ENPM 690 -Assignment 1

Abhilash Pravin Mane
UID-117402865

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.

The task for the robot or to be precised quadrotor is to do a aggressive forest fight. This can be done by actually making an expert to fly the drone through forest at maximum velocity and data can be generated. Here the input is the camera sensor reading especially camera feed, imu data ,along with it the motor speed for the corresponding camera feed acting as the output. Then an Convolution neural network can be trained which takes in input from camera, IMU and outputs the value of corresponding motor rpms. This is a supervised learning task and needs lot of data gathering another method is to gather data using an expert algorithm like MPC and train the model (CNN model) on that data this is known as imitation learning. Advantage of this method is that no extra manpower is needed and this can also be done in the simulation environment reducing any collateral damage that might happen due to drone. The only supervision for the later method needed is to see if the device is working and occasionally check the data annotation. During testing the person might need to supervise and if the drone somehow decides the action that can cause damage the supervisor will take over the control.I

2) 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 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 function used for creating the dataset is $y=x^2$, where $x \in [0,10)$ with an increment of 0.1

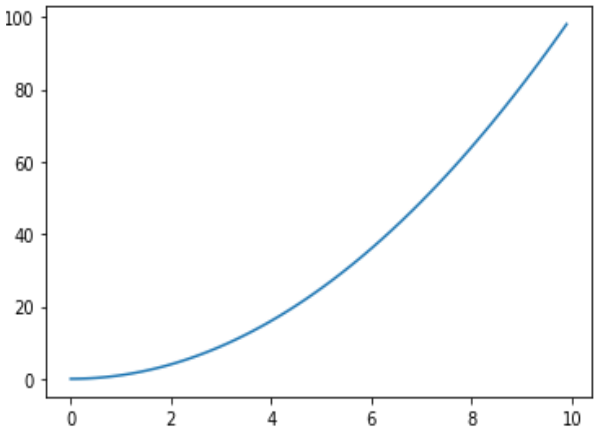


Fig1: Dataset

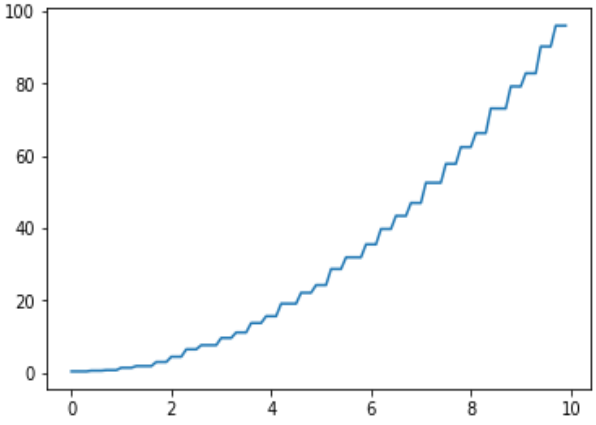


Fig2: CMAC output

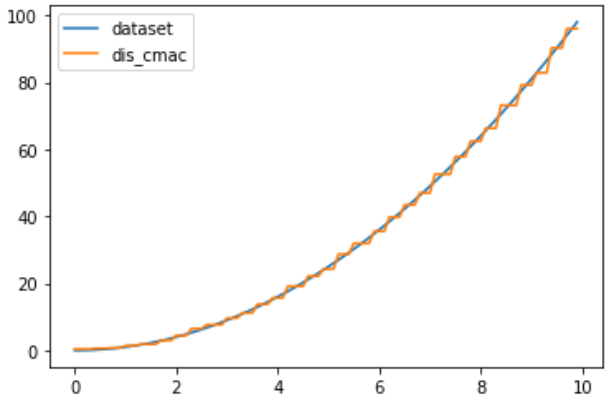


Fig3: Dataset +CMAC output overlap

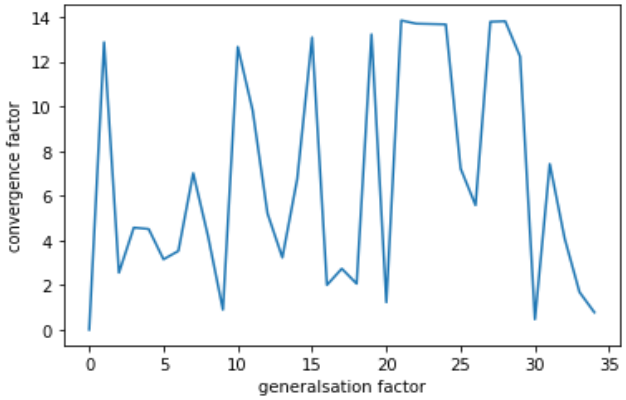


Fig4: Generalization vs Time to converge

For the code please refer - discrete_CMAC.ipynb in the submission
Acuracy for the discrete CMAC is 91.233%
As the generalization factor increases the prediction precision increases and the function becomes smooth and close to the datatset as evident in picture.

