Masters in Computer Science

Topics in Machine Learning & Neural Net

(COMP-5011)

Name:

Jaykumar Nariya (1116571)

Assignment2 Part 2:

1. **(5 points) Write a python or MATLAB based codes for the Incremental Extreme Learning Machine. Students could use the codes of Fixed Extreme Learning Machine as their reference. Test the written code by using two datasets (one classification dataset and one UCI regression dataset).**
2. **(2 points) Students need to report their testing accuracy/RMSE, training accuracy/RMSE, training time and testing time for each of the two datasets.**
3. **(3 points) It is required to use one of the three classification datasets (CIFAR10, CIFAR100, and MNIST) to complete the assignment.**

Code

```
function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
i_elm(TrainingData_File, TestingData_File, Elm_Type, NumberofHiddenNeurons,
ActivationFunction)



%%%%%%%%%%% Macro definition
REGRESSION=0;
CLASSIFIER=1;

%%%%%%%%%%% Load training dataset
train_data=TrainingData_File;
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                                   %   Release raw training
data array

%%%%%%%%%%% Load testing dataset
test_data=TestingData_File;
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                                    %   Release raw testing
data array

NumberofTrainingData=size(P,2);
NumberofTestingData=size(TV.P,2);
NumberofInputNeurons=size(P,1);

if Elm_Type~=REGRESSION
    %%%%%%%%%%%% Preprocessing the data of classification
    sorted_target=sort(cat(2,T,TV.T),2);
    label=zeros(1, 1);                               %   Find and save in
'label' class label from training and testing data sets
    label(1,1)=sorted_target(1,1);
    j=1;
    for i = 2:(NumberofTrainingData+NumberofTestingData)
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
    number_class=j;
    NumberofOutputNeurons=number_class;

    %%%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;
```

```
    %%%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(1,j) == TV.T(1,i)
                break;
            end
        end
        temp_TV_T(j,i)=1;
    end
    TV.T=temp_TV_T*2-1;

end                                                %   end if of Elm_Type

%%%%%%%%%%% Calculate weights & biases
start_time_train=cputime;

%%%%%%%%%%% Random generate input weights InputWeight (w_i) and biases Bi-
asofHiddenNeurons (b_i) of hidden neurons
% InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
% BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
% tempH=InputWeight*P;
% clear P;                                           %   Release input of
training data
% ind=ones(1,NumberofTrainingData);
% BiasMatrix=BiasofHiddenNeurons(:,ind);             %   Extend the bias ma-
trix BiasofHiddenNeurons to match the demention of H
% tempH=tempH+BiasMatrix;



%==================IELM main Part ================================
E=T; % E = t
for i=1:NumberofHiddenNeurons
   InputWeight(i,:)=rand(1,NumberofInputNeurons)*2-1; %  w = rand(1,noofin)
   BiasofHiddenNeurons(i,:)=rand(1,NumberofTrainingData); % bias

   H(i,:)=(1 ./ (1 + exp(-(InputWeight(i,:)*P+BiasofHiddenNeurons(i,:))))); %
H =  (1 / 1 + exp  (w *P + bias)
   %clear p;
   tempH(i,:) = pinv(H(i,:)'); % Temph =  H^-1
   B(i,:)= tempH(i,:) * E'; % Beta = H^-1 * E
   E = E - (H(i,:)'*B(i,:))';% E = E - H*beta
end
```

```
