Robot Learning-HW2 ENPM690

Discrete CMAC:

The oneD discrete CMAC is trained under y=cos(x)

A total of 100 samples were chosen from the range of (0,2*pi) out of which 70 samples were used as the training Data and 30 samples were chosen to train the Data

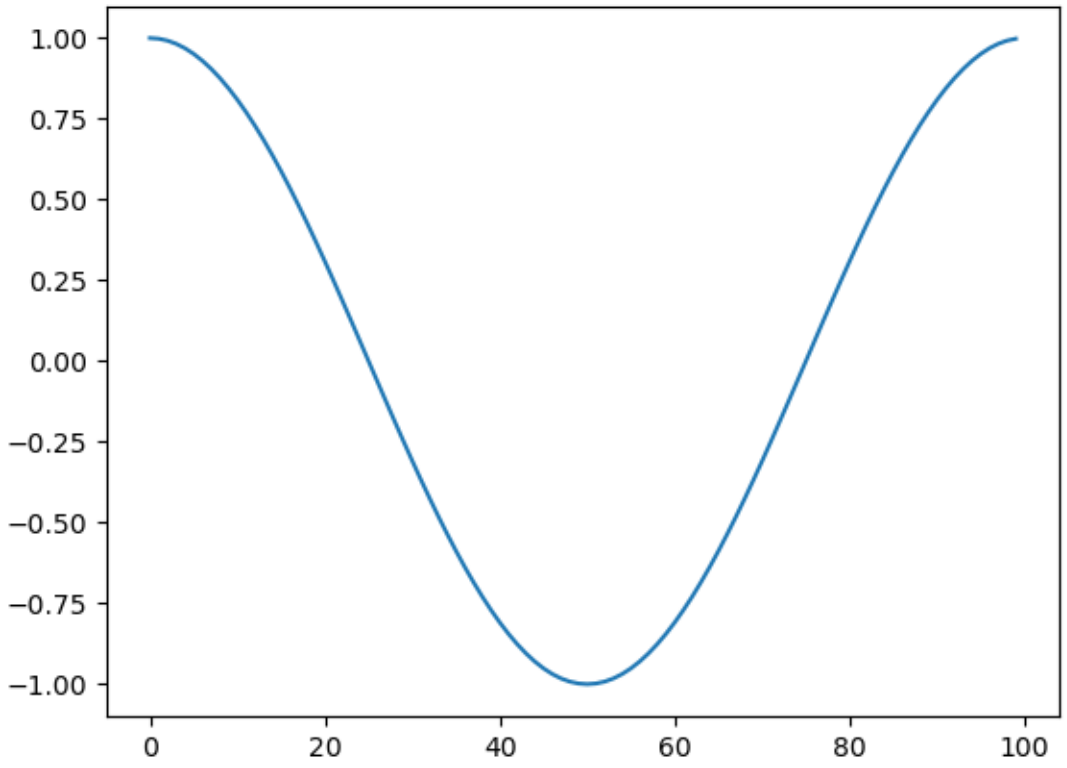

Fig1. Variance of Data over the given function

The above figure depicts the range of Data over 100 samples plotted onto the function y=sin(x).

https://github.com/Nishanth2708/CMAC/upload/master

For the Train Data a random 70 samples from the Data is chosen and the graph is plotted for the same function y=cos(x) and graph looks like