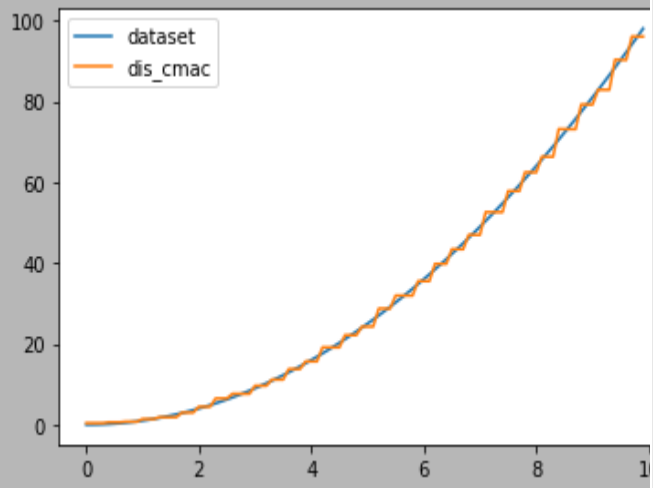


Fig1: CMAC - generalization_factor=5

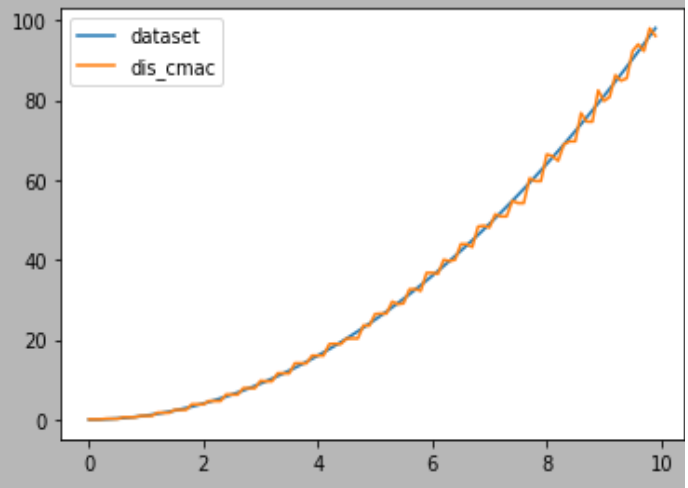


Fig2: CMAC - Genreralization_factor=2

Influence of generalization factor on CMAC

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

Method used for creating continuous CMAC and discrete CMAC 1) Moving Window has diven different percentage at the start and end elements 2) Using a Gaussian Kerenel like moving window[1]
Continuous CMAC approach 1 accuracy = 0.8967