%=============================================================================
========
%%%%%%%%%%% Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        %%%%%%%% Sine
        H = sin(tempH);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H = double(hardlim(tempH));
    case {'tribas'}
        %%%%%%%% Triangular basis function
        H = tribas(tempH);
    case {'radbas'}
        %%%%%%%% Radial basis function
        H = radbas(tempH);
        %%%%%%%% More activation functions can be added here
end
clear tempH;                                        %   Release the temparary
array for calculation of hidden neuron output matrix H

%%%%%%%%%%% Calculate output weights OutputWeight (beta_i)



end_time_train=cputime;
TrainingTime=end_time_train-start_time_train        %   Calculate CPU time
(seconds) spent for training ELM

%%%%%%%%%%% Calculate the training accuracy
Y=(H' * B)';                             %   Y: the actual output of the
training data
if Elm_Type == REGRESSION
    norm_TV_T=(T-min(T))/(max(T)-min(T));
    norm_TV_Y=(Y-min(Y))/(max(Y)-min(Y));
    TrainingAccuracy=sqrt(mse(norm_TV_T - norm_TV_Y))            %   Calcu-
late testing accuracy (RMSE) for regression case               %   Calculate
training accuracy (RMSE) for regression case
end
clear H;

%%%%%%%%%%% Calculate the output of testing input
start_time_test=cputime;
tempH_test=InputWeight*TV.P;
clear TV.P;             %   Release input of testing data
ind=ones(1,NumberofTestingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias ma-
trix BiasofHiddenNeurons to match the demention of H
tempH_test=tempH_test + BiasMatrix;
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
```

```
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        %%%%%%%% Sine
        H_test = sin(tempH_test);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H_test = hardlim(tempH_test);
    case {'tribas'}
        %%%%%%%% Triangular basis function
        H_test = tribas(tempH_test);
    case {'radbas'}
        %%%%%%%% Radial basis function
        H_test = radbas(tempH_test);
        %%%%%%%% More activation functions can be added here
end
TY=(H_test' * B)';                       %   TY: the actual output of the
testing data
end_time_test=cputime;
TestingTime=end_time_test-start_time_test           %   Calculate CPU time
(seconds) spent by ELM predicting the whole testing data

if Elm_Type == REGRESSION
    norm_TV_T=(TV.T-min(TV.T))/(max(TV.T)-min(TV.T));
    norm_TV_Y=(TY-min(TY))/(max(TY)-min(TY));
    TestingAccuracy=sqrt(mse(norm_TV_T - norm_TV_Y))            %   Calculate
testing accuracy (RMSE) for regression case
end

if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Training=0;
    MissClassificationRate_Testing=0;

    for i = 1 : size(T, 2)
        [x, label_index_expected]=max(T(:,i));
        [x, label_index_actual]=max(Y(:,i));
        if label_index_actual~=label_index_expected

MissClassificationRate_Training=MissClassificationRate_Training+0.35;
        end
    end
    TrainingAccuracy=1-MissClassificationRate_Training/size(T,2)
    for i = 1 : size(TV.T, 2)
        [x, label_index_expected]=max(TV.T(:,i));
        [x, label_index_actual]=max(TY(:,i));
        if label_index_actual~=label_index_expected
            MissClassificationRate_Testing=MissClassificationRate_Testing+1;
        end
    end
    TestingAccuracy=1-MissClassificationRate_Testing/size(TV.T,2)
end
end
```

