Masters in Computer Science

Topics in Machine Learning & Neural Net

(COMP-5011)

Name:

Jaykumar Nariya (1116571)

Assignment 3 Part 1:

1.(5points) Write a python or MATLAB based codes to train an incremental extreme learning machine with an intelligent optimization algorithm (GA or PSO or others). Students could use the codes of their Incremental Extreme Learning Machine code as their reference. Test the written code by using two datasets (one classification dataset and one UCI regression dataset)

Code:

```
function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
ipso_elm(TrainingData_File, TestingData_File, Elm_Type, NumberofHidden-
Neurons, 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 and PSo main Part ================================
E=T; % E = t

for i=1:NumberofHiddenNeurons
    %[x_best]=PSO(40,P,E,NumberofTrainingData,NumberofInputNeurons,2,1,-1);%
    %if don't allowed inbulit function of PSo for use then remove comment
    %from upper line and put in comment below four lines

    f = @(x)fitness(x,P,NumberofInputNeurons,NumberofTrainingData,E); % call
fitness function with additional args
    options = optimoptions('particleswarm','SwarmSize',100); % option of PSO
    nvars=NumberofInputNeurons+1; % number of variabls
    lb=-1;ub=1; % lower bound upper bound
    [x_best,f_val] = particleswarm(f,nvars,lb,ub); % find best weights for
IELM and less RMSE

    InputWeight(i,:)=x_best(1:NumberofInputNeurons)';
    BiasofHiddenNeurons(i,:)=x_best(NumberofInputNeurons+1);
    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
```

```
%%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
%OutputWeight=pinv(H') * T';                        % implementation without
regularization factor //refer to 2006 Neurocomputing paper
%OutputWeight=inv(eye(size(H,1))/C+H * H') * H * T';   % faster method 1
//refer to 2012 IEEE TSMC-B paper
%implementation; one can set regularizaiton factor C properly in classifica-
tion applications
%OutputWeight=(eye(size(H,1))/C+H * H') \ H * T';      % faster method 2
//refer to 2012 IEEE TSMC-B paper
%implementation; one can set regularizaiton factor C properly in classifica-
tion applications

%If you use faster methods or kernel method, PLEASE CITE in your paper
properly:

%Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang, "Extreme
Learning Machine for Regression and Multi-Class Classification," submitted to
IEEE Transactions on Pattern Analysis and Machine Intelligence, October 2010.

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'}
        %%%%%%%% 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'}
end
```

```
%%%%%%%% 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_Train-
ing+1;
        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
```

Fitness Function

```
function fit  = fitness(x,P,NumberofInputNeurons,NumberofTrainingData,E) %
FITNESS FUNCTION FOR pso
    InputWeight=x(1,NumberofInputNeurons); %  w = rand(1,noofin)
    BiasofHiddenNeurons=x(NumberofInputNeurons+1);
    ind=ones(1,NumberofTrainingData);
    BiasMatrix=BiasofHiddenNeurons(:,ind);% bias
    H=(1 ./ (1 + exp(-(InputWeight*P+BiasMatrix)))); % H =  (1 / 1 + exp  (w
*P + bias)
    %clear p;
    tempH = pinv(H'); % Temph =  H^-1
    B= tempH * E'; % Beta = H^-1 * E
    E = E - (H'*B)';% E = E - H*beta
    fit=sqrt(mse(E));
end
```

Results:

