ENPM808F Robot Learning

Homework 2- Report

1. **Program a Discrete CMAC and train it on a 1-D function (ref: Albus 1975, Fig. 5). Explore effect of overlap area on generalization and time to convergence. Use only 35 weights for your CMAC and sample your function at 100 evenly spaced points. Use 70 for training and 30 for testing. Report the accuracy of your CMAC network using only the 30 test points.**

   The sine wave over one entire cycle has been taken as an input (reference) curve for the training and testing of a 1-D CMAC. It is divided into 100 equal parts, with the width of each unit being equal to 0.0628 ((2 * pi)/100 = 0.0628). The value of the y co-ordinate for the curve is calculated from the sine function.

   Now, to prepare the training data, 70 random points out of the hundred are selected while the remaining 30 have been kept for testing. Now, since the generalization factor has to be an odd number to find the center cell and for the ease of calculation, it is divided by two and the integer part is taken which is added on both the sides of the cell under consideration. Hence, for the end points on both the sides, padding has been done. 35 random array of weights has been taken to train this CMAC.

   70 points have been taken for training, and the error is continuously calculated, and it is divided by the generalization factor for that loop. This updated error is then used again to find the correction factor which is added to the final value of weight and the value of weight is also updated. The mean square of the error is what keeps the training under control till the error reaches the value of 0.01.

   At the start of the loop, the generalization started from 1 and was continued till 35. Now, to get the idea of the influence of generalization factor on the output curve, three generalization factors of values 3,5 & 7 are considered, and three categories of updated weights are kept record of. These updated weights are then used in the trained model for the testing phase.

   In the testing phase, the 30 points are used in the pre-trained model to calculate the average weight using the updated weights and the generalization factor. An array out of that is created which would be used to create the final output graphs viz. sin(x) vs. radians, time vs. generalization factor and error vs. generalization factor.

   The graph of sin(x) vs. radians for three different values of generalization factor shows that as the value of gen increases, the accuracy of the curve decreases as it converges very less towards the original curve. The highest observed for gen = 3, while the maximum deviation is observed for gen = 7.

Dipam Patel
115809833

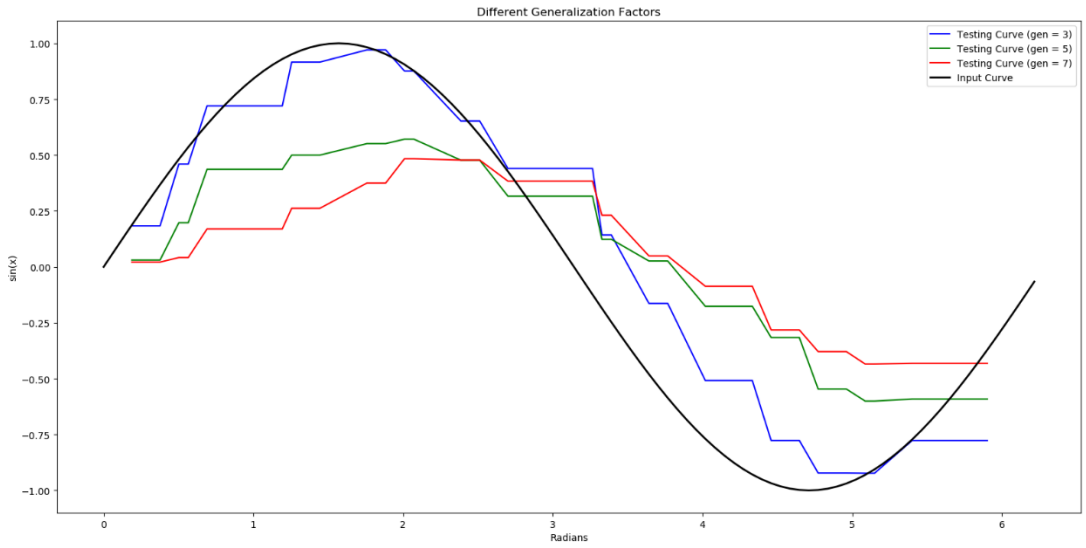

Figure-1: Graph of sin(x) vs. Radians for Different Generalization Factors

The graph of error vs. generalization factor shows the proportion of increase in error with the increase in the value of gen.

