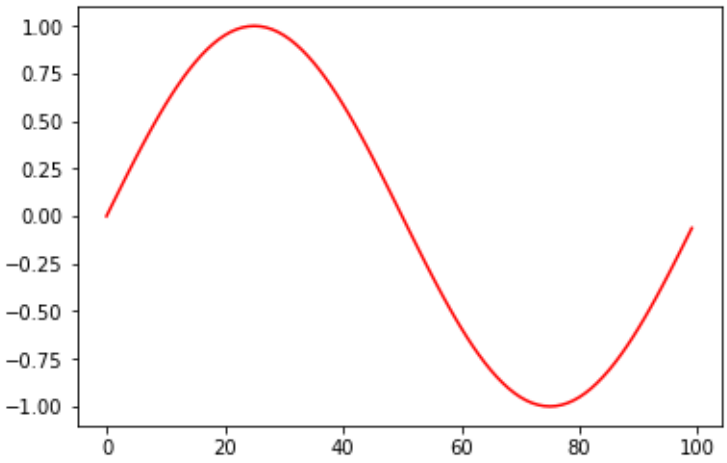
Implementation of CMAC controller

Implementation of Discrete CMAC – Preparing the data

The 1-dimensional discrete CMAC is trained for the following function:

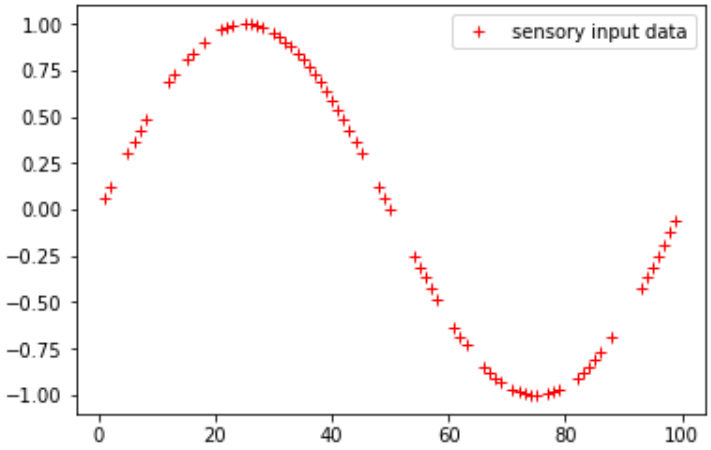
$$y = sin(x)$$



1D discrete CMAC assigns equal weightage for all the weightage vectors. In this work the CMAC is trained on a cosine wave that varies from 0 to 3π. The input is divided into 100 evenly sampled points.

Preparing the input data for training and testing:

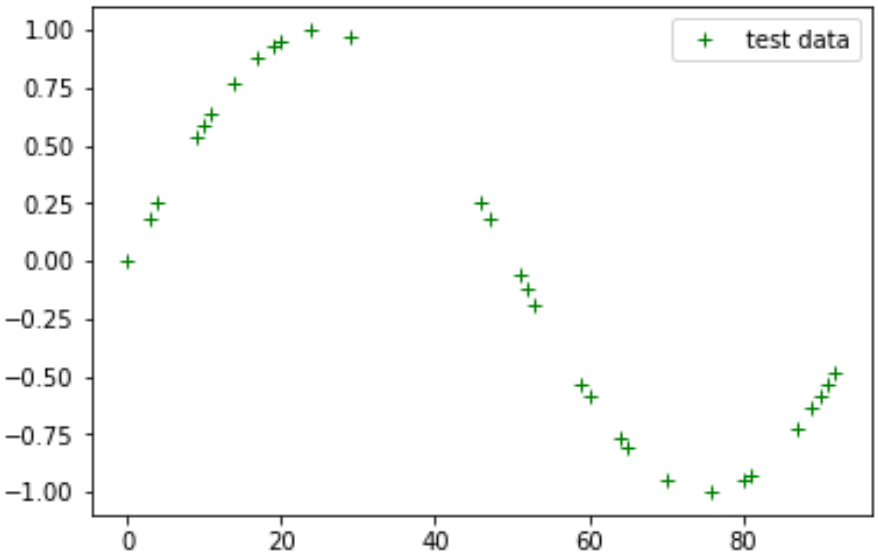
Separated train data:



