Home Assignment 2 - FFR135 Artificial Neural Networks

Table of Contents

Code for Homework 2, Linear separability of 4-dimensional Boolean functions(2020)

```
% Author: Arshad Nowsath
clc, clear all

% Loading the given training and validation data from OpenTA
training_data = csvread("input_data_numeric.csv");
training_data = training_data(:,2:end);

% Targets
E = [1, 1, 1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, -1, -1];
F = [-1, -1, -1, 1, 1, -1, 1, 1, 1, 1, 1, -1, -1, 1, 1, 1];
B = [-1, 1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1];
A = [-1, -1, 1, 1, 1, -1, 1, -1, -1, 1, -1, -1, -1, -1, 1, -1];
C = [-1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, -1, 1, -1, -1];
D = [1, 1, 1, 1, -1, -1, -1, -1, -1, 1, -1, -1, 1, -1, -1, -1];

targets = [E' F' B' A' C' D'];

% Parameters
learning_rate = 0.02;
iteration = 10^5;
tries = 10;
Output = zeros(size(targets));
linearly_separable = zeros(size(targets,2),1);

% loop for training the perceptron
for i = 1:size(targets,2)
    t = targets(:,i);

    for k = 1:tries
        W = -0.02 + rand(4,1)*(0.02-(-0.02)); % Weight Iitialization
        theta = -1 + rand*(1-(-1));
        for j = 1:iteration
            % Forward propagation
            b = sum(-theta + training_data*W);
            output =  tanh(0.5*(-theta + training_data*W));
            % Stochastic gradient descent
```

```
            mu = randi([1,size(targets,1)]);
            d_w = learning_rate*(t(mu)-output(mu))*(1-
tanh(b*0.5).^2)*0.5* training_data(mu,:)';
            d_theta = -learning_rate*(t(mu)-output(mu))*(1-
tanh(b*0.5).^2)*0.5;
            W = W + d_w;
            theta = theta + d_theta;
            % IF to check whether the linearly seperable break or not
            if sign(tanh(0.5*(-theta + training_data*W))) == t
                break
            end
        end
        if sign(tanh(0.5*(-theta + training_data*W))) == t
            break
        end
    end

    Output(:,i) = sign(tanh(0.5*(-theta + training_data*W)));
    if isequal(Output(:,i),t)
        linearly_separable(i) = true;
    else
        linearly_separable(i) = false;
    end
end
% To print which target is linearly seperable
linearly_separable
```

```
linearly_separable =

     1
     1
     1
     0
     0
     0
```

Code for Homework 2, Two-layer perceptron(2020)

Author: Arshad Nowsath

```
clc, clear all

% Loading the given training and validation data from OpenTA
training_data = csvread('training_set.csv');
validation_data = csvread('validation_set.csv');

% Splitting the data for training and validation set
training_x = training_data(:,1:2);   % contains input patterns 1&2
 columns
training_y = training_data(:,3);     % contains targets 3 columns
```

```
validation_x = validation_data(:,1:2);
validation_y = validation_data(:,3);

% Parameters
M1 = 10;     % M1 and M2 are given by random experimental values
M2 = 4;
learning_rate = 0.01;
epoch = 1000;
classification_error_criteria = 0.12;   % C is below 12%

% Intialize the weight with some Random Gaussian number
W1 = normrnd(0,1,[M1,2]);
W2 = normrnd(0,1,[M2,M1]);
W3 = normrnd(0,1,[1,M2]);

% Setting the thresholds to zero
theta_1 = zeros(1,M1);
theta_2 = zeros(1,M2);
theta_3 = zeros(1,1);

% loop for training the perceptron

for i = 1:epoch
    for j = 1:length(training_x)

        mu = randi([1,length(training_y)]);
        % forward propagation
        V1 = tanh(-theta_1' + W1*training_x(mu,:)');
        V2 = tanh(-theta_2' + W2*V1);
        Output = tanh(-theta_3 + W3*V2);

        % Backward propagation
        delta_3 = (training_y(mu)-Output)*(1-Output.^2);
        delta_2 = delta_3 * W3.*(1-V2.^2)';
        delta_1 = delta_2*W2.*(1-V1.^2)';

        % weights Update
        W3 = W3 + learning_rate * delta_3*V2';
        W2 = W2 + learning_rate * (V1*delta_2)';
        W1 = W1 + learning_rate * delta_1'.*training_x(mu,:);

        % Bias Update
        theta_1 = theta_1 - learning_rate * delta_1;
        theta_2 = theta_2 - learning_rate * delta_2;
        theta_3 = theta_3 - learning_rate * delta_3;
    end

    % Compute the classification error from the validation set
    V1_validation = tanh(-theta_1 + (W1*validation_x')');
    V2_validation = tanh(-theta_2 + (W2*V1_validation')');
    Output_validation = tanh(-theta_3 + (W3*V2_validation')');

    % classification error
```

```
    C = (1/(2*length(validation_y)))*sum(abs(sign(Output_validation)-
validation_y));

    % Break if the classification error is met
    if C < classification_error_criteria
        disp(['Epoch: ',num2str(i),' C: ',num2str(C)])
        disp('Classification error criteria is met');
        break
    else
        disp(['Epoch: ',num2str(i),' C: ',num2str(C)])
    end
end

% Export  weight matrices(W1,W2), vectors(W3) and thresholds(theta) to
 CSV format

% Weight Matrices
csvwrite('w1.csv',W1);
csvwrite('w2.csv',W2);

% Weight Vectors
csvwrite('w3.csv',W3);

%Thresholds
csvwrite('t1.csv',theta_1);
csvwrite('t2.csv',theta_2);
csvwrite('t3.csv',theta_3);
```

```
Epoch: 1 C: 0.1516
Epoch: 2 C: 0.144
Epoch: 3 C: 0.1486
Epoch: 4 C: 0.1372
Epoch: 5 C: 0.1362
Epoch: 6 C: 0.1352
Epoch: 7 C: 0.1356
Epoch: 8 C: 0.143
Epoch: 9 C: 0.1388
Epoch: 10 C: 0.133
Epoch: 11 C: 0.1304
Epoch: 12 C: 0.1288
Epoch: 13 C: 0.1334
Epoch: 14 C: 0.1276
Epoch: 15 C: 0.1326
Epoch: 16 C: 0.1322
Epoch: 17 C: 0.132
Epoch: 18 C: 0.1304
Epoch: 19 C: 0.124
Epoch: 20 C: 0.1248
Epoch: 21 C: 0.1236
Epoch: 22 C: 0.1242
Epoch: 23 C: 0.124
Epoch: 24 C: 0.1374
Epoch: 25 C: 0.1278
Epoch: 26 C: 0.124
```

```
Epoch: 27 C: 0.1224
Epoch: 28 C: 0.127
Epoch: 29 C: 0.1228
Epoch: 30 C: 0.1234
Epoch: 31 C: 0.1224
Epoch: 32 C: 0.1216
Epoch: 33 C: 0.123
Epoch: 34 C: 0.1216
Epoch: 35 C: 0.1198
Classification error criteria is met
```

Published with MATLAB® R2020a