Lab Assignement 1

Karamoulas Eleftherios - S3261859
Tzafos Panagiotis - S3302148

February 23, 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.

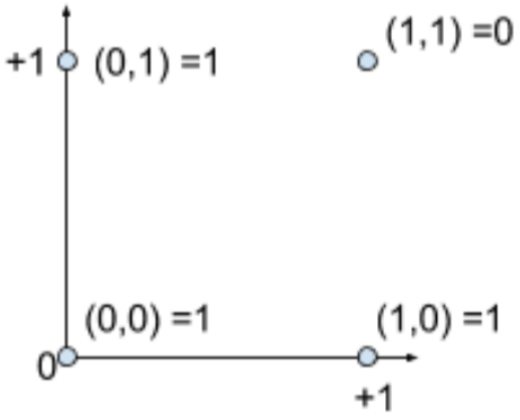

Figure 1: NAND

6 We observe that the weights, threshold and errors change all the time because the TLU cant solve nonlinear separable problems and XOR is one of them because of the positions that 1 and 0 have in the coordinate system we cant a line that separates the outputs in two planes.

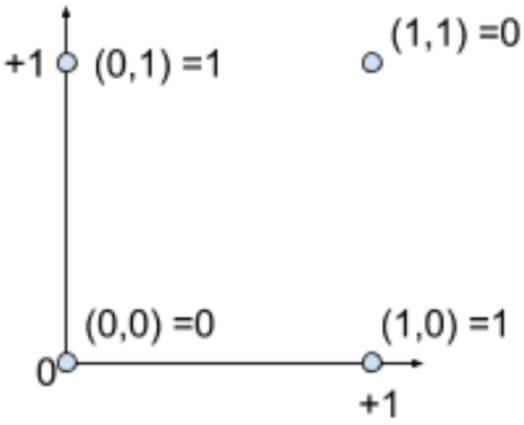

Figure 2: XOR

2 Code for TLU

1. 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

```
% TLU implementation
% Tzafos Panagiotis
% Karamoulas Eleftherios

% Parameters
learn_rate = 0.1;     % the learning rate
n_epochs = 100;       % the number of epochs we want to train

% Define the inputs
examples = [0,0;0,1;1,0;1,1];

% Define the corresponding target outputs
goal = [0;0;0;1];

% Initialize the weights and the threshold
weights = rand(1,2);
```

```
threshold = rand();

% Preallocate vectors for efficiency. They are used to log your data
% The 'h' is for history
h_error = zeros(n_epochs,1);
h_weights = zeros(n_epochs,2);
h_threshold = zeros(n_epochs,1);

% Store number of examples and number of inputs per example
n_examples = size(examples,1);      % The number of input patterns
n_inputs = size(examples,2);        % The number of inputs

for epoch = 1:n_epochs
    epoch_error = zeros(n_examples,1);

    h_weights(epoch,:) = weights;
    h_threshold(epoch) = threshold;

    for pattern = 1:n_examples

        % Initialize weighted sum of inputs
        summed_input = weights([1],[1]) * examples([pattern],...
            [1])+weights([1],[2]) * examples([pattern],[2]);
        % Subtract threshold from weighted sum
        a = summed_input - threshold;
        % Compute output
        if(a>=0)%if the value of a is possitive or equal
                %to zero then we have output=1
            output = 1;
        else%otherwise output is zero
            output = 0;
        end

        if(output~=goal([pattern]))%if the output isnt the same with
            %our goal we have to compute new weights threshold and error.
            new_weights = weights + learn_rate*...
                (goal([pattern])-output).*examples(pattern,:);
            new_threshold = threshold + learn_rate*...
                (goal([pattern])-output)*(-1);

        else %otherwise we keep the old values and error is zero
            new_weights = weights;
            new_threshold = threshold;

        end
        % Compute error
        error = (goal([pattern])-summed_input)^2;
        % Compute delta rule
        delta_weights = weights + learn_rate*...
            (goal([pattern])-summed_input).*examples(pattern,:);
        delta_threshold = threshold + learn_rate*...
            (goal([pattern])-summed_input)*(-1);
```

```
        % Update weights and threshold
        weights = new_weights;
        threshold = new_threshold;

        % Store squared error
        epoch_error(pattern) = error;%.^2 we have already
        %                     computed the squared error.
    end

    h_error(epoch) = sum(epoch_error)/4;%computation of the
                            %mean from the error paterns
end

%Plot functions
figure(1);
plot(1:n_epochs,h_error(:,1))
title('\textbf{TLU-error over epochs}', 'interpreter', 'latex',...
    'fontsize', 12);
xlabel('\# of epochs', 'interpreter', 'latex', 'fontsize', 12)
ylabel('Summed Squared Error', 'interpreter', 'latex', 'fontsize', 12)

figure(2);
plot(1:n_epochs,h_weights(:,1),'r-','DisplayName','weight 1')
hold on
plot(1:n_epochs,h_weights(:,2),'b-','DisplayName','weight 2')
plot(1:n_epochs,h_threshold,'k-','DisplayName','threshold')
xlabel('\# of epochs', 'interpreter', 'latex', 'fontsize', 12)
title('\textbf{Weight vector and threshold vs epochs}',...
    'interpreter', 'latex', 'fontsize', 12);
h = legend('location','NorthEast');
set(h, 'interpreter', 'latex', 'fontsize', 12);
hold off
```