Lab Assignement 1

Karamoulas Eleftherios - S3261859
Tzafos Panagiotis - S3302148

March 15, 2017

1 Answers for Lab 1 sections 5,6

5.a The error is not decreasing in each epoch because in some iterations with the weight changes the output is the desired one.

5.b Because in the case of goal/output 1/0 and 0/1 even though the error should be the same we have 1 and -1 thats why we use the square.

5.c Because the required iterations to reach 0 depend on the random starting values and the learn rate that we use.

5.d With learning rate 0.6 we observe that more epochs are required to reach 0 error. Using higher learning rate is not always good. The learning rate that should be used depends on the training set.

5.e The TLU is still capable of learning the AND-function after both changes however when we have 0.2 and 0.8 in some cases it needs more epochs to reach 0 error.

5.f In sub-question 5.e we encountered with Resilience to noise our artificial neural network checks if the weighted summation of the inputs is above or below the threshold no matter the difference so small changes in the inputs or the weights don't lead to changes.

5.g TLU can learn the NAND-function however both weights and threshold become negative because the NAND is exactly the opposite of AND so the weights and the threshold are from the other side.

2 Code for TLU

(a) The code starts by initializing the variables like learn rate, nepochs and the tables that are going to be used later like examples, goal, weights, threshold and others that store information about our learning procedure. Then starts the first iteration of how many times our learning algorithm will run and the first values of weight and threshold are stored and then the second iteration for every possible combination of inputs. Then we calculate the weighted summation and after we subtract our threshold from it to find out which will be the output value. Next we check if the output is the desirable and if not we calculate the new weights and threshold. The next step is the computation of the error and the delta values for the weights and threshold. Before plotting the results we update the weights and threshold for the next iteration and store the error values and their mean. Finally we create two plots one for the errors and one for the evolvement of weights and threshold through time.

Listing 1: tlu.m

```
% Clear workspace and close all previous windows
clear all;
close all;

% Initializing data and parameters
% PARAMETERS
n_examples = 2;              % The number of examples(0 < n_examples < 7)
n_epochs = 12;                % The number of epochs
normalize_weights = true;    % Normalization bool

random_percentage = 50;      % Percentage of bits that are flipped randomly
invert = false;              % Invert the input (test for spurious states)


% Do not change these lines
dim_x = 5;                   % Dimensions of examples
dim_y = 5;

% Compute size of examples
size_examples = dim_x * dim_y;

% Convert percentage to fraction
random_percentage = random_percentage/100;

% Set color for network plots
color = 20;

% The data is stored in .dat files. They have to be located in the same
% directory as this source file
data = importdata('M.dat');
data(:,:,2) = importdata('A.dat');
data(:,:,3) = importdata('S.dat');
data(:,:,4) = importdata('T.dat');
data(:,:,5) = importdata('E.dat');
data(:,:,6) = importdata('R.dat');

% Convert data matrices into row vectors. Store all vectors in a matrix
vector_data = zeros(n_examples,size_examples);
for idx = 1:n_examples
    % Every row will represent an example
    vector_data(idx,:) = reshape(data(:,:,idx)',1,size_examples);
end


% TRAINING THE NETWORK
% The network is trained using one-shot Hebbian learning

% The result should be a matrix dimensions: size_examples * size_examples
% Each entry should contain a sum of n_examples
weights = vector_data' * vector_data;

% A hopfield neuron is not connected to itself. The diagonal of the matrix
```

```
% should be zero.
weights(logical(eye(size(weights)))) = 0;

% These lines check whether the matrix is a valid weight matrix for a
% Hopfield network.
assert(isequal(size(weights),[size_examples size_examples]), ...
    'The matrix dimensions are invalid');
assert(isequal(tril(weights)',triu(weights)), ...
    'The matrix is not symmetric');
assert(isempty(find(diag(weights), 1)), ...
    'Some neurons are connected to themselves');

% Normalizing the weights
if normalize_weights
    weights = weights ./ n_examples;
end


% PLOT WEIGHT MATRIX
figure(1)
imagesc(weights)
colorbar
title('Weight matrix')

% INTRODUCE NOISE

% Copy the input data
input = vector_data;

% Create a matrix with the same dimensions as the input data in which
% random_percentage elements are set to -1 and the others are set to 1.
% We do this by sampling from a normal distribution
noise_matrix = (randn(size(input)) > norminv(random_percentage));
noise_matrix = noise_matrix - (noise_matrix==0);

% Flip bits (* -1) using the noise_matrix
input = input .* noise_matrix;

% Optionally invert the input
if invert
    input = -1 .* input;
end

% PLOTTING INPUT PATTERNS
figure(2)
for example = 1:n_examples
    subplot(n_epochs + 2,n_examples,example)
    test = reshape((input(example,:)),dim_x, dim_y)';
    image(test .* color + color)
    str = 'Ex:';
    str = strcat(str,int2str(example));
    title(str)
    if(example == 1)
```

```
        axis on
        ylabel('input')
    else
        axis off
    end
    axis square
end

% UPDATING THE NETWORK
% Feed the network with all of the acquired inputs. Update the network and
% plot the activation after each epoch.
for example = 1:n_examples

    % The initial activation is the row vector of the current example.
    activation = input(example,:)';

    for epoch = 1:n_epochs
        % Compute the new activation
        activation = weights * activation;

        % Apply the activation function
        %activation = activation_function(activation);
        activation = sigmoid(activation);
        % PLOTTING THE ACTIVATION

        % Reshape the activation such that we get a 5x5 matrix
        output = reshape(activation, dim_x, dim_y)';

        % Compute the index of where to plot
        idx = epoch * n_examples + example;

        % Create the plot
        subplot(n_epochs + 2,n_examples,idx)
        image(output .* color + epoch   + color  )

        % Only draw axes on the leftmost column
        if(example == 1)
            axis on
            str = 'Ep:';
            str = strcat(str,int2str(epoch));
            ylabel(str)
        else
            axis off
        end

        % Make sure the plots use a square grid
        axis square
    end
end

% Finally we plot the goal vector for comparison
for idx = 1:n_examples
    subplot(n_epochs + 2,n_examples,(n_epochs + 1) * n_examples + idx);
```

```
    image(data(:,:,idx).* color + n_epochs+1  + color)
    if(idx == 1)
       axis on
       ylabel('goal')
    else
        axis off
    end
    axis square
end
%-------------------------------------------------------------------------
```

Listing 2: tlu.m

```
function [output] = activation_function(x)
    % Define the sigmoid function here
    for i=1:size(x)
        if x(i)>=0
            x(i) = 1;
        else
            x(i)= -1;
        end
    end
    [output] = x;
end
```