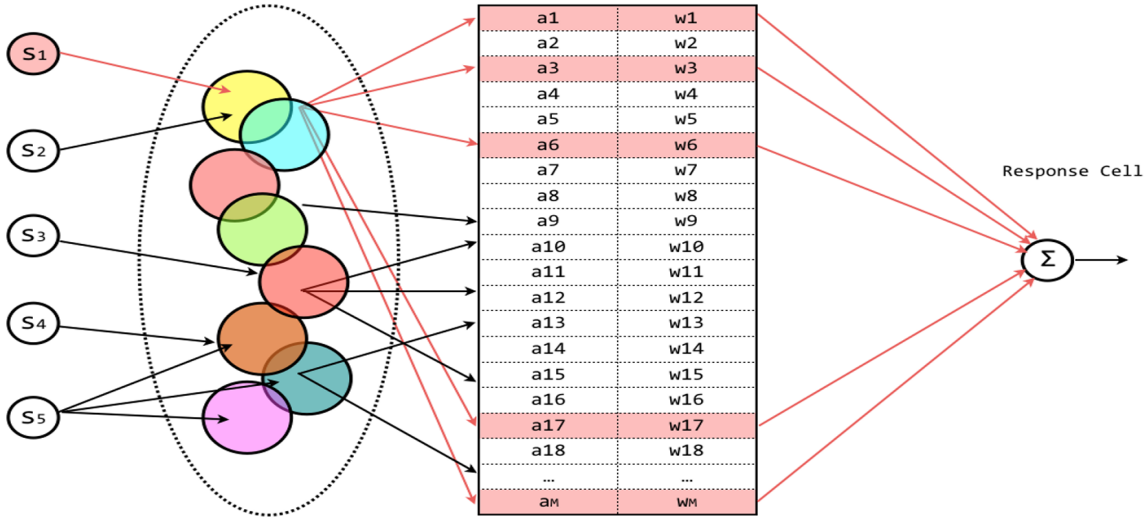
ENPM690 -Robot Learning

CMAC

Cerebellar Model Articulation Controller

Rahul Karanam 118172507



1. Formulate a robot learning task using machine learning terminology. Describe what are the inputs and outputs, and how and where the supervision takes part in.

Deep Q-Learning

Driving a self-driving car(a scaled down version) using a deep Q-learning model.

The inputs of this robot are basically three sensors such as left, middle and right sensors which will help the robot to navigate in the environment.

Input states:

Orientation of the robot(pose) , Sensor_1_signal , Sensor_2_signal , Sensor_3_signal

Output States(Actions) : Rotations or Velocity commands to the motor

Learning Strategy - This robot uses Q-learning as its model in order to learn the environment and be able to predict the output given the previous states. Q-learning is based upon a reward system where if a system is performed or traveled to a desired state then the reward is a positive one else we will punish the system by giving it a negative value. Basically, this model will generate a Q-Table which will be reflected to an action set such as going either directions such as left,right up or down. So after several iterations and trying to understand the environment by identifying obstacles and free paths based upon the reward based system, the system tends to perform better and better over more training data.

This was a similar model which I have done during my undergraduate project where I have a remote control car and I have attached Raspberry PI 3 for microcontroller, pi Camera for observing the environment , Motors, Ultrasonic Sensor and motor driver. The main objective of this project is to first take photos of the track while operating through the keyword, so after several thousands of photos. I have shuffled all the photos to the respective folders.

Then I created a supervised machine learning algorithm with 2 hidden layers where I have trained the cost function over 200 iterations to get the desired steering angle from the images taken from the camera input. The output of this model will be giving commands to the motor based upon the image input received from the sensor(camera).

2. Program a Discrete CMAC and train it on a 1-D function (ref: Albus 1975, Fig. 5)

Explore the effect of overlap area on generalization and time to convergence.