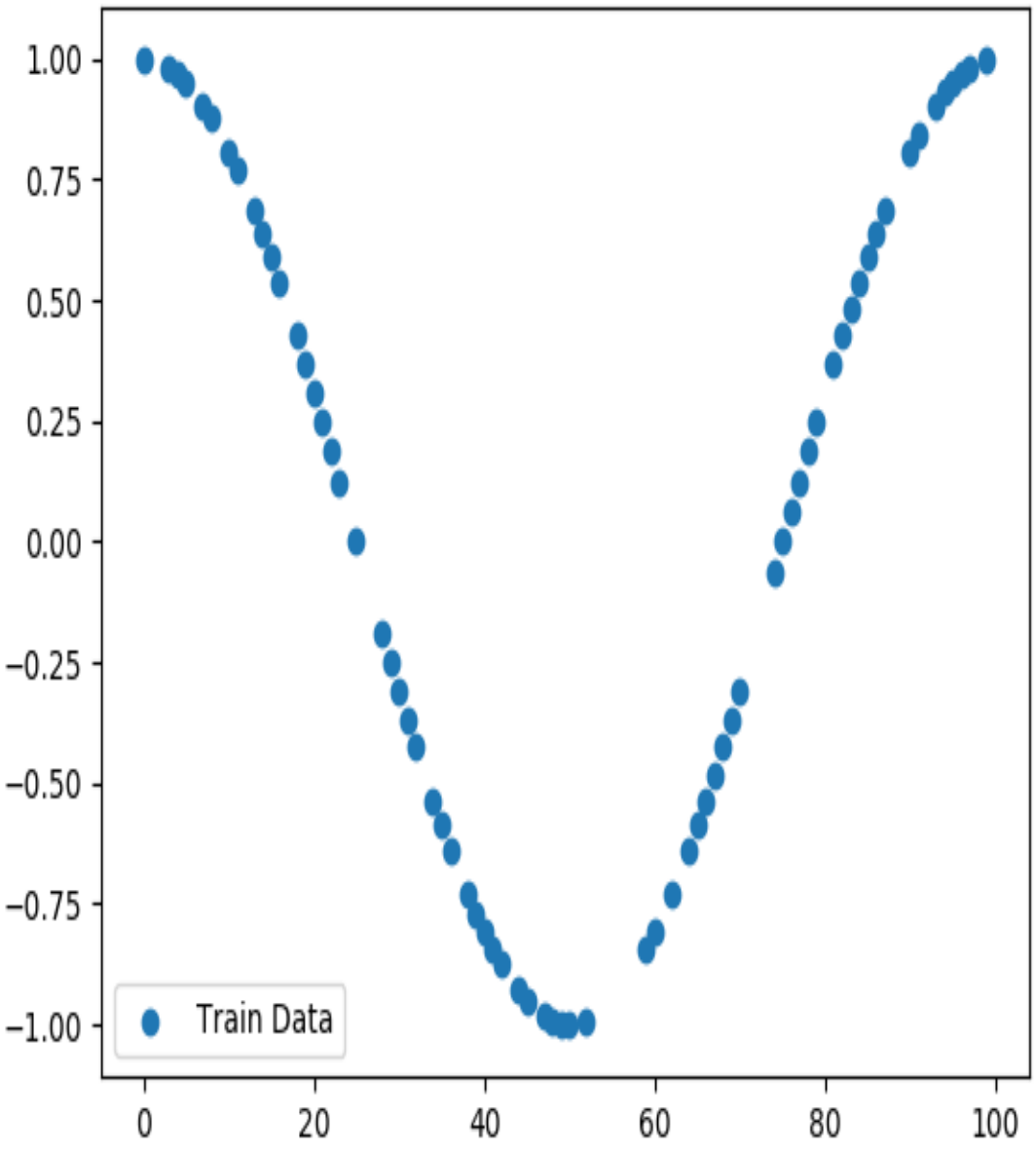

Fig2. 70% of input Data

https://github.com/Nishanth2708/CMAC/upload/master

The Test Data is 30 samples of the Overall Data ond the below plot shows the spread of test Data Fig3. Spread of Test Data