The input data used is separated as 70% to the training data. The following plot shows the training data

Separated test data:

The below plot shows the separated test data from the 100 samples of input data



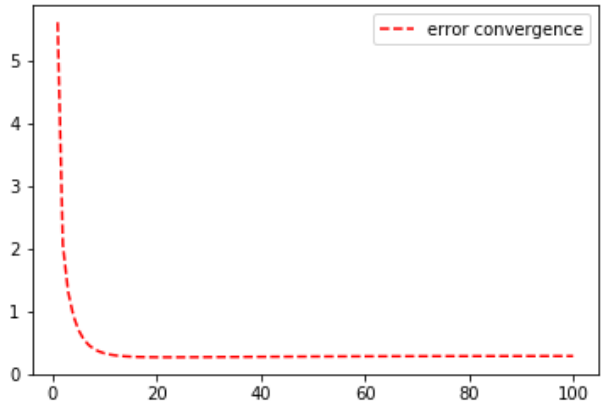
PROGRAMING THE DISCRETE CMAC:

The CMAC controller is implemented using Python3. Steps involved in implementing CMAC are:

Training the Discrete CMAC:

1. A class is defined for the creation of associative cells. This class has the associative cells index and the corresponding weights as the class attributes.
2. A function is defined to map the 100 sensory inputs to 35 different cells.
3. A function is defined to map associative indexes to their corresponding weights based on the generalization factor.
4. The generalization factor (beta) is considered as 5 and the CMAC is implemented.
5. The weights are updated continuously until the absolute error drops to the order of 0.00001

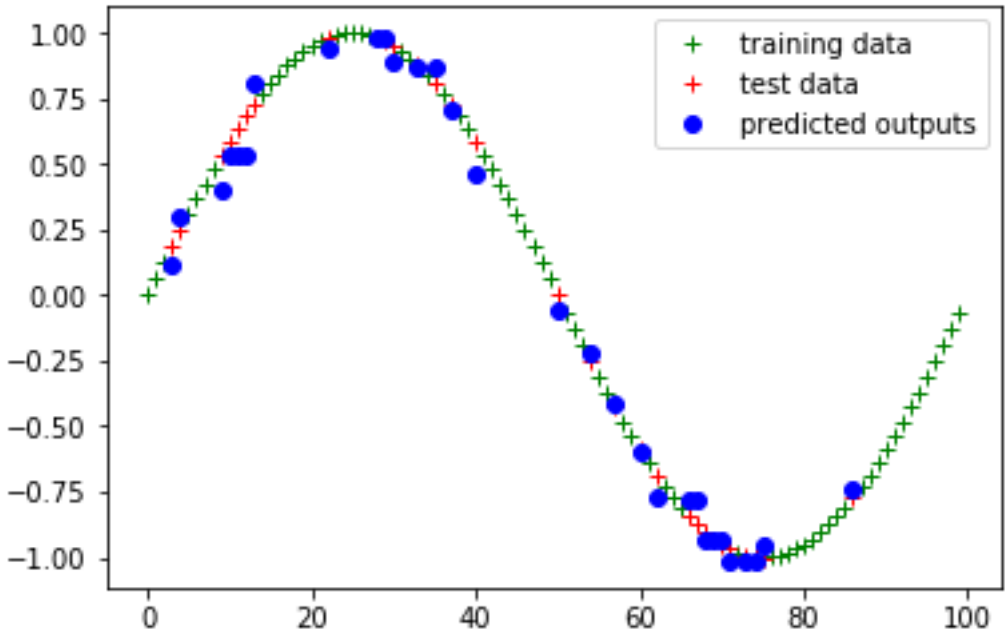
The below plot shows the error convergence:



Predicting the CMAC outputs on the test data:

1. The test data is applied to the updated weights to predict the output.

The below plot shows the predicted outputs against the training and the test data.