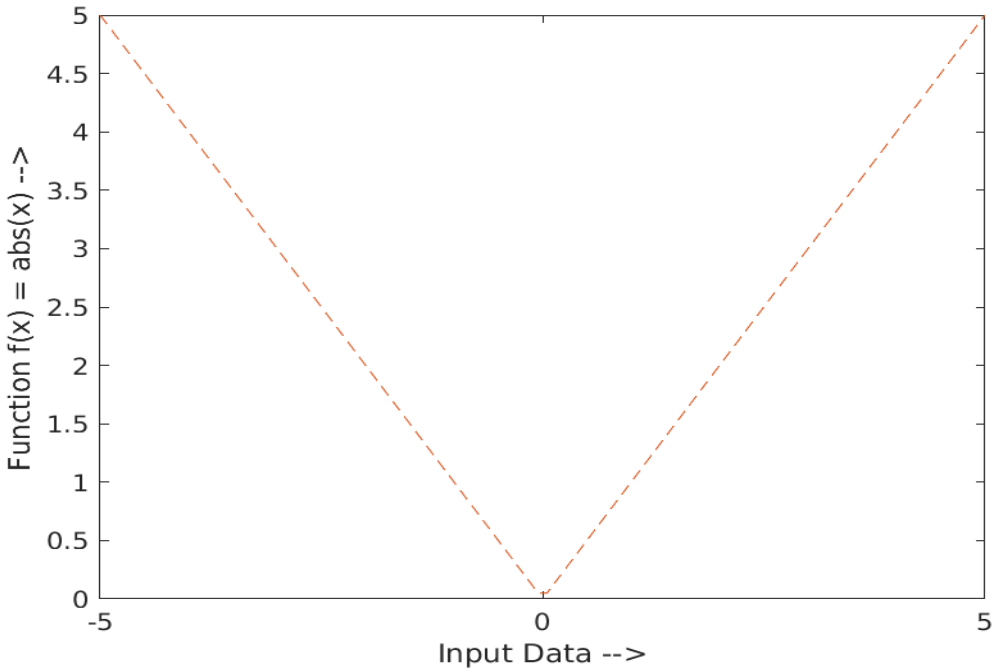
Use only 35 weights weights for your CMAC, and sample your function at 100

evenly spaced points. Use 70 for training and 30 for testing. Report the

accuracy of your CMAC network using only the 30 test points.

I have considered the absolute function for creating my discrete CMAC. I have generated the training data in the range of [-5,5]. Given below is the plot of function vs input data over the given range.

Function **f(x) = abs(x)** Range : [-5,5]



Furthermore, I have considered the input which is 100 evenly spaced points spread among the range.The weight matrix or the overlap area of the CMAC is spread from 1-34 as it has 35 weights associated with the input data. The output from the function f(x) i.e abs(x) is being computed over the input points which are associated with the weights.

First, we create the input vector using the number of inputs, number of weights and the generalizing factor for the associated weights. Then we tend to compute the mapping table or binary look-up table to check for association between the inputs and weights. Now we are ready to train our discrete CMAC using the 70 input points , the remaining 30 points will be used for testing our model.

Second, while training the model for 'n' iterations or epochs , we tend to calculate the difference between predicted output and our function output. The output is calculated by just adding up the values of the associated weights as the lookup table is binary.After calculating the error we add this error to update our weights for every iteration. The weights are updated according to the associated inputs during the training.In the end we tend to calculate the mean error and also we return the updated weights after the iterations.