Fi

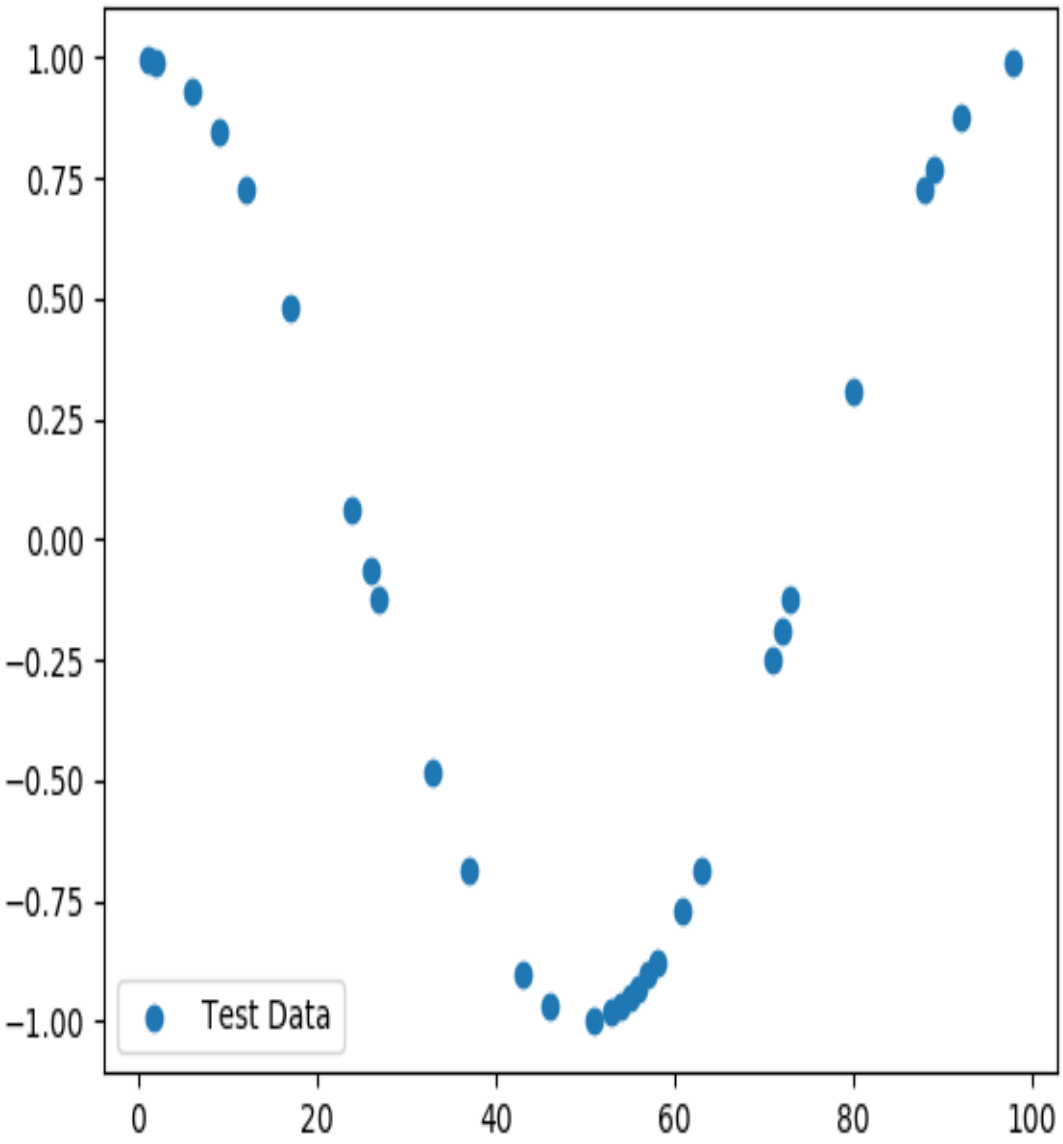

Fig3. Test Data for 30 samples

https://github.com/Nishanth2708/CMAC/upload/master

The local area for the cells is considerd as 5 and the function is defined to map 100 sensory inputs to 35 different assosiative cells. The weights are updated continuosly till the error drops to 0.001 and the convergence of the error has been plotted belo