Continuous CMAC approach 2 accuracy = 0.8978

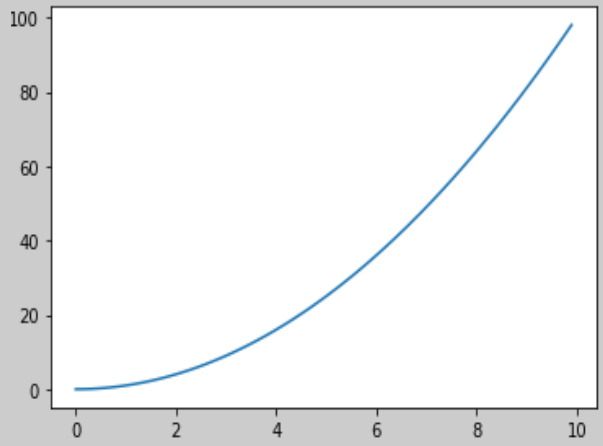


Fig1: Dataset

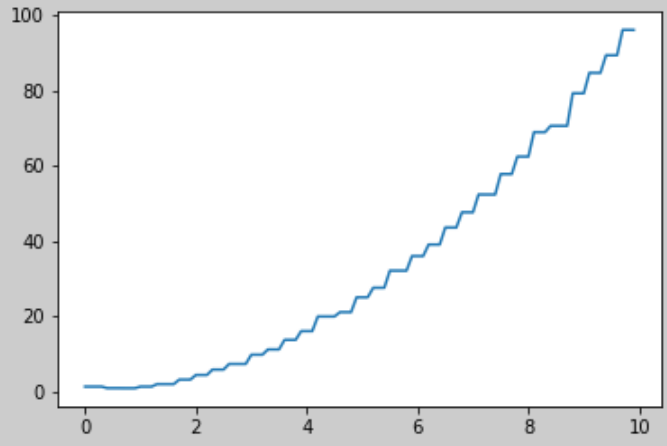


Fig2: Continious CMAC output

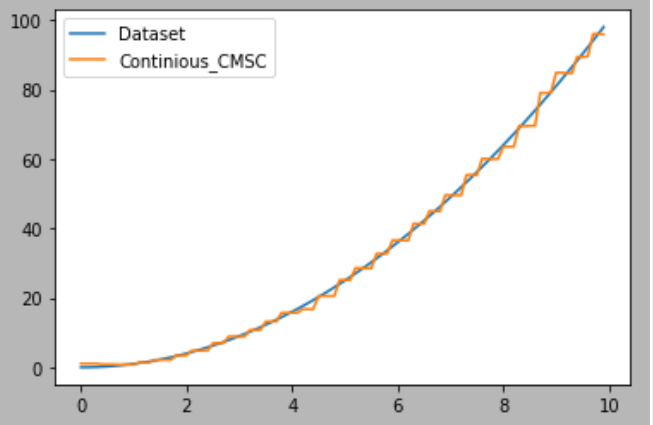


Fig3: Dataset +CMAC output overlap

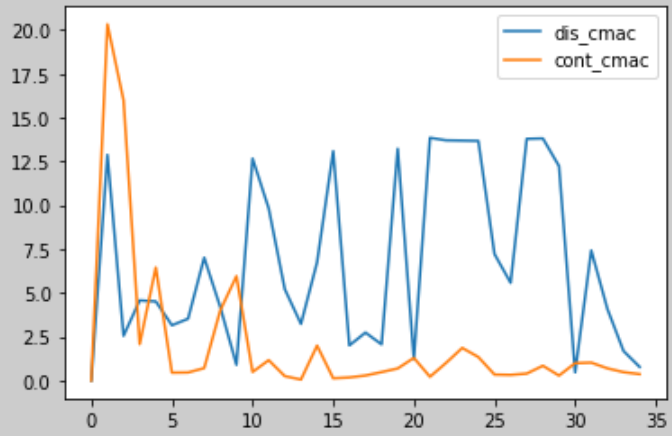
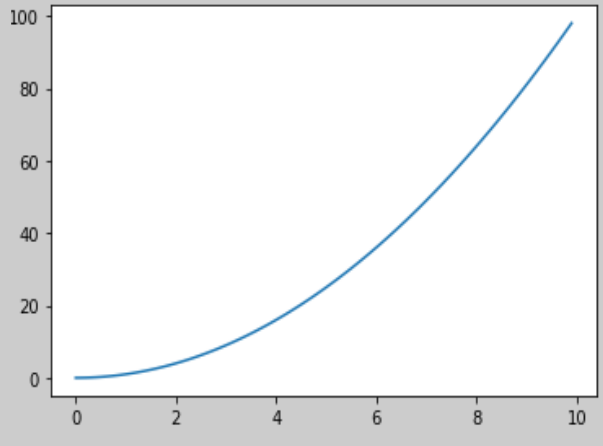


Fig4: Generalization vs Time to converge
(Comparison with discrete)

