ENPM808F- Robot Learning HW#2

Yash Manian

9/26/2016

<u>HW#2_A1:</u>

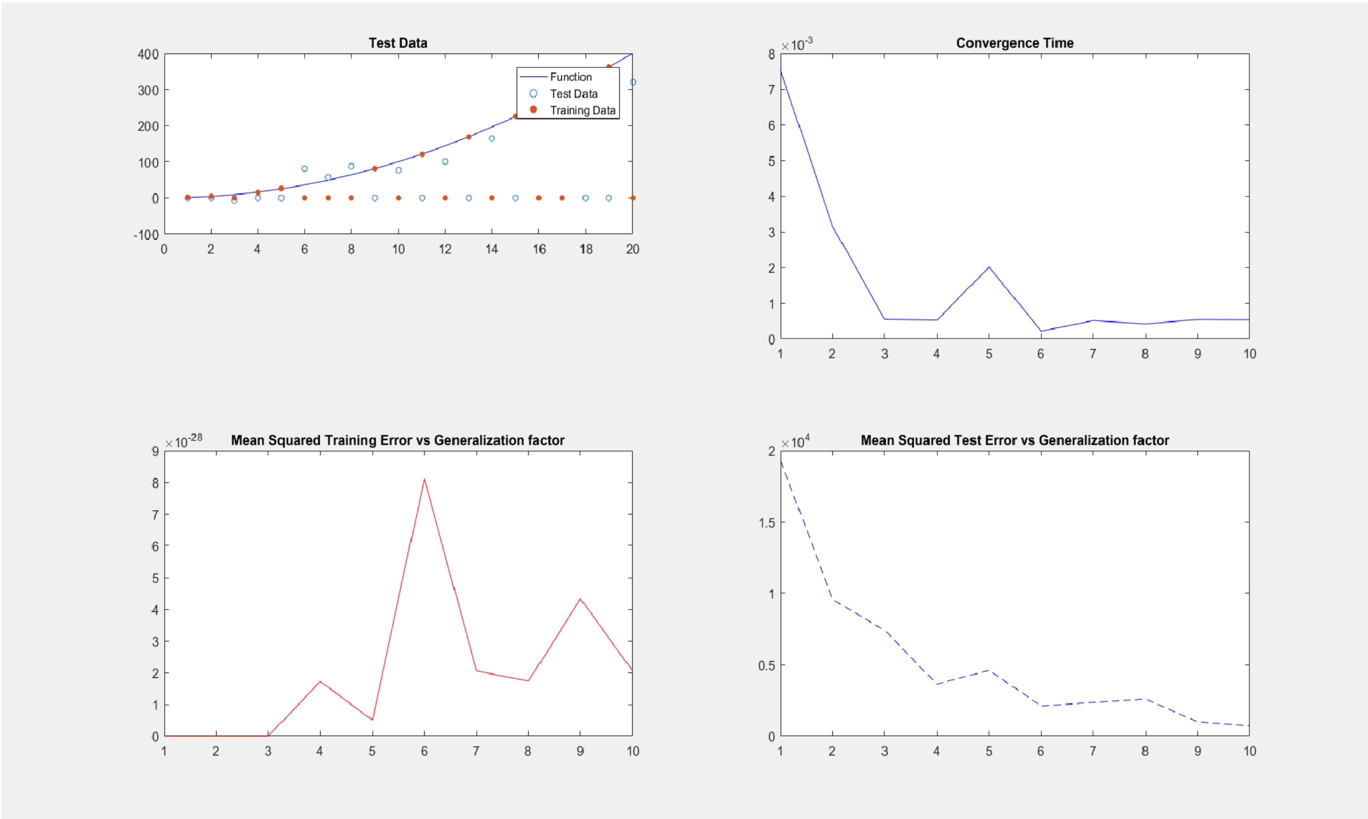
<u>Discrete CMAC MATLAB Program:</u>

```
%ENPM 808F Discrete CMAC
% By : Yash Manian
clear all
%%
% Variables
%Given data
Input=1:1:20;
Output = transpose(Input.^2);
GenFactor = 10;

% Data points

 TrainingData=[1;2;4;5;9;11;13;15;18;19];
 TestData = sort(setdiff(Input,TrainingData))';
 [TrainingRow,TrainingColumn]=size(TrainingData);

%CMAC variables
[OutputRow,OutputColumn] = size(Output);
W_Dimension=OutputRow+(GenFactor);
Weights=zeros(W_Dimension,1);
Output_Des=zeros(OutputRow,1);
Output_Des_test=zeros(OutputRow,1);
Error_1 = 0;
Error_2 = 0;
StopVal = 0;
Error_Test = 0;
Training_Output = zeros(OutputRow,1);
Test_Output = zeros(OutputRow,1);
Time=zeros(10,1);

% Error variables
MeanSquaredError_Test = zeros(10,1);
MeanSquaredError_Training = zeros(10,1);
%%

 for GenFactor = 1:1:10
```

```
tic

%%
% Training Weights
while(1)
    for i=1:1:TrainingRow
        Index = TrainingData(i);
        Input_Current = Input(Index);
        Training_Output(Index) = Output(Index);
        Output_Current= Output(Index);
        W_Current = Weights(Index:Index+GenFactor-1);
        Error_Current = (Output_Current - sum(W_Current));
        W_Value = Error_Current/GenFactor;

        for j=Index:1:Index+GenFactor-1
            Weights(j)= Weights(j) + W_Value;
        end


Output_Des(Index)=sum(Weights(Index:Index+GenFactor-1));
        Error_1=(Error_1+Error_Current)/Output_Current;
    end
    if abs(Error_1) <= 0.01
            break;
    end
end

%%
% Timing
toc
Time(GenFactor) = toc;

%%
% Running on Test Data
for k=1:1:TrainingRow
    Index_Test = TestData(k);
    IC_Test = Input(Index_Test);
    OC_Test = Output(Index_Test);
    Test_Output(Index_Test) = Output(Index_Test);

Output_Des_test(Index_Test)=sum(Weights(Index_Test:Index_Te
st+GenFactor-1));
end
```

```
%%
% Error
Error_Training = Training_Output-Output_Des; % Training
Error Array
Error_Test = Test_Output-Output_Des_test;  % Test Error


MeanSquaredError_Training(GenFactor) =
immse(Training_Output,Output_Des); % Mean Squared Training
Error
MeanSquaredError_Test(GenFactor) =
immse(Test_Output,Output_Des_test); % Mean Squared Test
Error
end
%%

% Plotting Results
subplot(3,2,1)
plot(Input,Output,'-b')
title('Training Data')
hold on
scatter(Input,Output_Des_test,20)
scatter(Input,Output_Des,20,'Filled')
legend('Function','Test Data','Training Data')
title('Test Data')
hold off

% Plotting Error
subplot(2,2,3)
plot(1:1:10,MeanSquaredError_Training,'-r')
title('Mean Squared Training Error vs Generalization
factor')
subplot(2,2,4)
plot(1:1:10,MeanSquaredError_Test,'--b')
title('Mean Squared Test Error vs Generalization factor')

% Plotting Convergence Time
subplot(2,2,2)
plot(1:1:10,Time,'-b')
title('Convergence Time')
```

