Training CMAC to give output for a given function

The 1D discrete CMAC is trained under the following function.

y = Sin(x)

100 Samples were taken from the range 0 to 2*π. Out of that random 70 data were stored as training data and remaining 30 data were stored as testing data. Random array of 35 weights was taken to train the CMAC.

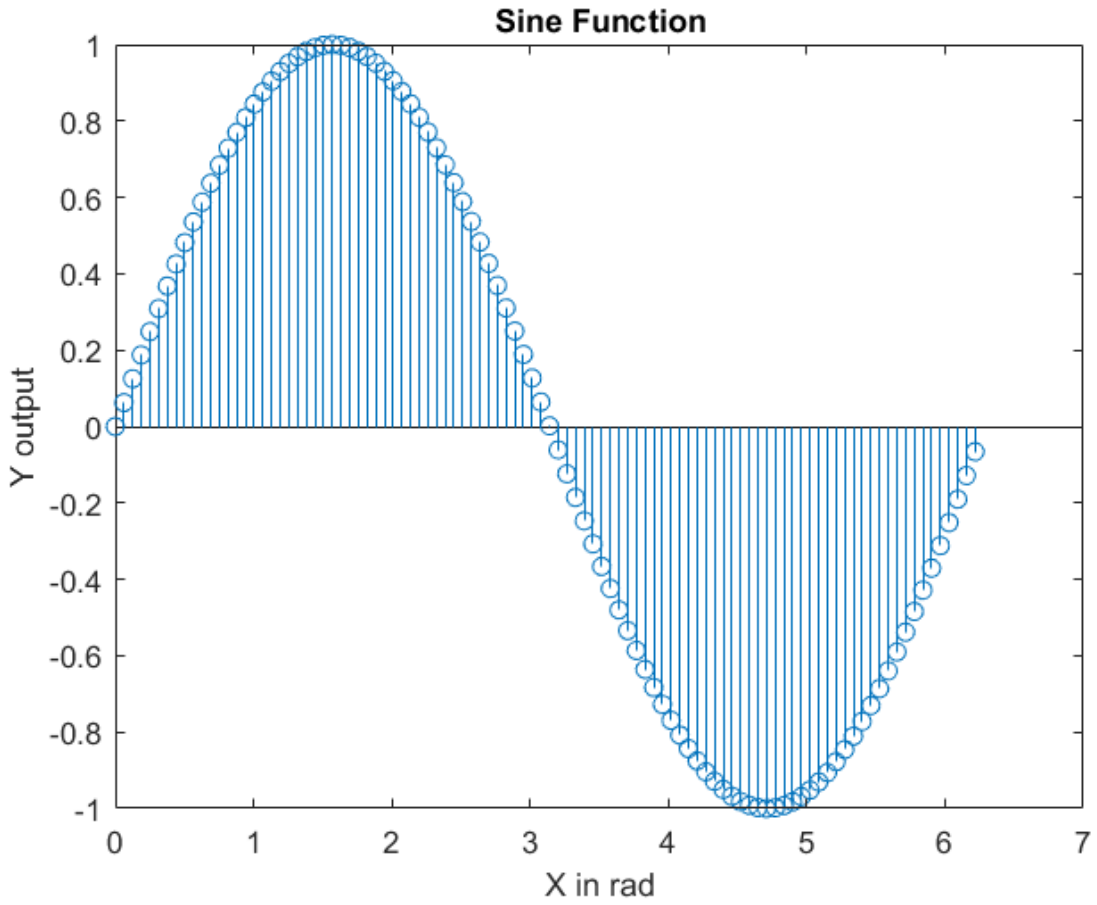

Fig: Discrete CMAC 1D function of Sin(x) over the interval (0-2π)

During the training, all the input values were mapped to the weighted cells by hashing, and the output was obtained as per the generalization factor. This output was compared to the actual output values and the error was then equally divided among the activated weights. This process was continued till the mean error was reduced to about 0.01.