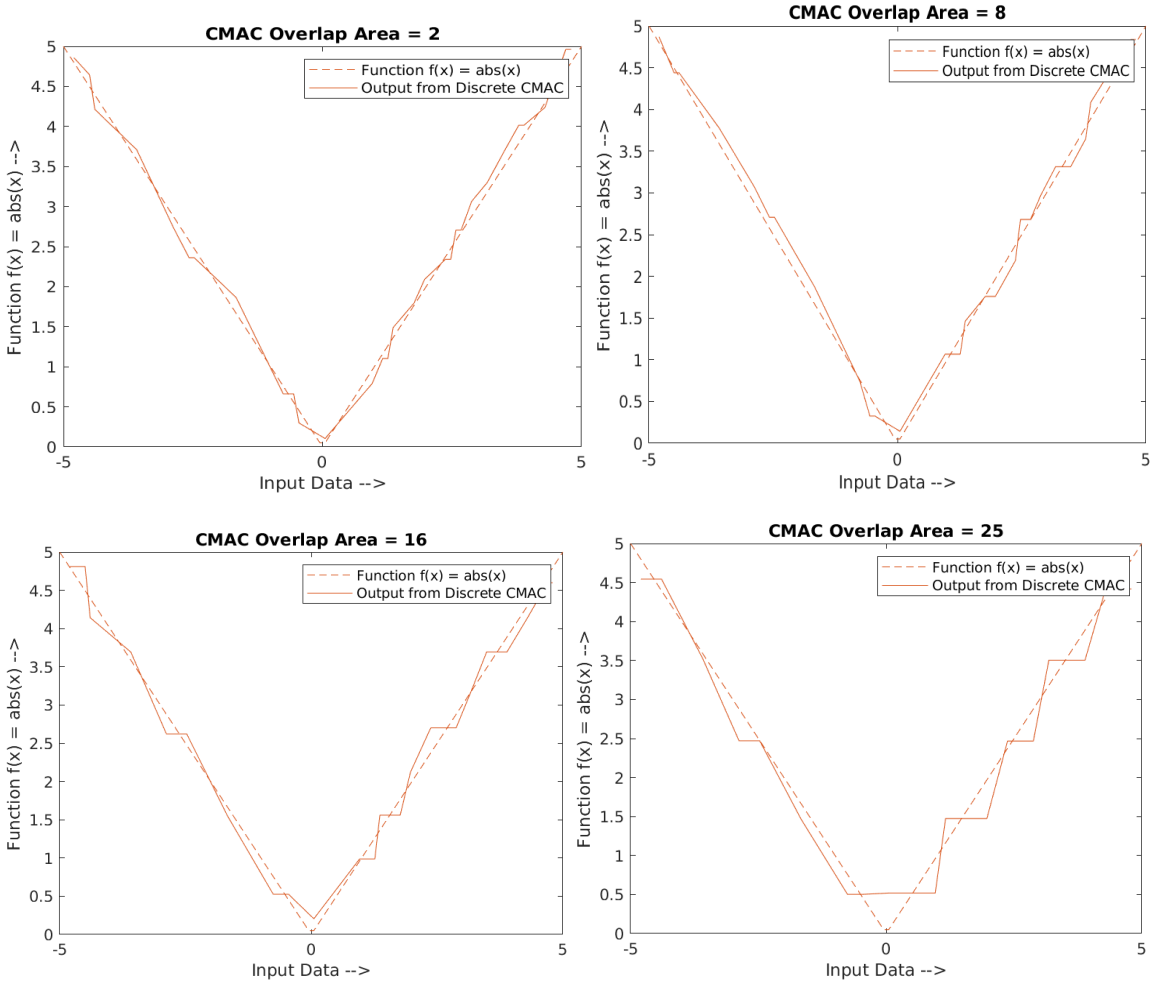
$Weight_{i-1} = Weight_{i} + error_{i} * LearningRate / genFactor$

Third, we will use the same procedure for testing the CMAC as well , the only difference is the number of datapoints used here as 30 points. The testing data will be similarly looked up from the table and is used to calculate the output based upon the associated weights(corresponding indexes) . I have used a different number of epochs but using 25 epochs has made the graph converge faster.

The following figure will explain about the overlapping area for each iteration. The accuracy of this model is quite okay but not better than the continuous cmac model as it smoothes the weights over previous iterations.

The output from the model significantly contains more overlap between the associated weights for each iteration and it decreases as the genFactor increases or the number of weights are increased

as more overlapping will cause the model to be prone to errors. Generalization is difficult to achieve when multiple weights are affected by change in a single weight.