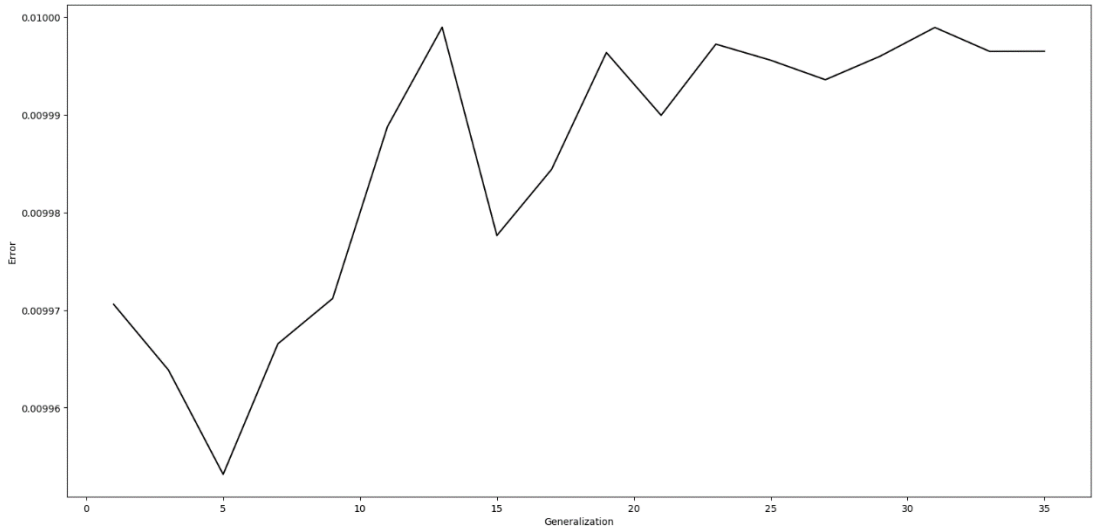

Figure-2: Graph of Error vs. Generalization Factor

And the graph of time vs. generalization factor shows that with the increase in the value of gen, the time increase nearly exponentially. Hence, it would take a long time for the graph to converge for higher values of gen.

Dipam Patel
115809833

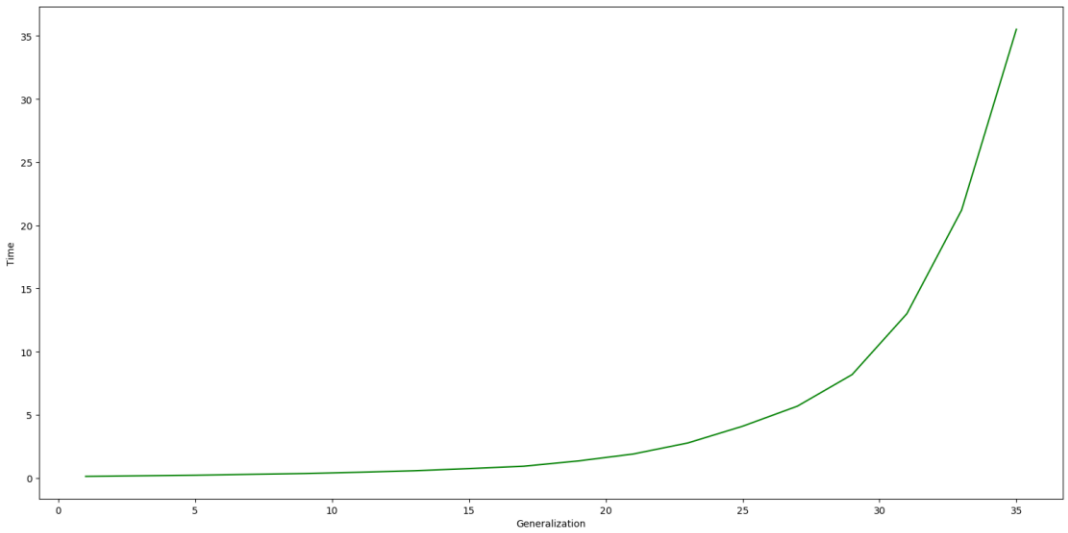

Figure-3: Graph of Time vs. Generalization Factor

The mean square error for smaller generalization factor is small. Hence, the overlap between two nearby inputs will give nearly same outputs. However, if the 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. This is evident from figure-2 and figure-3.