At the start of the program generalization factor was set to be 1 and it was looped till value of generalization factor reached 29. During the training period, the graph of generalization

factor vs. mean error was recorded and the graph of generalization factor vs. time to convergence was also recorded as shown in figures below.

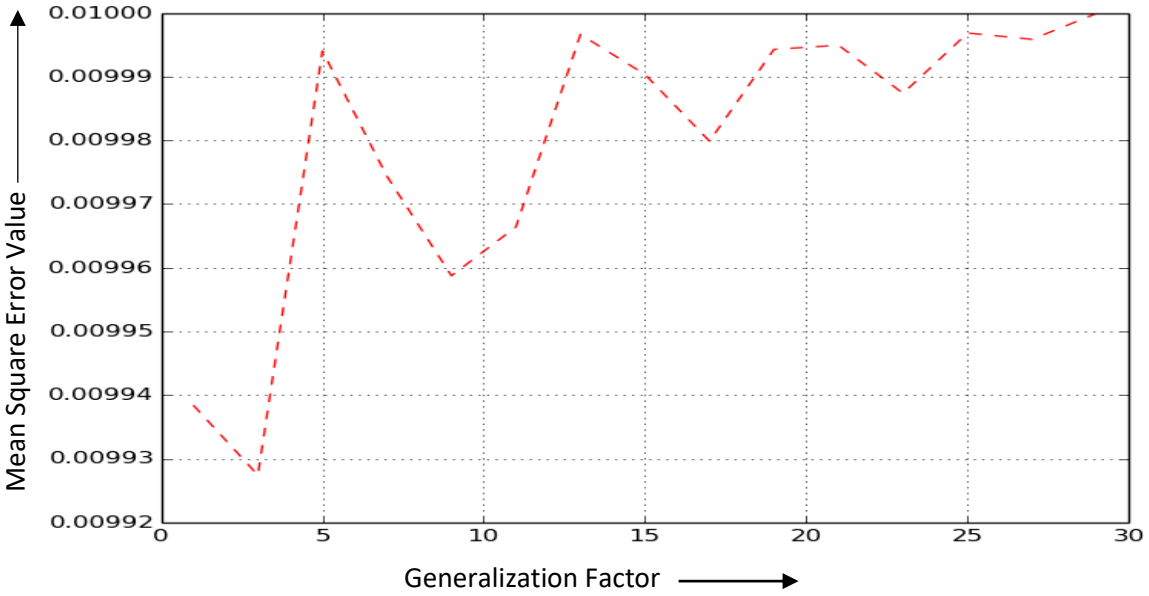

Fig: Mean square error wrt Generalization for training data

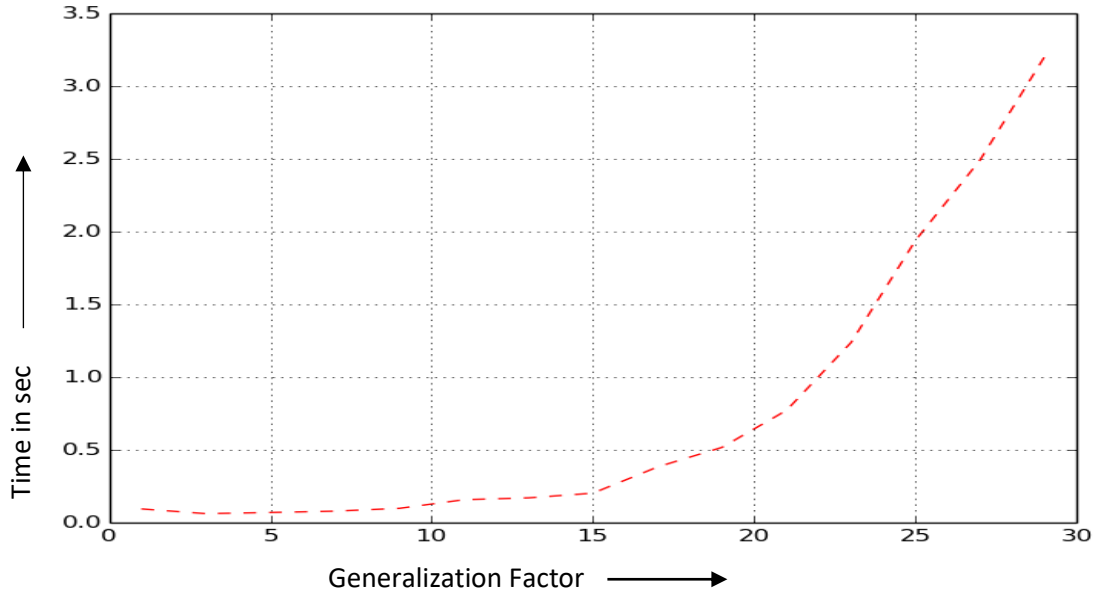

Fig: Training time wrt Generalization for training data

The effect of overlap on generalization can be described from the above 2 figures. General trend here is that mean square error is small for smaller generalization factor. This suggests