Masters in Computer Science

Topics in Machine Learning & Neural Net

(COMP-5011)

Name:

Jaykumar Nariya (1116571)

Assignment2 Part 1:

1.(5 points) Implement the Least-Mean-Square algorithm to deepen the understanding of the single layer network.

Code:

```
tic
clear all;clc; % clear output
seed=2e5;
rand('seed',seed);
%============================== Data Prepro-
cessing=============================================

Data = readtable('sonar.txt');% https://archive.ics.uci.edu/ml/datasets/Con-
nectionist+Bench+(Sonar,+Mines+vs.+Rocks)
Data = Data(randperm(size(Data,1)),:); % Shuffle dataset
Data = table2cell(Data);
for i=1:208, % Give ALphabate value to integer
    Data(i,61) = cellfun(@double,Data(i,61),'uni',0);
end
Data = cell2table(Data);
for i=1:208, % Give Replace Label R with 1 and M with -1
    if table2array(Data(i, 61)) == 82,
        Data{i,61} = -1;
    else
        Data{i,61} = 1;
    end
end
Data = Data{:,:};
%============================== Variable Declara-
tion============================================
n = 0.1;
t = 1E-8; % weight changes threshold
weight = rand(60,1)./2  - 0.25; % Initial weights
train_data = Data(1:125,1:61); % Taking 60% data as trainting
train_data =train_data';
test_data = Data(126:208,1:61); % Taking 40% data as testinging
test_data = test_data';
Data = Data';
%============================Training LMS algorithm using generated
data============================
for epoch = 1:50, % No of Epoch is 50
    miss = 0;

    if epoch > 10, % learining rate after 10 epochs
        n = 0.01;
    end
    if epoch > 30, % learning rate after 30 epochs
        n = 0.001;
    end
    for i = 1:125, % for one epoch for no of instance 125
        X_train = [train_data(1:60,i)]; % getting input X training data from
dataset
        d = train_data(61,i); % getting true label from dataset
```

```
error = d-weight'*X_train; % find error e = ( d - w'x)
        weight_delta = n*X_train*error; % ?w = n*x*e
        er_v(i)= error;

         if (norm(weight_delta) < t) && (i >= 90) % check if NN Got sta-
balized or not
                fprintf('   W got stable sample %d of dataset\n',i);
                break;
         end

        weight = weight + weight_delta; %Wnew = w + ?w

    end



    mse(epoch) = mean(er_v.^2); % FInd RMSE for rach epoch
    fprintf(' For epoch %f Training rmse is %d \n ',epoch,mse(epoch));

end

trainingtime = toc;
bias = 0;
weight = [bias;weight]; % trained weights

%====================Training Accuracy get-
ing==================================
miss=0;
for i = 1 : 125, % for 125 samples trainting LMS algorithm
    X_train = [1;train_data(1:60,i)]; % getting input X training data from
dataset
    Y_test(i) = sigmoid(weight'*X_train);% find predicted value Y = sig(W.X)
    if (Y_test(i) - train_data(61,i)) ~= 0, % Find error using ( Y pred - Y
true)
        miss = miss + 1;
    end

end


Accuracy = ((125-miss)/125)*100;
fprintf(' Training Accuracy is %f \n', Accuracy); % Training Accuracy
fprintf(' Training Error is %f \n', 100-Accuracy);
fprintf(' Training Time is %f \n', trainingtime);
```