Discrete CMAC Results:



HW#2_A2:

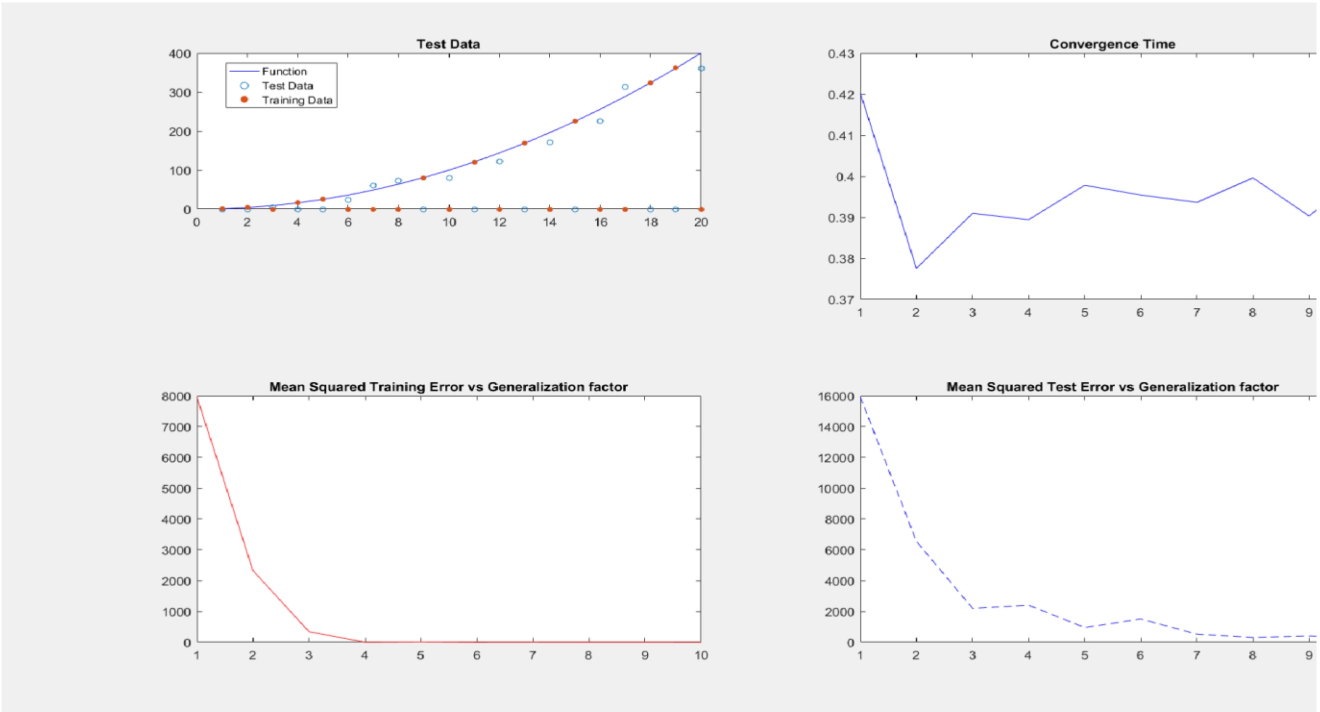
Continuous CMAC MATLAB Program:

```
%ENPM 808F Continuous CMAC
% By : Yash Manian

clear all
%%
% Variables
%Given data
Input=1:1:20;
Output = transpose(Input.^2);
GenFactor = 10;

% Data points

 TrainingData=[1;2;4;5;9;11;13;15;18;19];
 TestData = sort(setdiff(Input,TrainingData))';
 [TrainingRow,TrainingColumn]=size(TrainingData);

%CMAC variables
[OutputRow,OutputColumn] = size(Output);
W_Dimension=OutputRow+(GenFactor);
Weights=zeros(W_Dimension,1);
Output_Des=zeros(OutputRow,1);
Output_Des_test=zeros(OutputRow,1);
Error_1 = 0;
Error_2 = 0;
StopVal = 0;
Error_Test = 0;
Training_Output = zeros(OutputRow,1);
Test_Output = zeros(OutputRow,1);
Time=zeros(10,1);

% Error variables
MeanSquaredError_Test = zeros(10,1);
MeanSquaredError_Training = zeros(10,1);
%%

for GenFactor = 1:1:10

tic

%%
```

```
% Training Weights
 while(1)
    for i=1:1:TrainingRow
        Index = TrainingData(i);
        Input_Current = Input(Index);
        Training_Output(Index) = Output(Index);
        Output_Current= Output(Index);
        W_Current = Weights(Index:Index+GenFactor);
        Error_Current = (Output_Current - sum(W_Current));
        W_Value = Error_Current/GenFactor;

        for j=Index+1:1:Index+GenFactor-1
            Weights(j)= Weights(j) + W_Value;
        end
        Weights(Index)= W_Value/2;
        Weights(Index+GenFactor) = W_Value/2;

Output_Des(Index)=sum(Weights(Index:Index+GenFactor));
        Error_1=(Error_1+Error_Current)/Output_Current;
    end
    if abs(Error_Current) <= 0.01
            break;
    end
end


%%
% Timing
toc
Time(GenFactor) = toc;

%%
% Running on Test Data
for k=1:1:TrainingRow
    Index_Test = TestData(k);
    IC_Test = Input(Index_Test);
    OC_Test = Output(Index_Test);
    Test_Output(Index_Test) = Output(Index_Test);

Output_Des_test(Index_Test)=sum(Weights(Index_Test:Index_Te
st+GenFactor-1));
end

%%
```