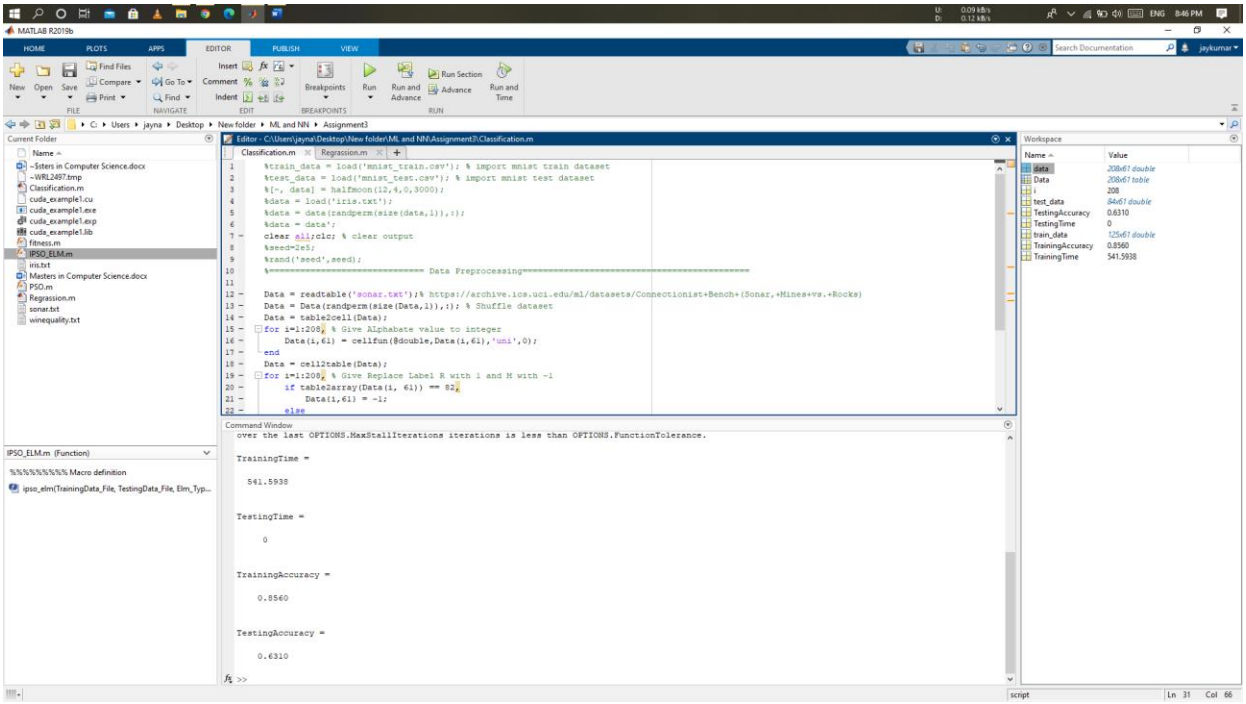
Sonar(Classification)

- **Training Accuracy: 85.60%**
- **Training Time: 541.5938s**
- **Testing Accuracy: 63.10%**
- **Testing Time: 0.007s**

Wine Quality(Regrassion)

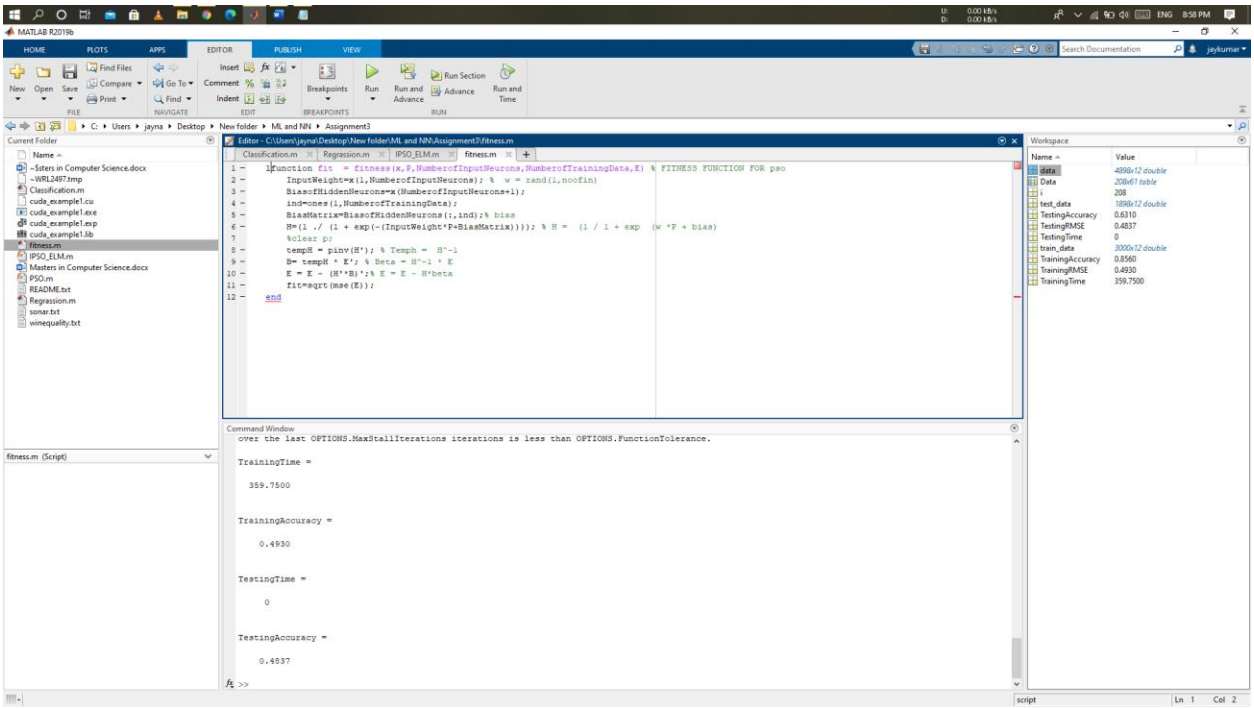
- **Training RMSE:0.4930**
- **Training Time: 359.7500s**
- **Testing RMSE: 0.4837**
- **Testing Time: 0.0005s**

