Homework 2

ENPM690 - ROBOT LEARNING

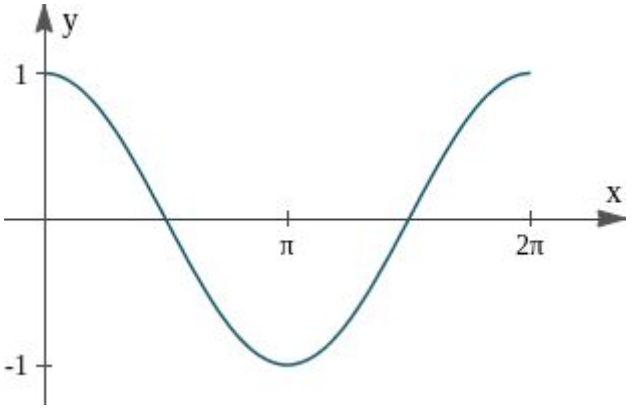
Submitted by

Nalin Das

UID - 116698290

M.Eng. Robotics

1-D Function used for training - Cosine

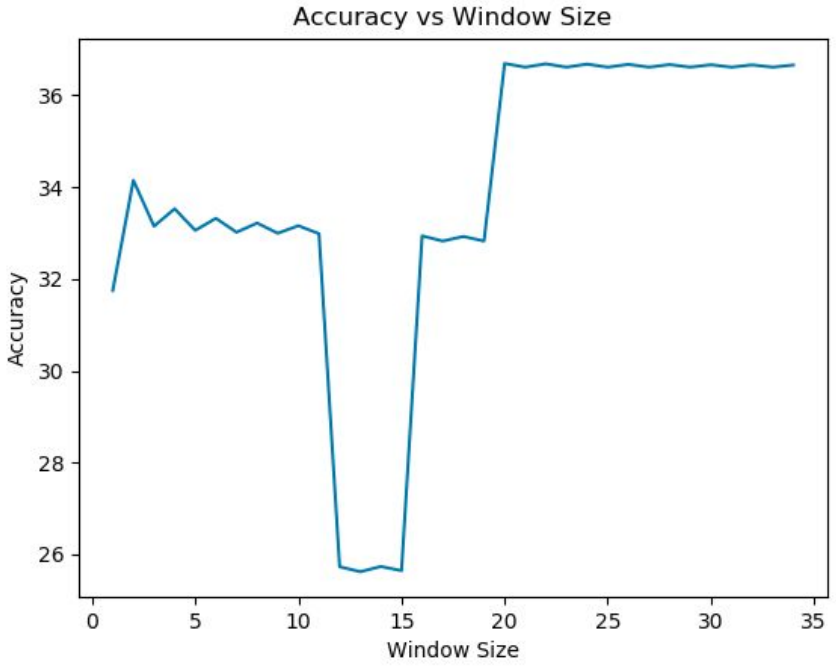


Algorithm:

1. 100 Sampling Points taken in total out of which some allotted for training and some reserved for testing.
2. Training input data taken from 0 to 2π with 70 equally spaced points.
3. Testing input data taken randomly with 30 points multiplied by 2π.
4. Trained the CMAC by creating association cells mapping with inputs and weights.
5. The predicted output was obtained after using the testing input.
6. Weights updated at each training iteration until error is below acceptable value or no of iterations has been completed.

Problem 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.