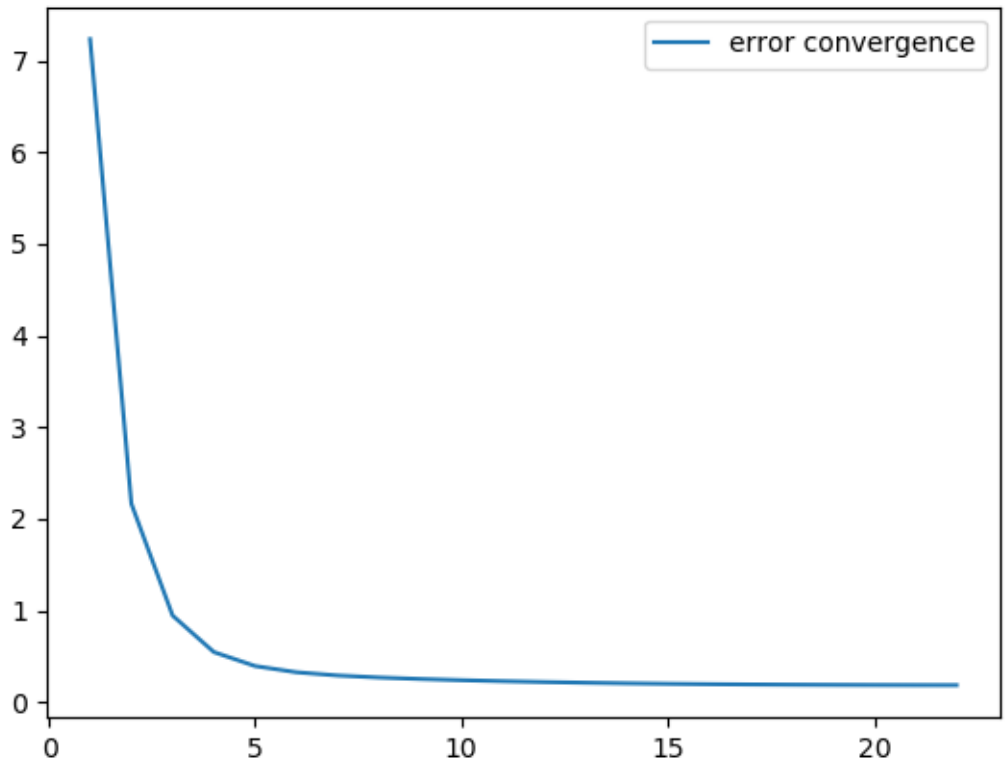

Fig4. Error convergence when updating the weight

https://github.com/Nishanth2708/CMAC/upload/master

Plotting the final output Data on the input Data to observe the discrepancies in the predicted outputs and the plot is drawn below