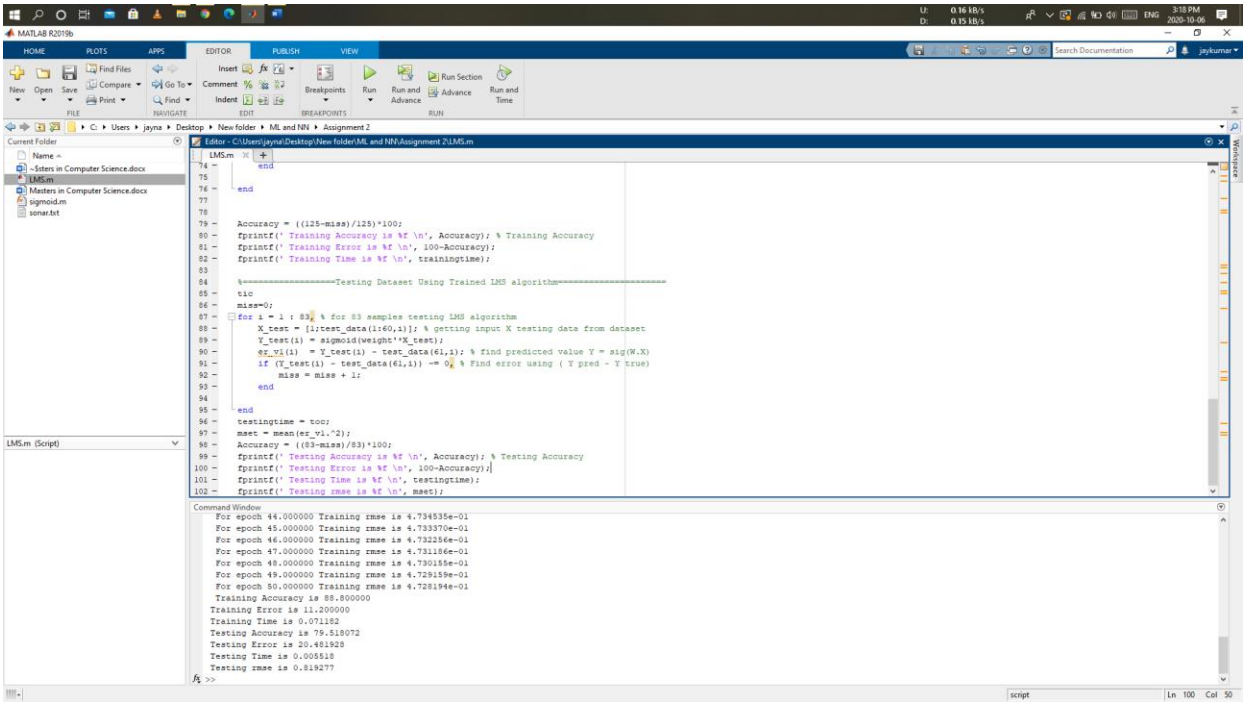
```
%=================Testing Dataset Using Trained LMS algo-
rithm====================
tic
miss=0;
for i = 1 : 83, % for 83 samples testing LMS algorithm
    X_test = [1;test_data(1:60,i)]; % getting input X testing data from da-
taset
    Y_test(i) = sigmoid(weight'*X_test);
    er_v1(i)  = Y_test(i) - test_data(61,i); % find predicted value Y =
sig(W.X)
    if (Y_test(i) - test_data(61,i)) ~= 0, % Find error using ( Y pred - Y
true)
        miss = miss + 1;
    end

end
testingtime = toc;
mset = mean(er_v1.^2);
Accuracy = ((83-miss)/83)*100;
fprintf(' Testing Accuracy is %f \n', Accuracy); % Testing Accuracy
fprintf(' Testing Error is %f \n', 100-Accuracy);
fprintf(' Testing Time is %f \n', testingtime);
fprintf(' Testing rmse is %f \n', mset);
```

Results:

- **Training Accuracy: 88.80%**
- **Training Error: 11.20%**
- **Training RMSE: 0.47**
- **Training Time: 0.07s**
- **Testing Accuracy: 79.52%**
- **Testing Error: 20.48%**
- **Testing RMSE: 0.81**
- **Testing Time: 0.005s**

Outputs:



```
Command Window
   For epoch 44.000000 Training rmse is 4.734535e-01
   For epoch 45.000000 Training rmse is 4.733370e-01
   For epoch 46.000000 Training rmse is 4.732256e-01
   For epoch 47.000000 Training rmse is 4.731186e-01
   For epoch 48.000000 Training rmse is 4.730155e-01
   For epoch 49.000000 Training rmse is 4.729159e-01
   For epoch 50.000000 Training rmse is 4.728194e-01
   Training Accuracy is 88.800000
Training Error is 11.200000
Training Time is 0.071182
Testing Accuracy is 79.518072
Testing Error is 20.481928
Testing Time is 0.005518
Testing rmse is 0.819277
fx >>
```

- Like the perceptron learning rule, the least mean square error (LMS) algorithm is an example of supervised training, in which the learning rule is provided with a set of examples of desired network behavior.
- perceptron is unstable and bounces around if dataset is not separable.
- In LMS error signal is determined by cost function while in Perceptron is deference between designed output and true output.
- LMS developed with adaptive filters.
- In the LMS algorithm the error is feed back into the algorithm after each sample is input and the weights are adjusted.
- The LMS algorithm adjusts the weights and biases of the Adaptive linear neuron so as to minimize this mean square error.

Reference:

- https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)