IMPLEMENTATION OF CONTINUOUS CMAC:

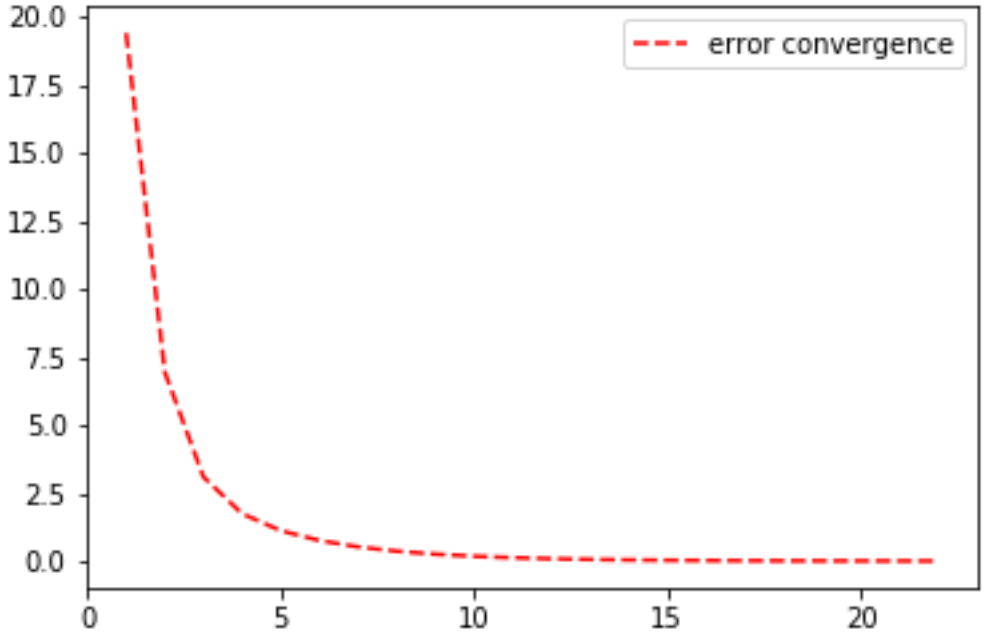
The data that is prepared as for the Discrete CMAC is only used for the continuous CMAC as well

PROGRAMMING THE CONTINUOUS CMAC:

Training the Continuous CMAC:

1. A class is defined for the creation of associative cells. This class has the associative cells index, and the corresponding weights as the class attributes.
2. A function is defined to map the 100 sensory inputs to 35 different cells.
3. A function is defined to map associative indexes to their corresponding weights based on the generalization factor.
    a. In the same function weightages are also given for each weight. Floor and ceil values are obtained to assign weightages to the corresponding weights corresponding to each index.
4. The generalization factor (beta) is considered as 5 and the CMAC is implemented.
5. The weights are updated continuously until the absolute error drops to the order of 0.001

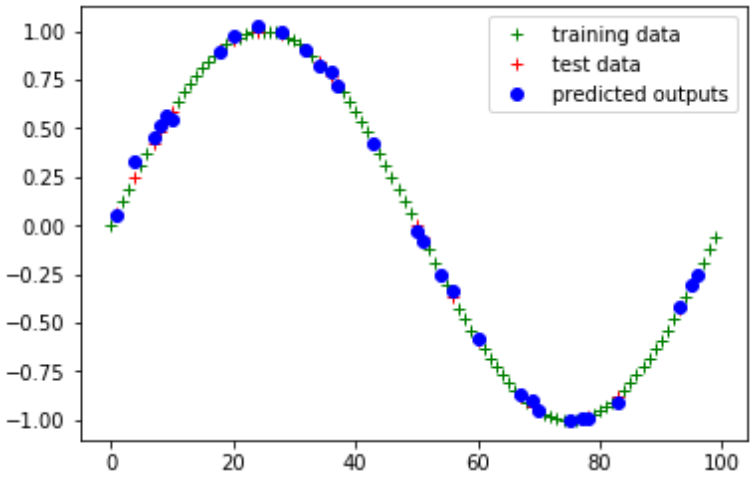
The below plot shows the error convergence for the continuous CMAC:



Predicting the CMAC outputs on the test data:

1. The test data is applied to the updated weights to predict the output.

The below plot shows the predicted outputs against the training and the test data.



Comparison between discrete and continuous CMAC:

1. The error convergence is faster in the case of continuous CMAC compared to the Discrete CMAC