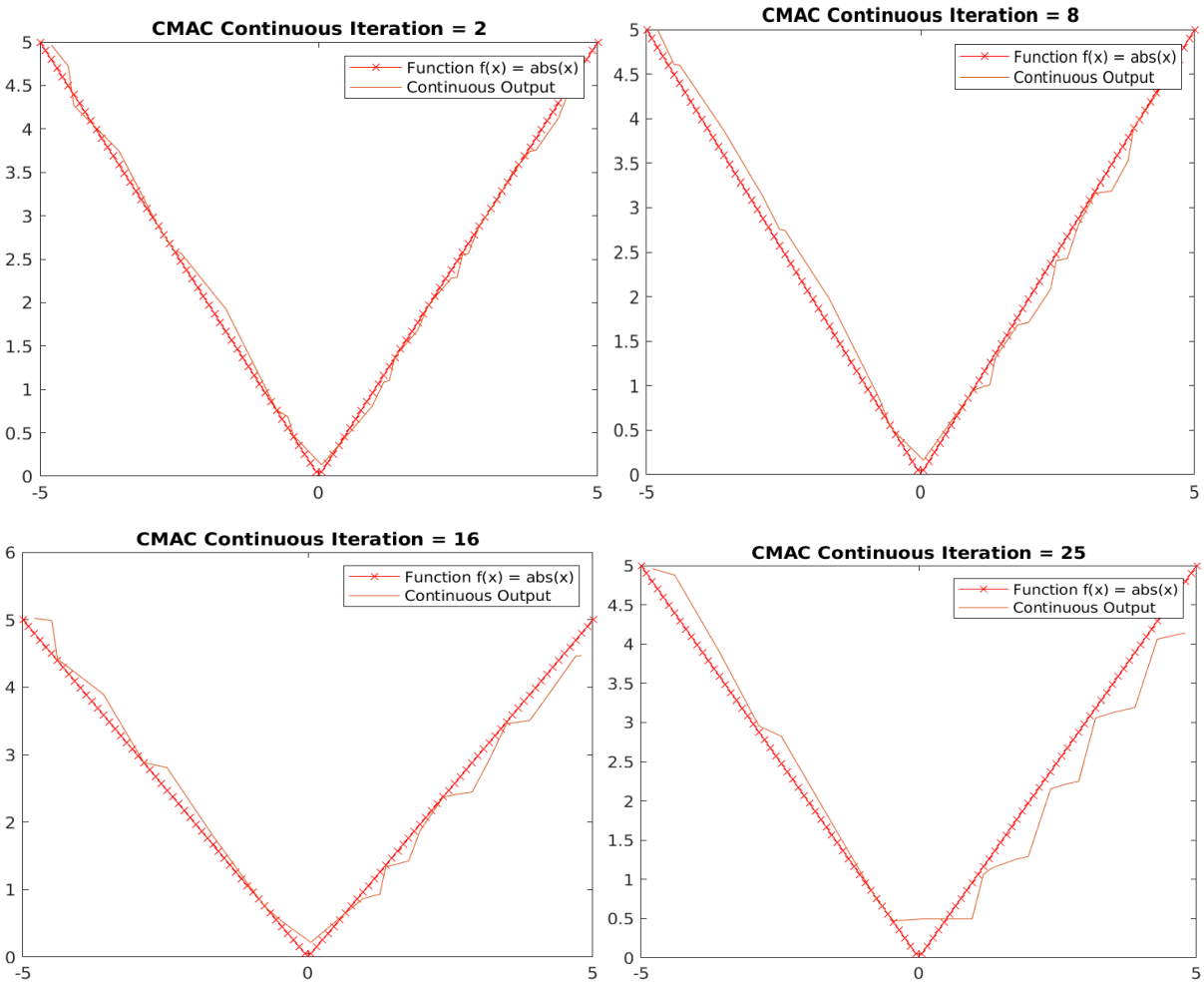
3. Program a Continuous CMAC by allowing partial cell overlap, and modifying the weight update rule accordingly. Use only 35 weights weights for your CMAC, and sample your function at 100 evenly spaced points. Use 70 for training and 30 for testing. Report the accuracy of your CMAC network using only the 30 test points. Compare the output of the Discrete CMAC with that of the Continuous CMAC. (You may need to provide a graph to compare)

I have used the similar structure for Continuous CMAC, the only difference is that while updating the weights over each iteration, we update it partially depending on the number of areas of overlap between the inputs.

