2x5000]
validation_target = validation_set(:,3);        % [5000x1]
scatter(validation_input(1,:),validation_input(2,:),[],validation_target,'.')
```




```
p_val = size(validation_input,2);
```

Initialization

Different number of hidden neurons in the hidden layer.

```
M = 10;  % M goes to 100???
w = normrnd(0, 1/M, [M,2]);     % [Mx2] w_jk
theta = zeros([M,1]);           % [Mx1] theta_j
W = normrnd(0, 1/M, [1, M]);    % [1xM] W_ij i=1
Theta = 0;
V = zeros([M,1]);               % [Mx1] V_j
Out = 0;
H_train = [];
H_val = [];
```

Set energy function

$$H = \frac{1}{2}\sum_{\mu i}\left(t_i^{(\mu)} - O_i^{(\mu)}\right)^2 . \qquad (6.4)$$

```
H_train(end+1) = energy(W, w, Theta, theta, train_input, target);
H_val(end+1) = energy(W, w, Theta, theta, validation_input, validation_target);
```

Training

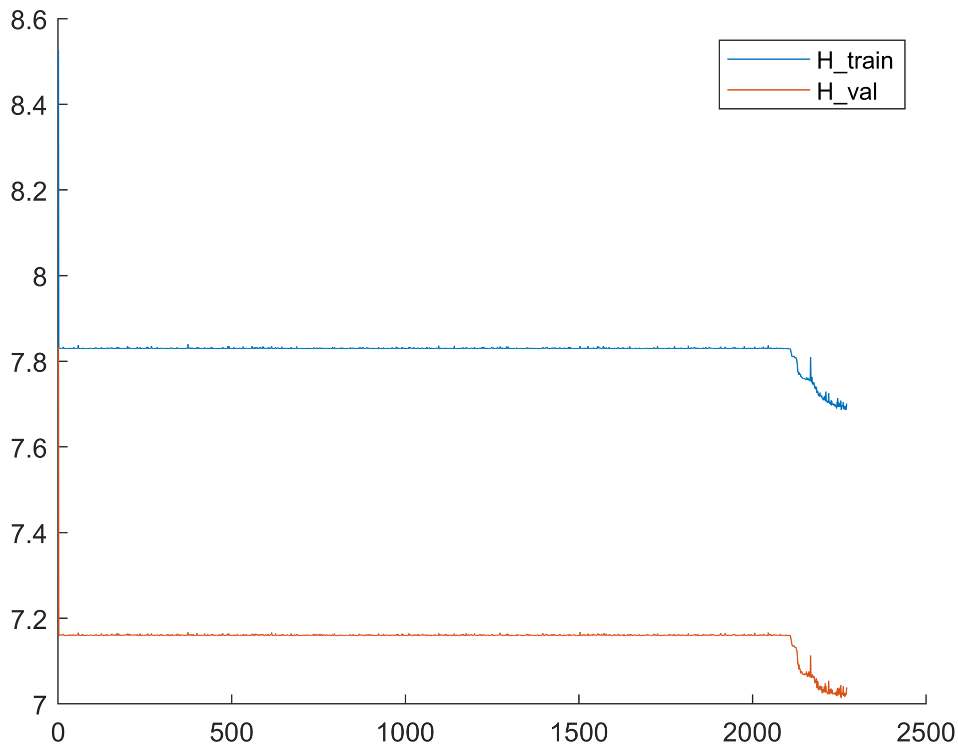
```
for ep=1:epoch_n
    for nu = 1 : p
        mu = randi([1,p]);
        x_mu = train_input(:,mu); % randomly pick an input from training set
        t_mu = target(mu);

        % propagate forward
        b_mu = w*x_mu-theta;    % [3x1]
        V = tanh(b_mu);         % activation function g = tanh()
        B_mu = W*V-Theta;  % scalar
        Out = tanh(B_mu);       % scalar
        % end of propagate forward