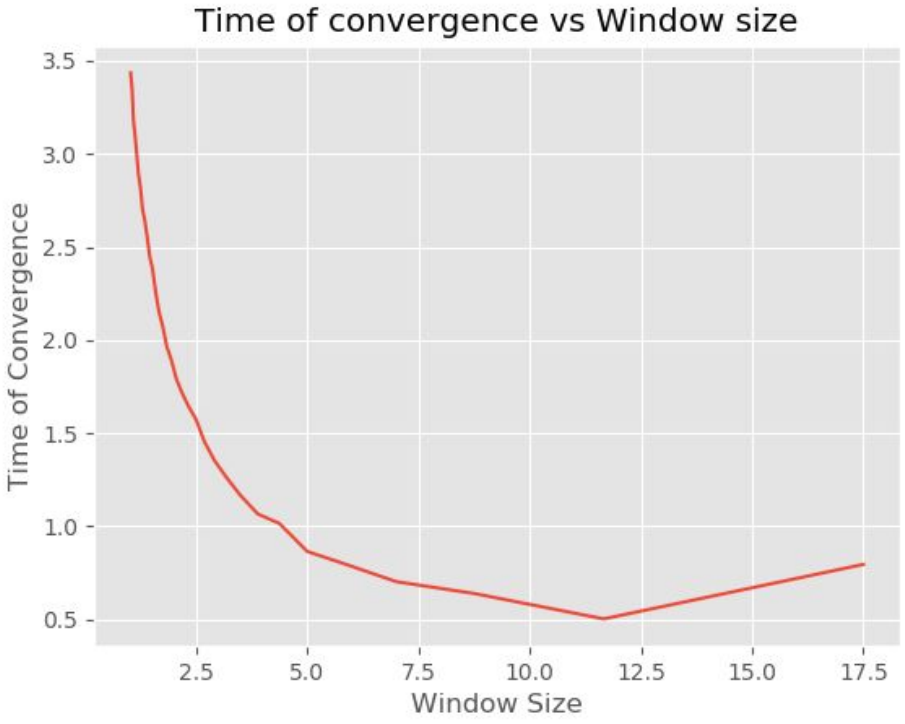
Overlap Area and Accuracy:



After training the CMAC, the accuracy is compared to w.r.t window size. It can be seen from the above figure that the accuracy of the model decreases as the window size increases and then increases

This is because as the generalization increases, change in any of the weights would have an influence on the neighboring weights, hence affecting the accuracy.

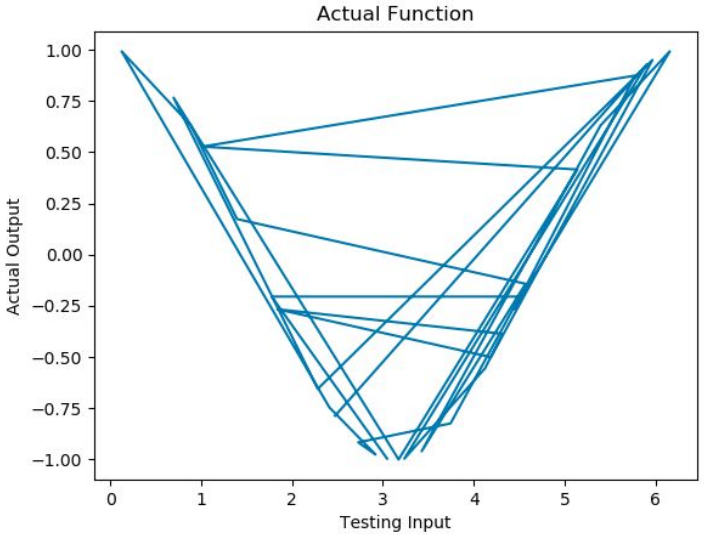
Time of Convergence:

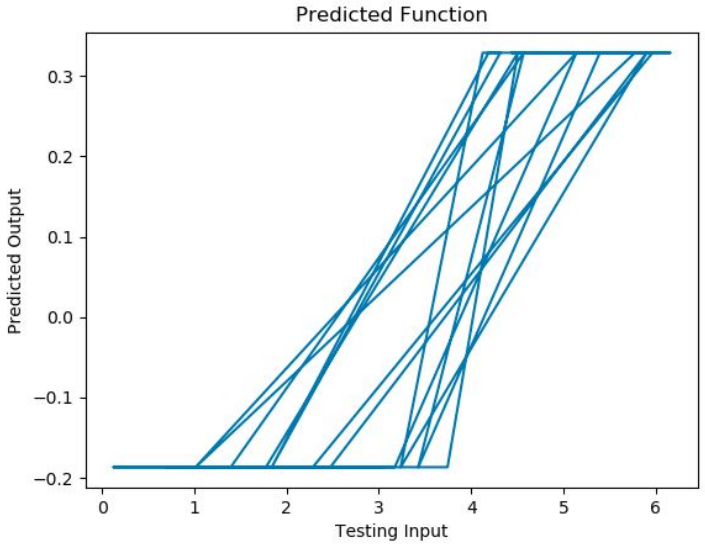


The time of convergence obtained first decreased and then increased with the window size increasing.

