```
%% One layer perceptr
clc
% Settings %%%%%%%%%%%%%%%%%%%%
M1 = 20;
learningrate = 0.005;
maxEpochs = 3000;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Read and process data
readTraining = readmatrix('training_set.csv');
readValidation = readmatrix('validation_set.csv');

meanData = mean([readTraining; readValidation]);
stdData = std([readTraining; readValidation]);

training = [(readTraining(:,1)-meanData(1))/stdData(1), (readTraining(:,2)-meanData↙
(2))/stdData(2), readTraining(:,3)];
validation = [(readValidation(:,1)-meanData(1))/stdData(1), (readValidation(:,2)-↙
meanData(2))/stdData(2), readValidation(:,3)];

% Initializing
w = normrnd(0, 1, [M1, 2]);
W = normrnd(0, 1, [M1, 1]);
theta = zeros(M1,1);
THETA = zeros(1,1);
V = zeros(1,M1);
numberOfInputs = 2;
C = 1;
nEpoch = 1;


% Train
while (C >= 0.118 && nEpoch < maxEpochs)
    C=0;
    for i = 1:size(training, 1)
        r = randi(size(training,1));
        V = tanh(w * training(r,1:2)'-theta);
        O = tanh(W'*V - THETA);

        % Gradiant descent
        delta1 = (training(r,3)-O) .* (1 - tanh( W' *V-THETA ).^2);
        delta = (W.*delta1) .* (1 - tanh(w*training(r,1:2)'-theta ).^2);

        W = W + learningrate * delta1 * V;
        w = w + learningrate * delta * training(r,1:2);

        THETA = THETA -learningrate*delta1;
        theta = theta - learningrate*delta;
    end

    for k = 1:length(validation)
        V = tanh(w * validation(k,1:2)'-theta);
        O = tanh(W'*V - THETA);
        C = C + abs(sign(O)-validation(k,3));
    end
    C = C/(2*size(validation,1))
    nEpoch = nEpoch + 1
end

% Create CSV
csvwrite('w1.csv',w)
csvwrite('w2.csv',W)
csvwrite('t1.csv',theta)
csvwrite('t2.csv',THETA)
```