For approach 1



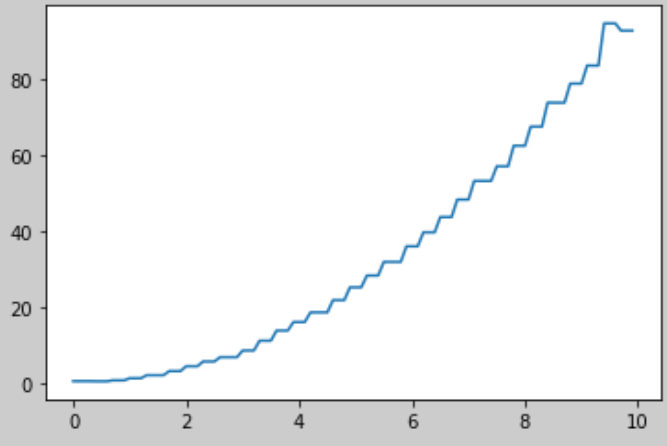
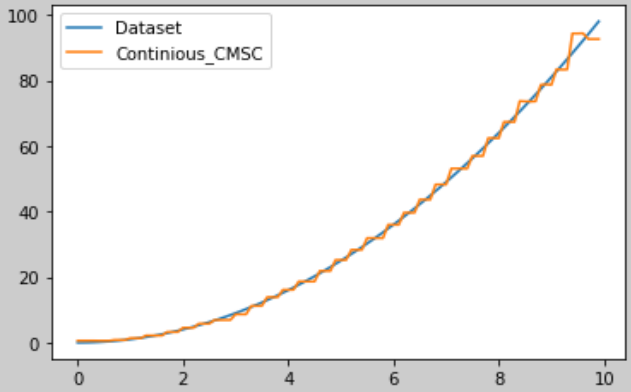
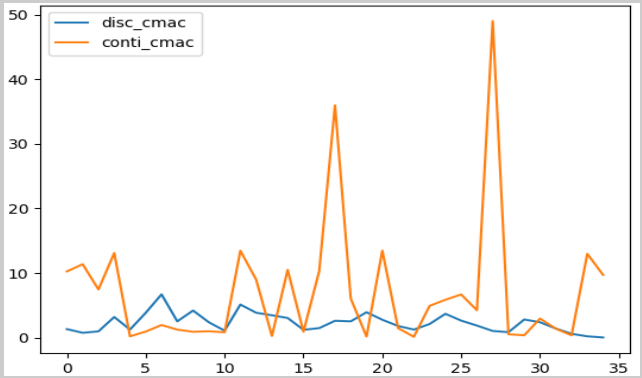


Fig1: Dataset	Fig2: Continious CMAC output
 Fig3: Dataset +CMAC output overlap	 Fig4: Generalization vs Time to converge