2. **Program a Continuous CMAC by allowing partial cell to overlap and modifying the weight update rule accordingly. Use only 35 weights for your CMAC and sample your function at 100 evenly spaced points. Use 70 for training and 30 for testing. Report the accuracy of your CMAC network using only the 30 test points. Compare the output of the Discrete CMAC with that of the Continuous CMAC.**

The continuous CMAC is nearly the same as discreet with the only difference being the end cells take part in the training only in some proportions instead of whole cells like in discrete.

Here, the top most cell is considered for its 80% share, while the bottom most for 20% to make them as one whole cell. This is what separated the continuous CMAC from the discrete one. Different proportions of cell division can be implemented.

In terms of output, it is visible from the graphs that discrete CMAC converges a little better than continuous one. One of the major factors being less major fluctuation in error in discrete. The time taken for convergence however, is nearly the same for both the cases.

Dipam Patel
115809833

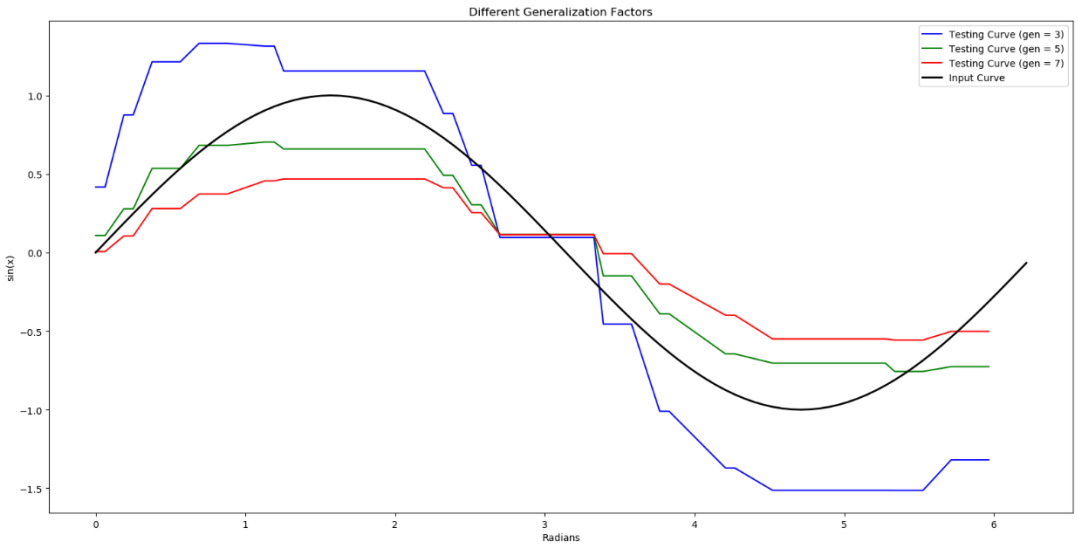

Figure-4: Graph of sin(x) vs. Radians for Different Generalization Factors

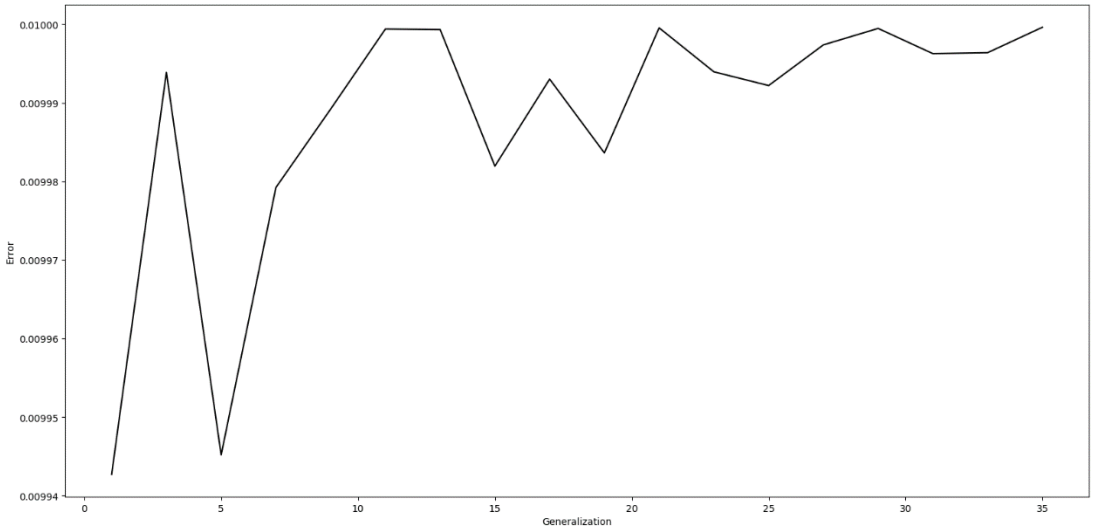

Figure-5: Graph of Error vs. Generalization Factor

The mean square error for smaller generalization factor is small. Hence, the overlap between two nearby inputs will give nearly same outputs. However, if the 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. This is evident from figure-5 and figure-6.

Dipam Patel
115809833

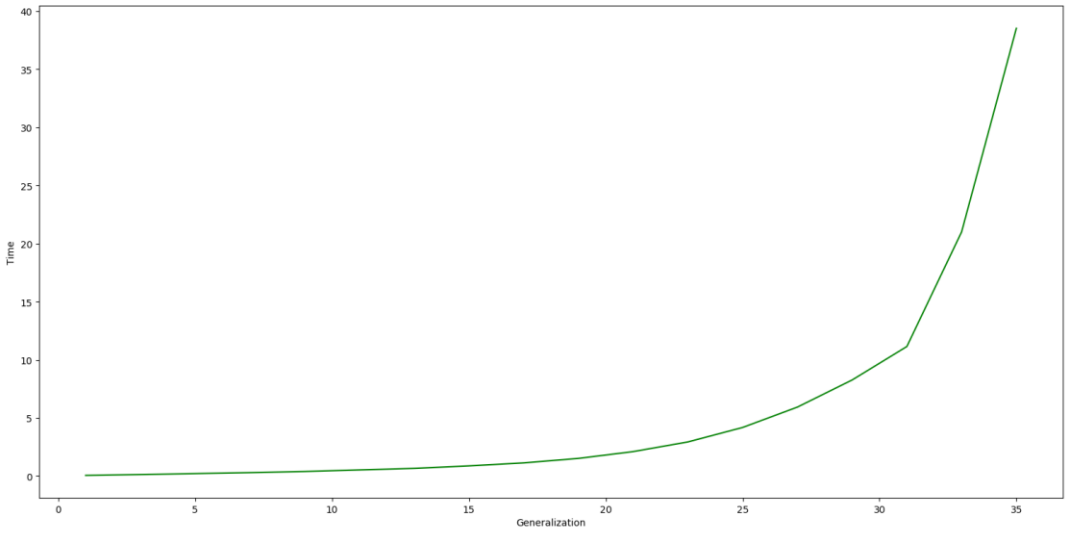

Figure-6: Graph of Time vs. Generalization Factor