that two inputs in nearby neighborhood i.e. increasing overlap will give nearly same outputs which is desired from the controller. On the other hand, if overlap is too large than it will

produce somewhat similar results for two inputs which are reasonably far from each other in the input space.

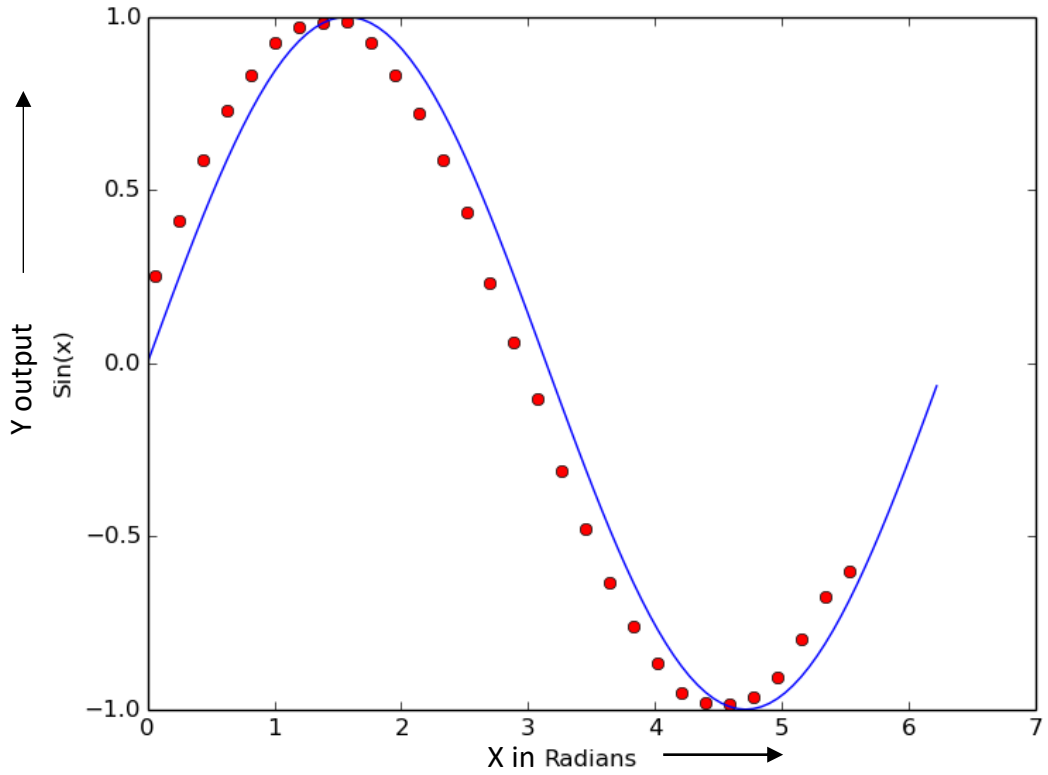

Fig: Comparison between actual data and test data processed by CMAC after learning

Time to convergence can be seen to increase with increase in generalization factor that is the convolution time increases. The result for test data on the trained CMAC in shown in the above figure. Hence, we can say that the CMAC has almost been trained for the sine function (but with some error).