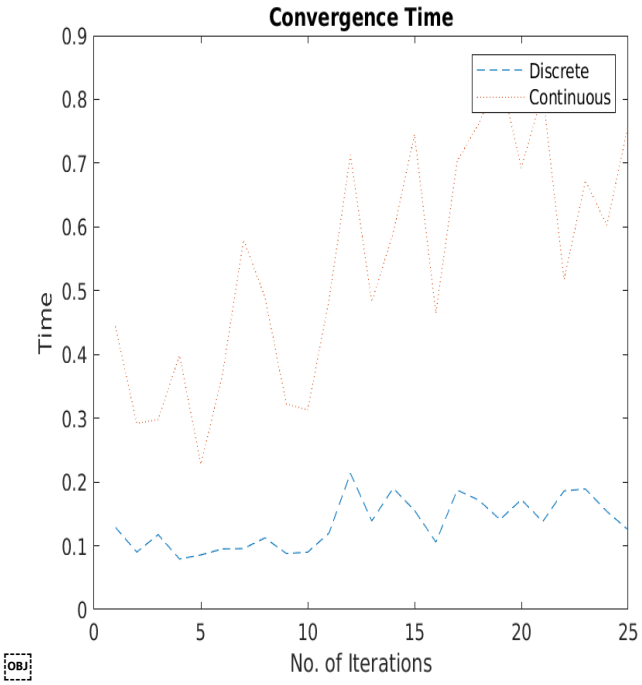
For example , for iteration no:5 , if there is overlap between the 3rd and 4th input cell , we find the error and then partially update the weight of the previous overlapped cell and the current weight by a proportion. If the value is 10.5 and this value lies in between two values such as 11 and 10 , i.e idx =4 and idx =5(values associated with the input vector) , then the output is a combination of partial updation of weights considering both the values i.e 10.5.

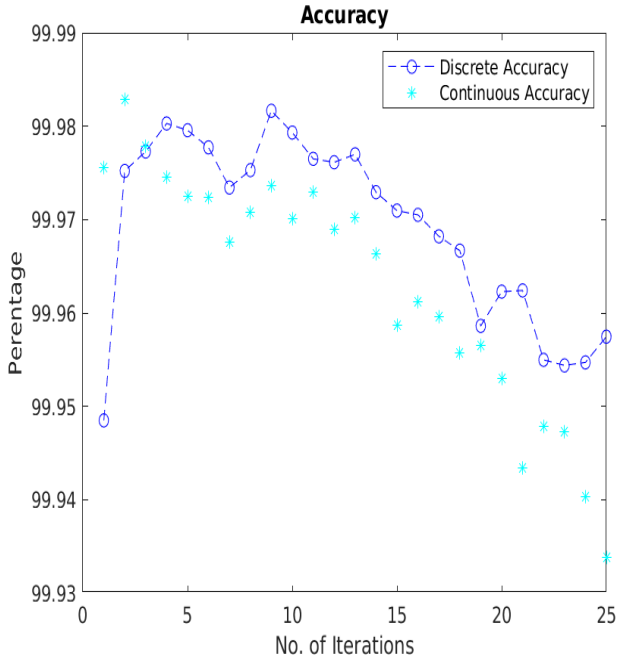
The training and testing for CMAC continuous model is quite similar to the discrete model.

Please find below the plot between the continuous cmac output and the function output over four different iteration or overlapping iterations.



Please find below the comparison between both discrete and continuous outputs and accuracies.