        % back propagation
        %% Calculate Delta
%         Delta = (target(mu) - Out) * eval(subs(diff(g), B_mu));
        Delta = (target(mu) - Out) * (1-tanh(B_mu)^2);

        %% Calculate delta
%         delta = Delta * W'.* eval(subs(diff(g), b_mu));
%         delta = Delta * W'.* (1-tanh(b_mu).^2);
        delta = Delta * W'.* (1-tanh(b_mu).^2);

        %% update W and w
        delta_w = eta * delta * x_mu';
        delta_W = eta * Delta * V';

        W = W + delta_W;
        w = w + delta_w;

        %% update Theta
        delta_Theta = -eta * Delta;
        Theta = Theta + delta_Theta;

        %% update theta
        delta_theta = -eta*delta;
        theta = theta + delta_theta;
        % end of back propagation
    end
    W;
    w;
    H_train(end+1) = energy(W, w, Theta, theta, train_input, target);
    H_val(end+1) = energy(W, w, Theta, theta, validation_input, validation_target);

%     plot(1:length(H_train),log(H_train))
%     plot(1:length(H_val),log(H_val))
%     legend('H\_train','H\_val')
```

```
    b = w*validation_input-theta;
    V = tanh(b);
    B = W*V-Theta;
%     C = 1/(2*p_val)*sum(abs(validation_target'-sign(B)));
    tmp = 0;
    for mu = 1:p_val
        m = randi(p_val);
%         m = mu;
        x_mu = validation_input(:,m);
        b_mu = w * x_mu - theta;    % [Mx1]
        V = tanh(b_mu);          % activation function g = tanh()
        B_mu = W * V - Theta;  % scalar
        Out = tanh(B_mu);       % scalar
        tmp = tmp + abs(validation_target(m)-sign(Out));
    end
    C = 1/(2*p_val)*tmp;
    if C < error_criteria
        disp('Classification error criteria is met');
        disp(['Epoch: ',num2str(ep),' C: ',num2str(C)])
        break
    else
        disp(['Epoch: ',num2str(ep),' C: ',num2str(C)])
    end
end
```

```
Classification error criteria is met
Epoch: 270 C: 0.1184
```

Plot H

```
figure
hold on
plot(1:length(H_train),log(H_train))
plot(1:length(H_val),log(H_val))
legend('H\_train','H\_val')
hold off
```




```
w
```

```
w = 10×2
   -1.5759    2.0822
   -0.3253   -0.1967
    3.5666    2.1493
   -4.0089    5.1979
    4.1902    1.0640
    0.6088    0.2444
   -0.4945   -4.6993
    0.3614    0.0382
    0.6054    1.6781
    2.7554   -1.4534
```

```
W
```

```
W = 1×10
    1.1119   -0.2276   -2.0984   -1.9447    1.6450    0.5558   -2.5909    0.2866 ···
```

```
theta
```

```
theta = 10×1
   -0.3132
   -0.0148
   -0.0026
    0.0715
   -1.4094
   -0.1113
    0.5144
   -0.0027
   -1.0124
```

```
1.7783
```

```
Theta
```

```
Theta = 2.8069
```

```
csvwrite("w1.csv",w)
csvwrite("w2.csv",W')
csvwrite("t1.csv",theta)
csvwrite("t2.csv", Theta)
```

```
function Out = fd_prop(W, w, Theta, theta, x)
    b_mu = w * x - theta;    % [Mx1]
    V = tanh(b_mu);         % activation function g = tanh()
    B_mu = W * V - Theta;  % scalar
    Out = tanh(B_mu);       % scalar
end

function H = energy(W, w, Theta, theta, input_set, target_set)
    H = 0;
    for mu = 1:size(input_set, 2)
        Out = fd_prop(W,w,Theta,theta,input_set(:,mu));
        H = H + (target_set(mu)-Out)^2;
    end
    H = H * 0.5;
end
```