After this, the time of convergence started increasing as the mapping was no longer one to one but one to many.

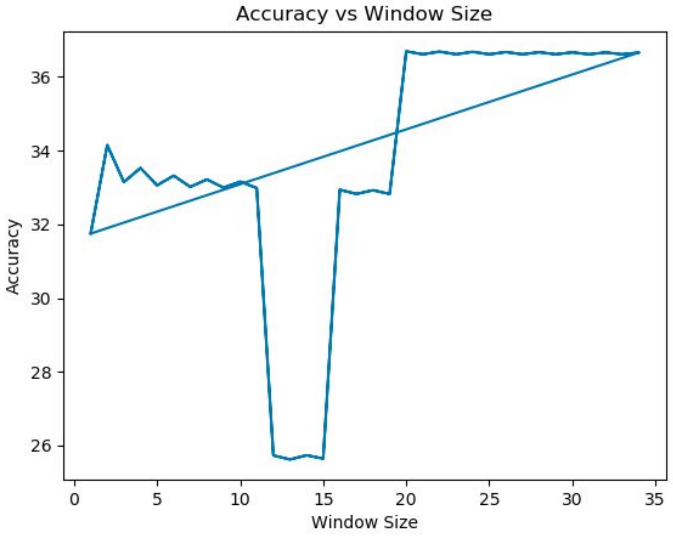
Plots for actual vs predicted function values are shown as follows:





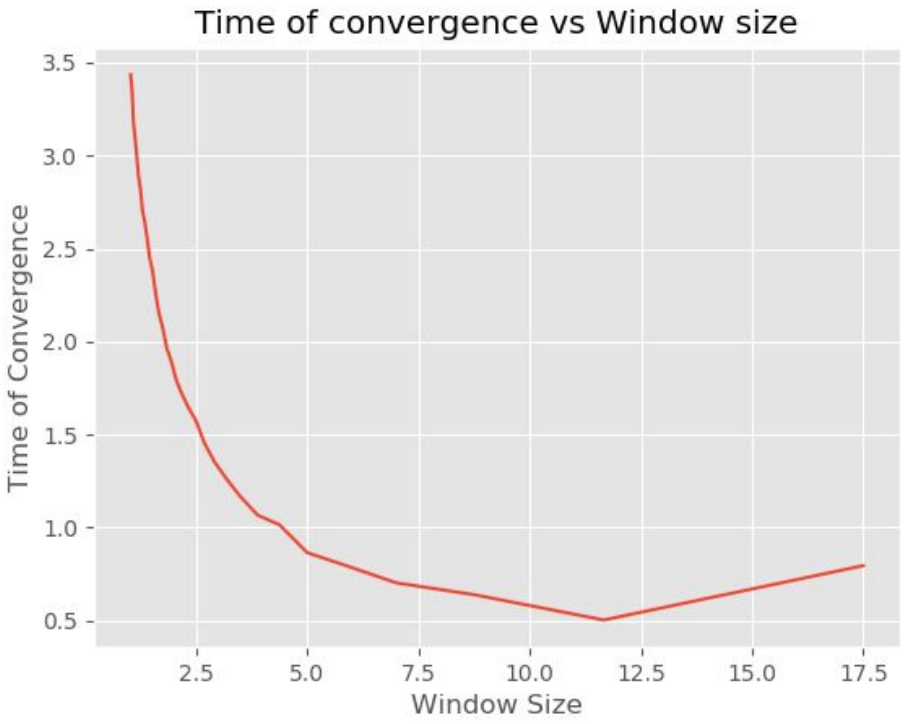
<u>Problem 2</u>: Program a Continuous CMAC by allowing partial cell overlap, and modifying the weight update rule accordingly. Use only 35 weights weights for your CMAC, and sample your function at 100 evenly spaced points. Use 70 for training and 30 for testing. Report the accuracy of your CMAC network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