Results:

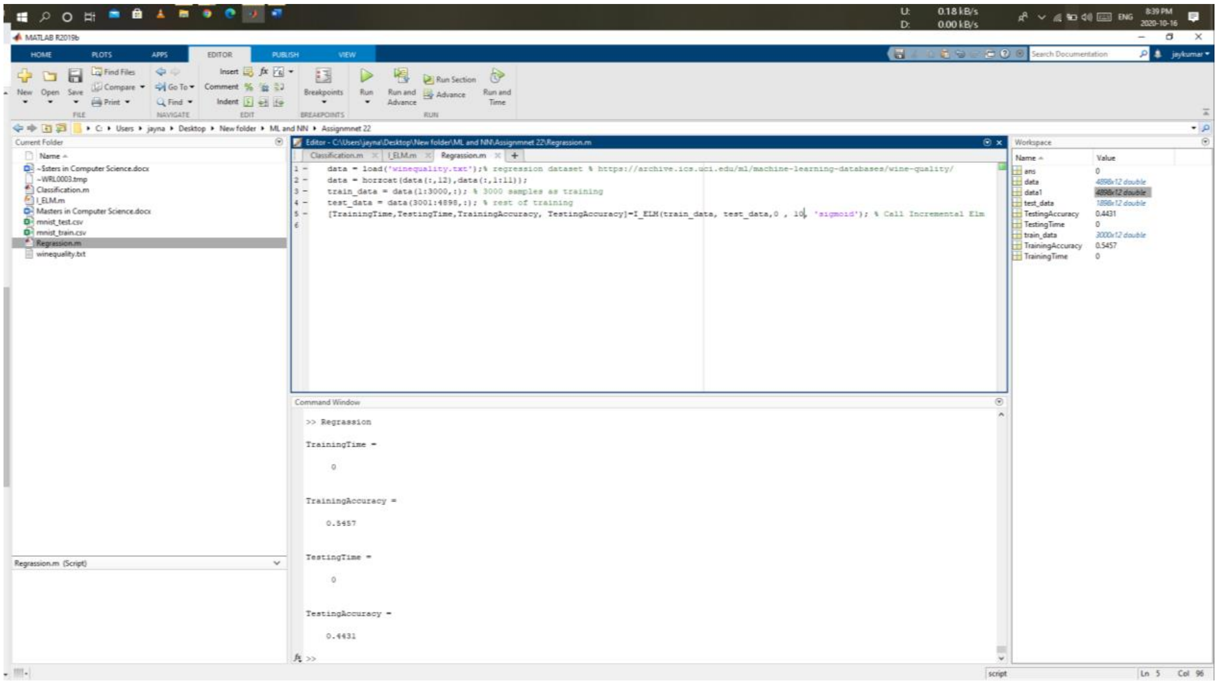
Mnist(Classification)

- Training Accuracy: 68.93%
- Training Time: 6.9698s
- Testing Accuracy: 67.71%
- Testing Time: 0.125s

Wine Quality(Regrassion)

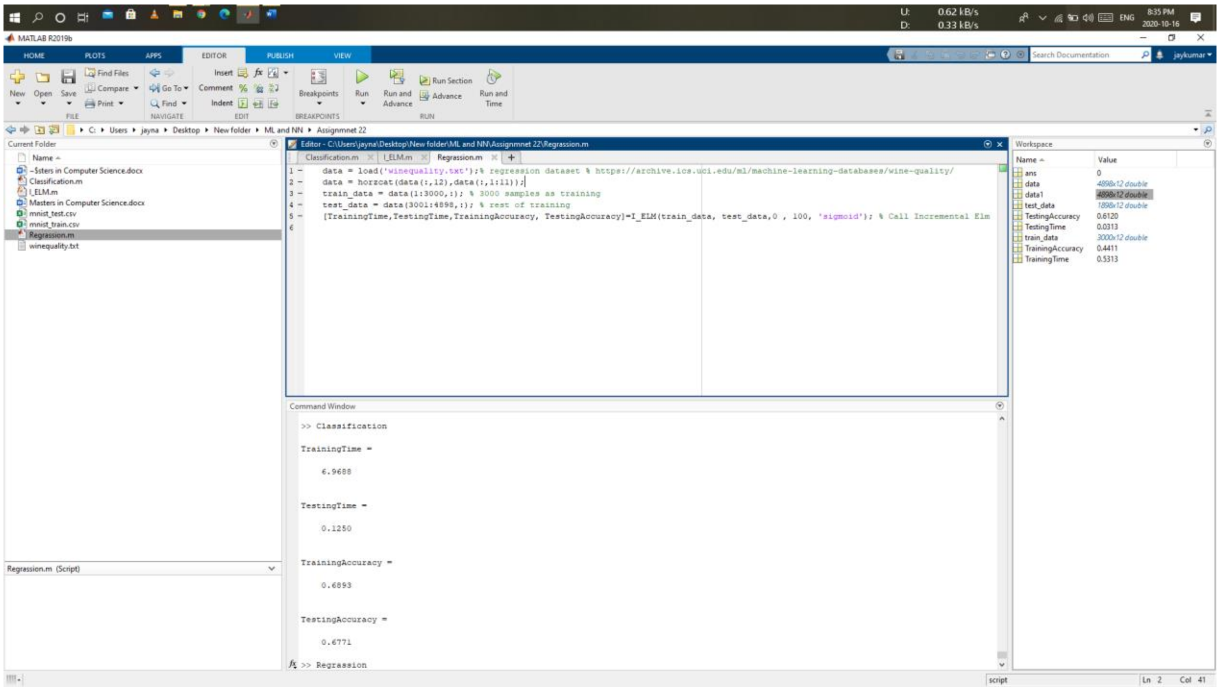
- Training Accuracy:54.57 %
- Training Time: 0.007s
- Testing Accuracy: 44.31%
- Testing Time: 0.0005s