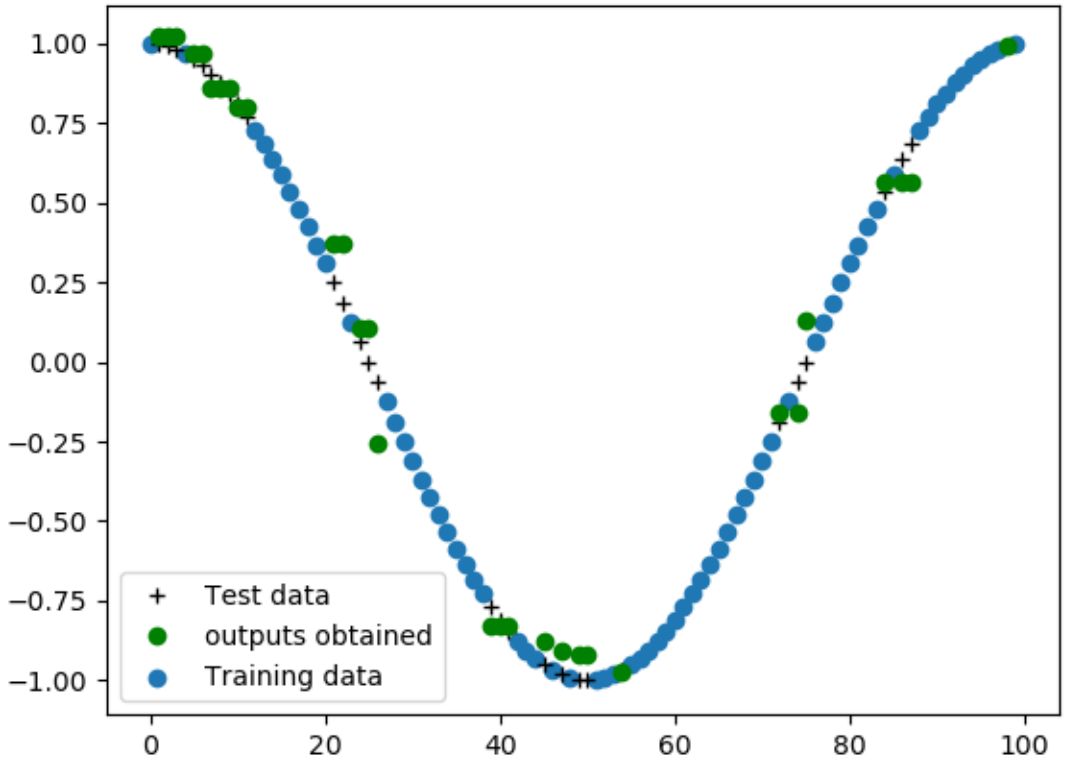

Fig5. Accuracy of the outputs to the Testing Data

Continuos CMAC:

The function y=cos(x) is used for the continuos CMAC over the 100 samples of the Data. There are 70 training Data and 30 test data used in the program.

The below plot shows the error convergence of 0.001

https://github.com/Nishanth2708/CMAC/upload/master

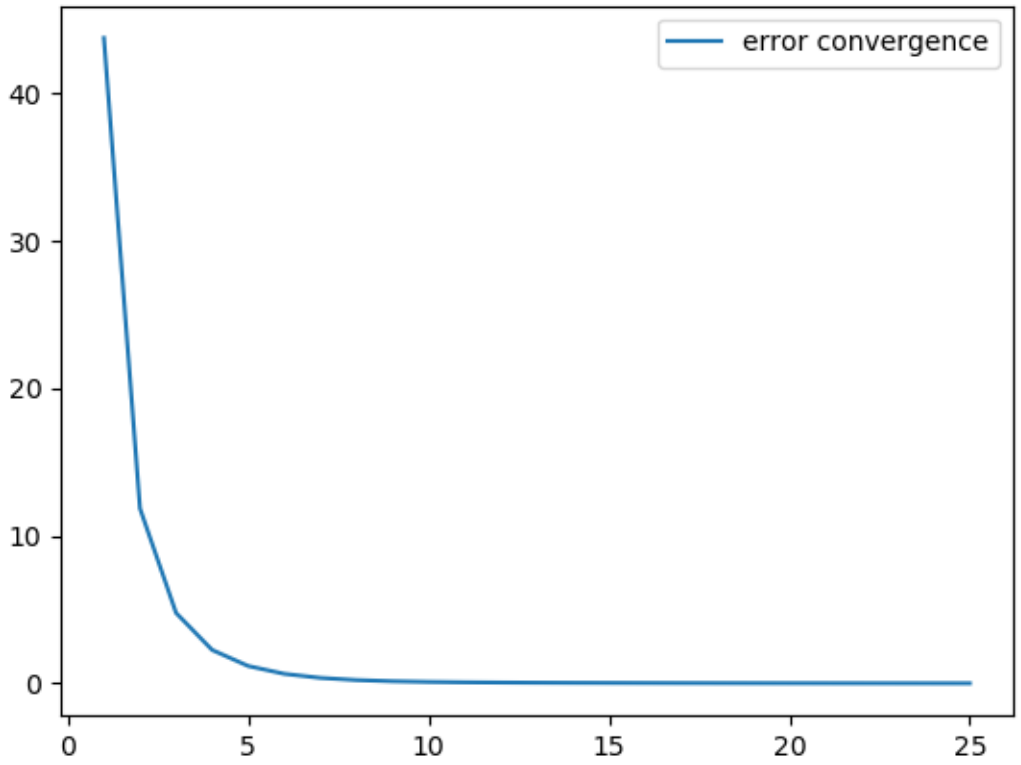

Fig6 Error Convergence

We have used 35 weights for the 100 sampled data and a function is defined to map the associative indexes for the corresponding weights. The floor value and ciel values are used to assign the weights for its correspondive index and the local area is considerd as 5 and the weights are continuos update **until**