Overlap Area and Accuracy:



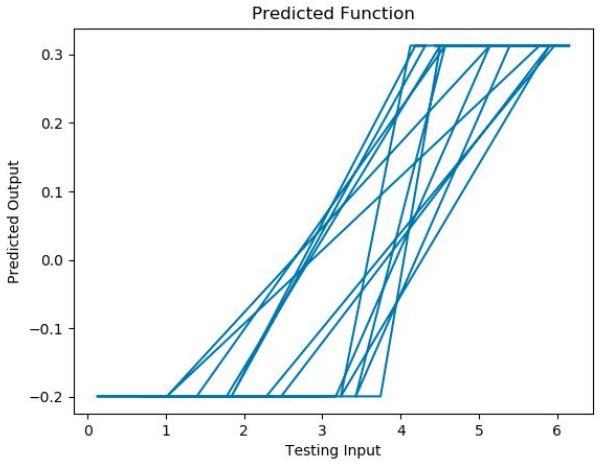
After training the CMAC, the accuracy is compared to w.r.t window size. It can be seen from the above figure that the accuracy of the model decreases and then increases as the window size increases because of generalization, similar to the discrete CMAC.

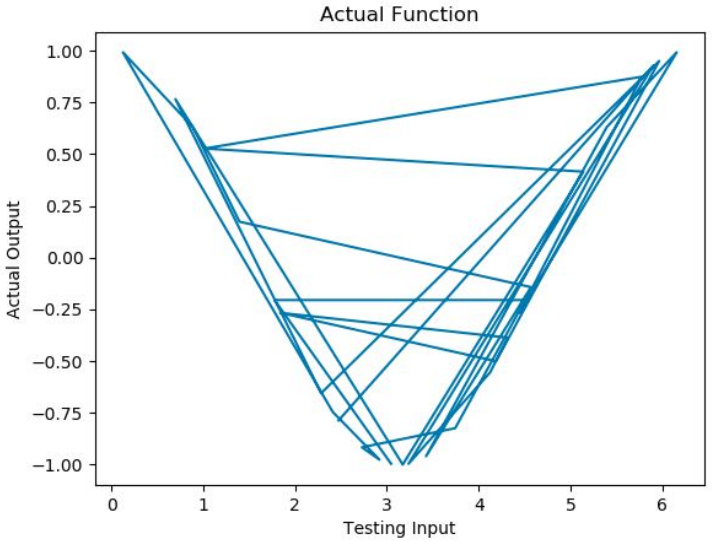
Time of Convergence:



The time of convergence obtained first decreased and then increased with the window size increasing.

Plots for actual vs predicted function are shown as follows:





Problem 3:Discuss how you might use recurrent connections to train a CMAC to output the 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.

Recurrent Neural Networks or RNN's are a type of neural network that allows the previous output to be used as the input for the next node.

RNN's are different from the standard Feed Forward networks in the sense that on top of remembering while training, it also learns and remembers prior inputs while generating the output. The advantage of an RNN is that it remembers all intermediary information. It possesses Long Short Term Memory or LSTM, which is useful in time series prediction. However, the training for an RNN is itself difficult.

To train the CMAC using the recurrent connections, a function must be added that depends on the output. This will make the function depends on itself, hence forming the basis of recurrent connections.

REFERENCES

1. Albus, J. S. (September 1, 1975). "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)." ASME. *J. Dyn. Sys., Meas., Control*. September 1975; 97(3): 220–227.
2. Sofge Local Learning and Differentiable CMAC
3. http://neupy.com/modules/generated/neupy.algorithms.CMAC.html#neupy.algorithms.CMAC
4. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
5. https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce
6. https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

Code Link:

https://github.com/nalindas9/enpm690/blob/master/Project_1/HW2_Code_Nalin_Das.py