As you can see, the accuracy of the continuous model is better than the discrete model at the beginning due to the fact that the weights are updated parallel to each other(partially) which is maintaining the smoothness of the graph.

The convergence rate of the discrete model is quite faster than the continuous one due to more overlaps, the continuous model might take more iterations to converge.

4. Discuss how the results will change if the number of weights used in the model increases and what are the main disadvantages of CMAC.

Condition 1 : If the number of weights are increased

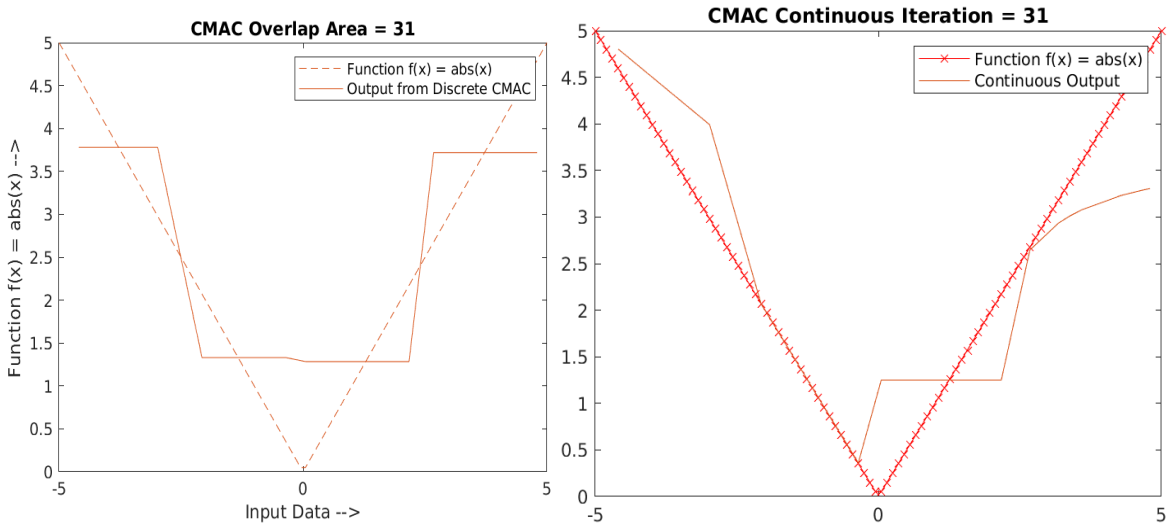
Condition 2 : If the number of gen_factor is increased or decreased

- If we make changes to our model based upon the above observations, we tend to see that it is difficult to generalize as there will be more overlap between input cells and associated weights.
- Change in single weight will affect changes in all other associated cells , which is called a global generalization error.
- If the genFactor is too low, then it is also a problem similar to the above.

Disadvantages of CMAC

- Computationally expensive as the number of weights will increase with the increase with the input data.
- Uses larger memory and needs more time to converge.
- Accuracy will decrease when the inputs are increased.
- Wrong Interpolation of some training data(sparse data)
- Creation of Virtual memory will increase the overall memory usage

For example look at the below graphs, I have increased the number of genFactor and also increased the weights to check how my model performs. It predicts or outputs erratic output which is because of the overgeneralization.



5. 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.

A: I have tried to implement the same using LSTM(Long Short Term Memory) and vanilla RNN(Recurrent Network), which will be helpful as it restores the previous memory.

Recurrent Neural Network a.k.a vanilla neural networks are good at predicting output based upon previous data but the issue is when the input is large or the dimensions of the input is increasing then using a Vanilla RNN will not be useful as the old information will be diminished as new data is being updated where the previous states are given less importance.

More advanced methods which will negate the issues with CMAC is to use an attention mechanism which basically takes care of association of inputs to the previous states maintaining the strength of the importance.

Using Transformers will make this model run for higher dimensions.

Please refer to the below articles which i have written during my personal time to get to know about this better.

Articles about the above mentioned methods

As I have mentioned, Using these above methods will improve my model drastically.