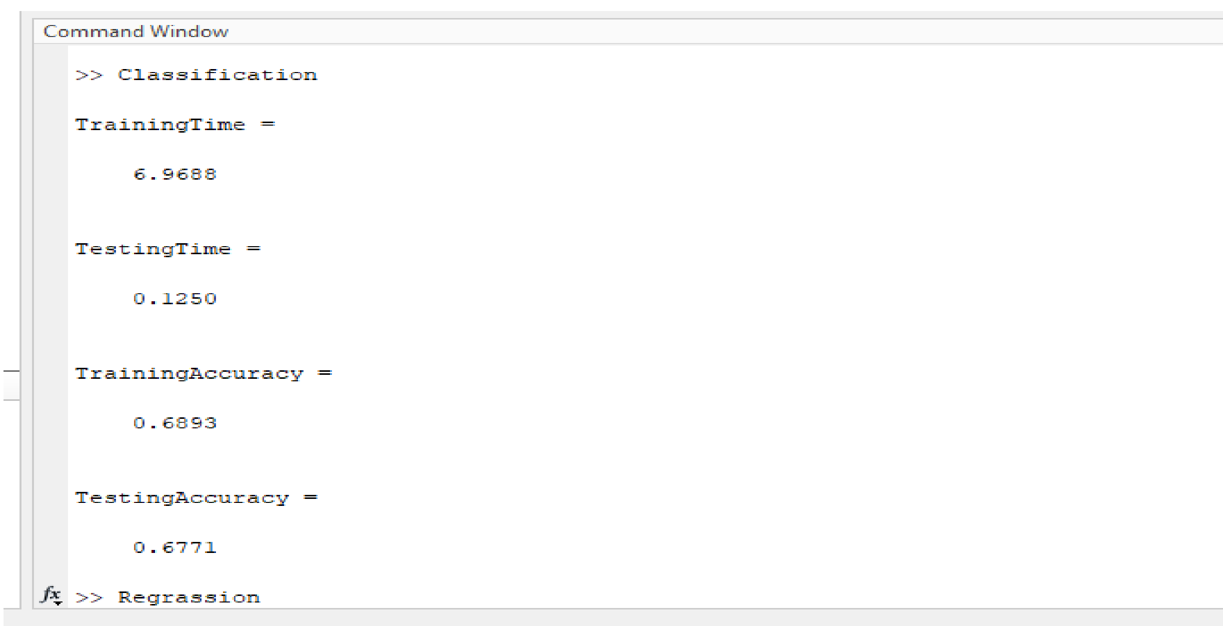
Outputs:



Command Window

```
>> Regrassion

TrainingTime =

     0


TrainingAccuracy =

    0.5457


TestingTime =

     0


TestingAccuracy =

    0.4431

>>
```





Command Window
>> Classification
TrainingTime =
6.9688
TestingTime =
0.1250
TrainingAccuracy =
0.6893
TestingAccuracy =
0.6771
>> Regrassion

Reference:

- https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
- https://www.ntu.edu.sg/home/egbhuang/elm_codes.html