For approach 2

Comparision of output of all approaches -

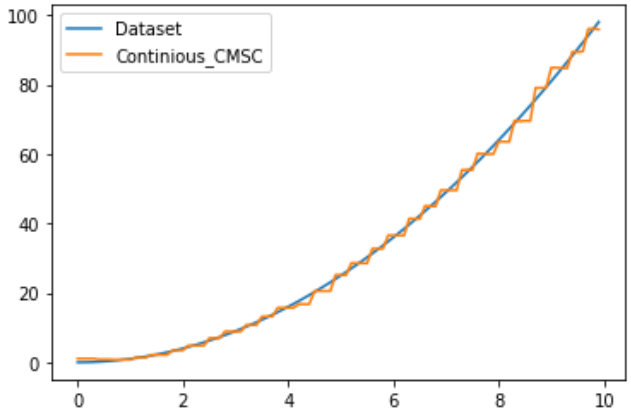


Fig1: Approcah1

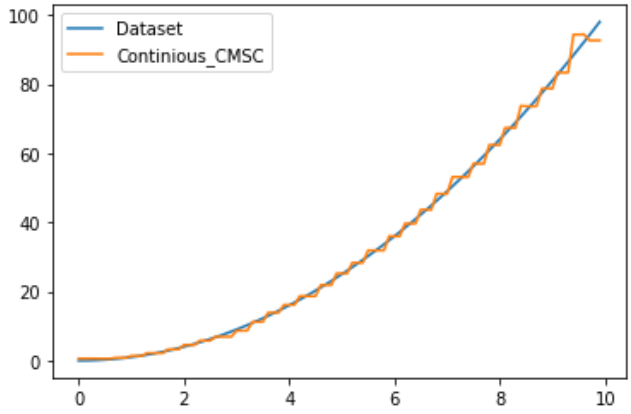


Fig2: Approach 2

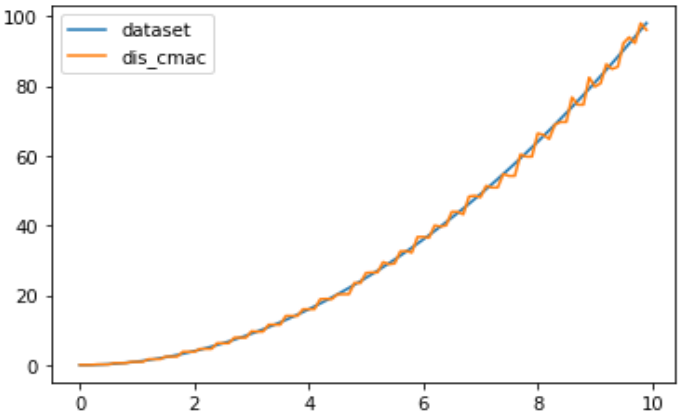


Fig3: Discrete Approach

Comparision for discrete,continuous , approach1,approach2

As it is evident that discrete CMAC is outperformed by both Continuous CMAC as it tries to retain the continuity of data .
In continuous CMAC - Approach where in gaussian kernel (approach2[1]) is used as a sliding window outperforms the Normal approach of reducing weight value of extreme element (approach1) as it considers the higher order derivative of the functions.
For the code please refer - continous_CMAC_class.ipynb in the submission for approach1
For the code please refer - continous_CMAC_class.ipynb in the submission for approach1

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.

As the number of weights increase the bin size decrease so we can incorporate more numbers For example if we have no_of _weights=5 for x= 1 to 10. A_x becomes [a,b,c,d,e,f] becomes , then 2 numbers are depicted by on weight. But as we increase the number of weights to 10 the A_x becomes [a,b,c,d,e,f,g,h,i,j] so no more binning or approximation is seen sand hence as the number of weight increase the model accuracy increases as the model get to see more data.
Disadvantages of CMAC -

1. CMAC is more memory intensive because of the CMAC model weights and also because of the representation of data.
2. Resource allocation has to be done manual;lly
3. It tends to perform poor generally when the dataset has an higher order terms(square ,cube, exponential etc) because of discontinuity and indifferentiablity of mapping.

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.

CMAC can be trained similar to RNN networks where in the output of the previous layer(here it is state) is feed along with input. So indirectly we have included the previous inputs. This sorts of help the CMAC to consider the current state and the input to make prediction of the next action /output. This prevents rash actions that can be predicted due to only inputs like abruptly increasing the wheels speed which can damage the robot. It is similar to the feedback controller which takes in the previous output along with current input and tries to rectify the error with small or incremental changing the output.

I have implemented the Recurrent CMAC. This is done by knowing the current input we try to predict the previous input ands current state. This current state along with the CMAC that tends to input are conctaenated to find the next state or the output. I have assumed the robot is moving in 2nd order path (x^2)

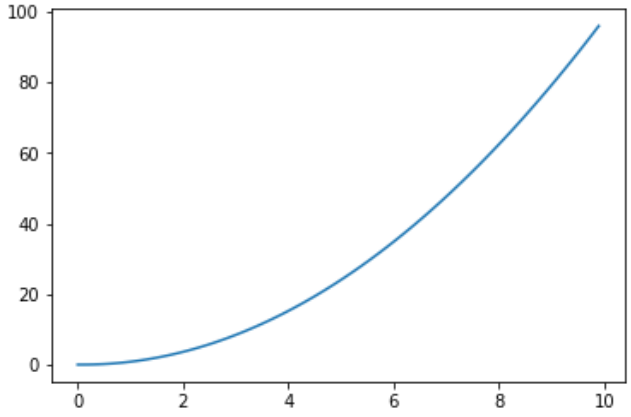


Fig1:Path

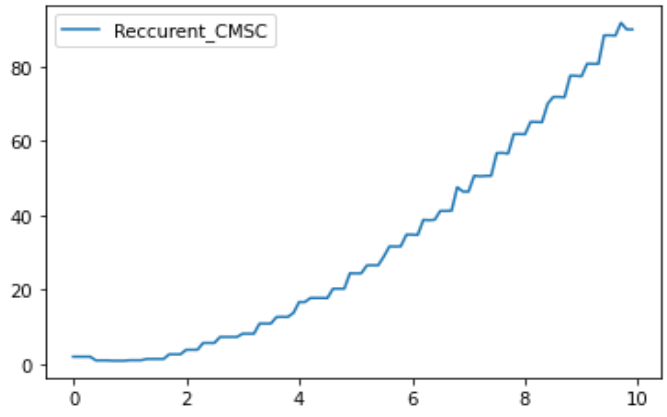


Fig2: Recurrent CMAC + original path

Fig3: Discrete Approach

For code refer recurrent_cmac.ipynb

Reference:

1. Lane, Stephen & Handelman, David & Gelfand, Jack. (1991). Higher-Order CMAC Neural Networks - Theory and Practice. Trends in Food Science & Technology - TRENDS FOOD SCI TECHNOL. 2. 1579 - 1585. 10.23919/ACC.1991.4791645.
2. Recurrent Neural Network