```
% Error
Error_Training = Training_Output-Output_Des; % Training
Error Array
Error_Test = Test_Output-Output_Des_test;  % Test Error


MeanSquaredError_Training(GenFactor) =
immse(Training_Output,Output_Des); % Mean Squared Training
Error
MeanSquaredError_Test(GenFactor) =
immse(Test_Output,Output_Des_test); % Mean Squared Test
Error
end
%%

% Plotting Results
subplot(3,2,1)
plot(Input,Output,'-b')
title('Training Data')
hold on
scatter(Input,Output_Des_test,20)
scatter(Input,Output_Des,20,'Filled')
legend('Function','Test Data','Training Data')
title('Test Data')
hold off

% Plotting Error
subplot(2,2,3)
plot(1:1:10,MeanSquaredError_Training,'-r')
title('Mean Squared Training Error vs Generalization
factor')
subplot(2,2,4)
plot(1:1:10,MeanSquaredError_Test,'--b')
title('Mean Squared Test Error vs Generalization factor')

% Plotting Convergence Time
subplot(2,2,2)
plot(1:1:10,Time,'-b')
title('Convergence Time')
```

Continuous CMAC Results:



HW#2_A3:

Non recurrent networks typically have a feed-forward structure. Thus, typically, one of the inputs in these networks is time. Inputs go to a hidden layer of elements or weights and the output of the system is computed in a forward manner. In recurrent networks, the hidden units have a feedback from themselves to themselves. Thus creating a sample of (t-1) with respect to the current sample. Implementing this in a CMAC, the weights have a feedback allowing them to recalculate based on the following factors:

1. Current Input
2. Required output
3. Error
4. Weights at time (t-1)

In a typical feed-forward CMAC, only the first three are available, thus making a recurrent CMAC more accurate.
Also, this makes processing higher dimensional inputs easier. Tasks such as handwriting analysis can make use of a high dimensional input which is independent of time, allowing analysis of structures such as loops and spirals, as the state of the input is compared to the previous state.