Outputs:



Command Window

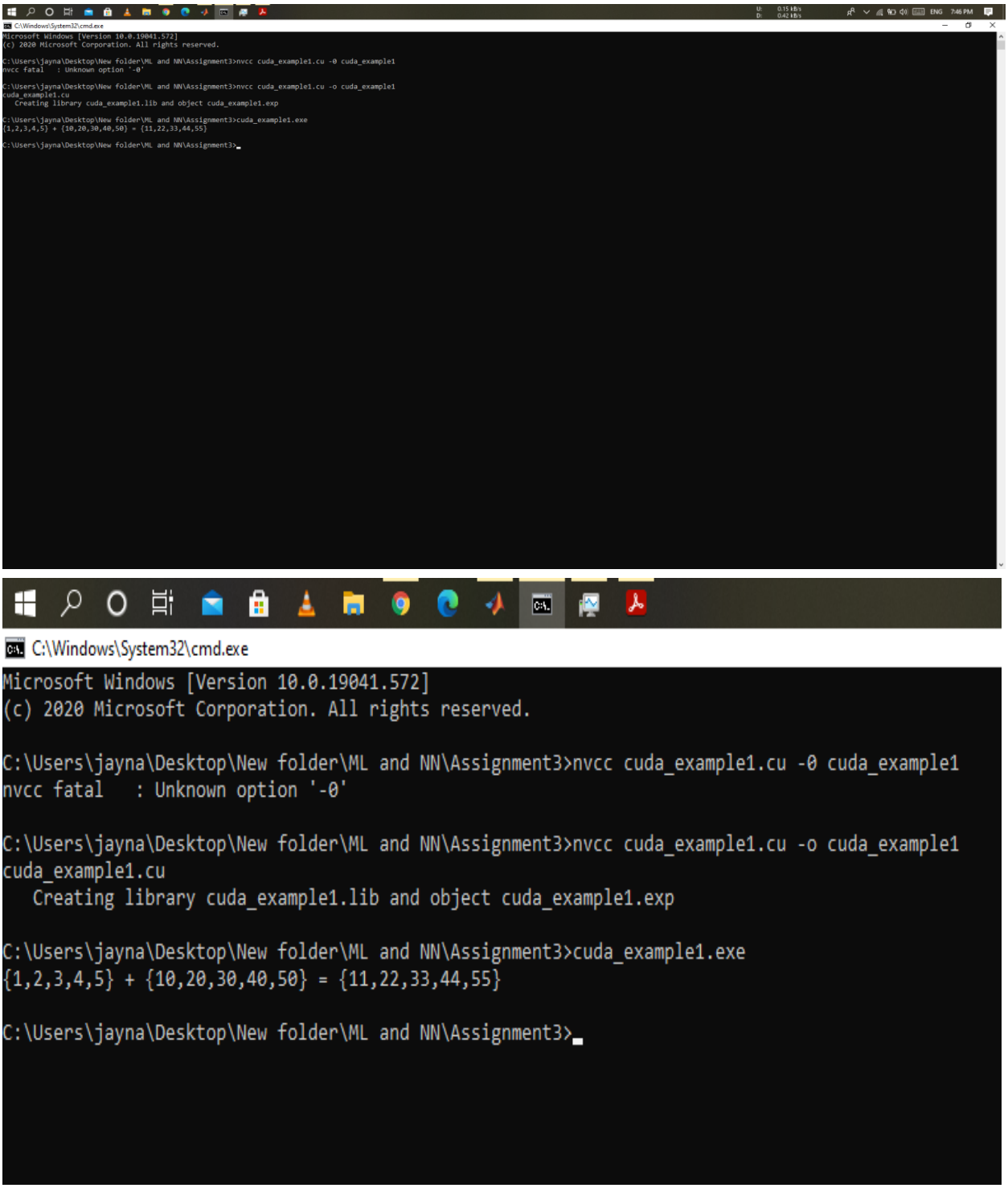
```
over the last OPTIONS.MaxStallIterations iterations is less than OPTIONS.FunctionTolerance.

TrainingTime =

  541.5938


TestingTime =

     0


TrainingAccuracy =

    0.8560


TestingAccuracy =

    0.6310

fx >>
```



ommand Window

```
over the last OPTIONS.MaxStallIterations iterations is less than OPTIONS.FunctionTolerance.

TrainingTime =

  359.7500


TrainingAccuracy =

    0.4930


TestingTime =

     0


TestingAccuracy =

    0.4837

>>
```

2. (3 points) Run the basic CUDA code example shared in the D2L, screen-shot your result.



Reference:

- https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)
- https://www.ntu.edu.sg/home/egbhuang/elm_codes.html