3. Discuss how you might use recurrent connections to train a CMAC to output a desired trajectory without using time as an input (e.g., state only). You may earn up to 5 extra homework points if you implement your idea and show that it works.

Most the neural network learning methods, compute the gradients of the errors on the outer most layer once the model has reached its final value. However, computing the partial derivative of error of the temporal trajectory of the states weights of that network can allow us to perform gradient descent in the weights to minimize the error.

By adding feedback connections on the first layer to embed temporal relations in the network, the concept of recurrent neural network to only use state instead of time can be made possible. A Gaussian basis function can be used to model the structure and adjust the weight, while the gradient descent learning algorithm can be used to adjust the parameters. We can have a similar approach to use recurrent neural network to train the CMAC without using time as an input.

Dipam Patel
115809833

References

1. Barak A. Pearlmutter, School of Computer Science, Carnegie Mellon University, “Learning State Space Trajectories in Recurrent Neural Networks”
2. Jinzhu Peng, Yaonan Wang and Wei Sun, “Trajectory-Tracking Control for Mobile Robot Using Recurrent Fuzzy Cerebellar Model Articulation Controller”, *Neural Information Processing – Letters and Reviews*, Vol. 11, No. 1, January 2007
3. Chin-Ling LEE1 , Cheng-Jian LIN, “An efficient recurrent fuzzy CMAC model based on a dynamic-group–based hybrid evolutionary algorithm for identification and prediction applications”, *Turkish Journal of Electrical Engineering & Computer Sciences,*