the error is dropped to 0.001

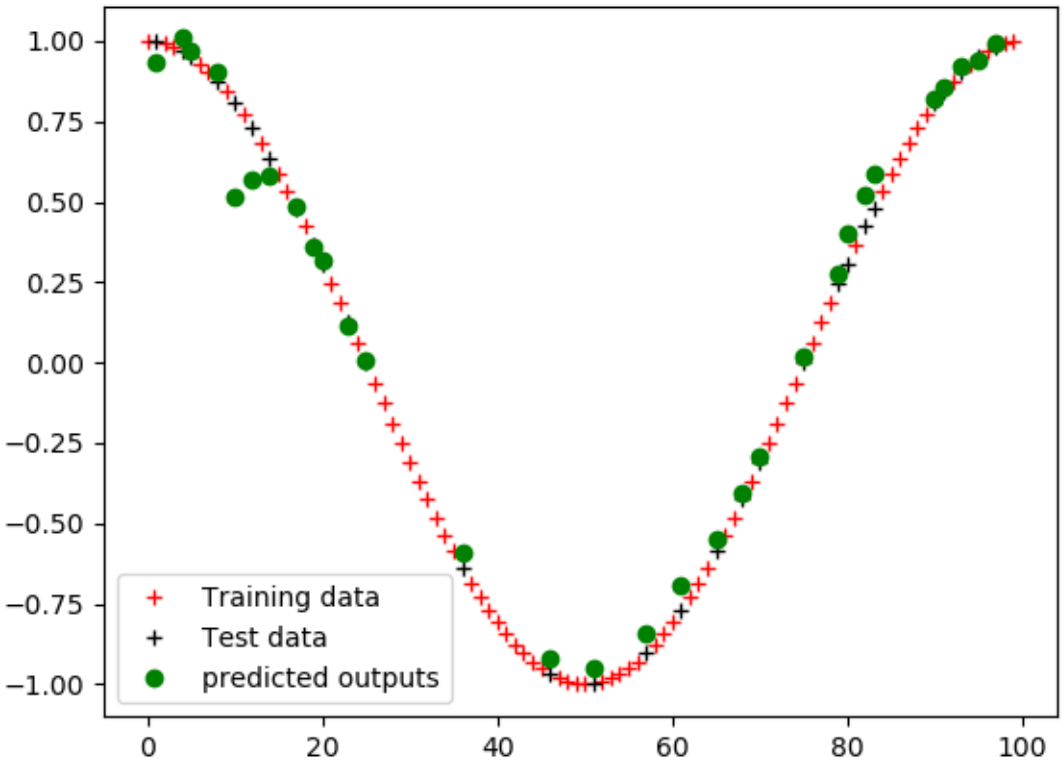

Fig7. Accuracy of the Ouputs to the Testing Data

3**.** Recurrent Connections are based on the idea that they are fed information not just from the previous layer but also from themselves from the previous pass. This means that the order in which you feed the input and train the network matters.

- The recurrent networks have a feedback from the same networks and this sequential information in the networks hidden state eventually spans and effects the processing of each sample presented.

- Implemneting recurrent networks in CMAC the weights have a feedback that can be reclauclated based on the several factors such as the sensory data like the input data, State, error and the desired output that is dependent on time However, during the complex tasks can make high dimensional input independent of time.

- Major problem with this, is the vanishing (or exploding) gradient problem where, information rapidly gets lost over time. Intuitively this wouldn't be much of a problem because these are just weights and not neuron states, but the weights through time is where the information from the past is stored. The weights are also adjusted by the recurrent network model.

https://github.com/Nishanth2708/CMAC/upload/master

• To use this model in our code, the output should be rectified using an error function and we also require another function which will depend on the output generated. Following this would make the function depend on itself and also on the output.This method can be used in many fields as most forms of data that don't have a timeline can be represented as a sequence. In general, recurrent networks are a good choice for advancing or completing information, such as autocompletion.