ENPM690 - Robot Learning

Homework 1

Koyal Bhartia (116350990)

Date : 18/2/2019

Contents

List of Figures

1 Question 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 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.

Following is the approach that was followed to program and train the CMAC network.

1.1 The 1D Function

The 1-D function considered for the training and testing of CMAC is

$$y = sin(x) \tag{1}$$

This function is made to vary from 0 to 2π which is divided into 100 equally spaced sample points as shown in the figure below.

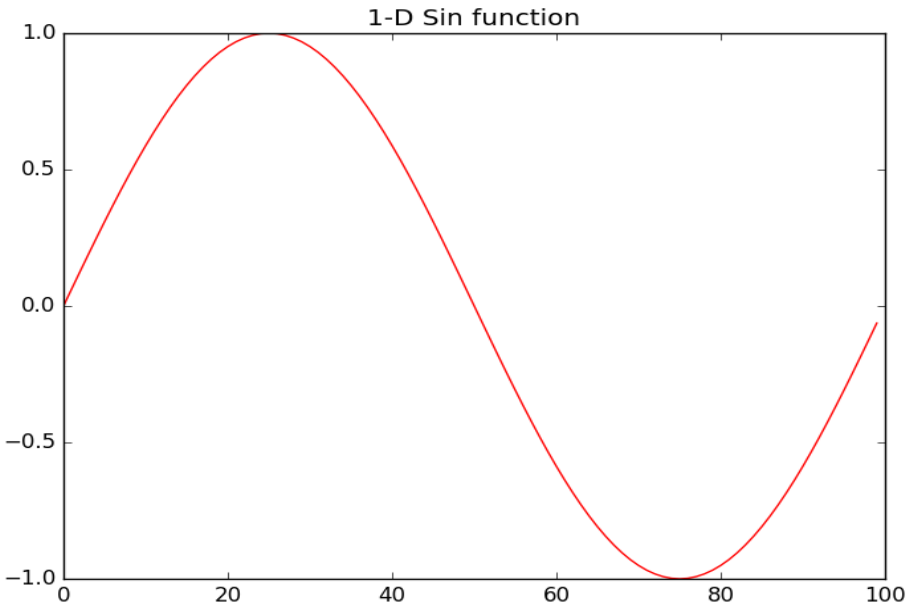

Figure 1: 1-D sin function used to train and test CMAC program

1.2 Data Preparation

The input data is randomly separated into 70% and 30% testing data. The following graph show the separated test and training data.

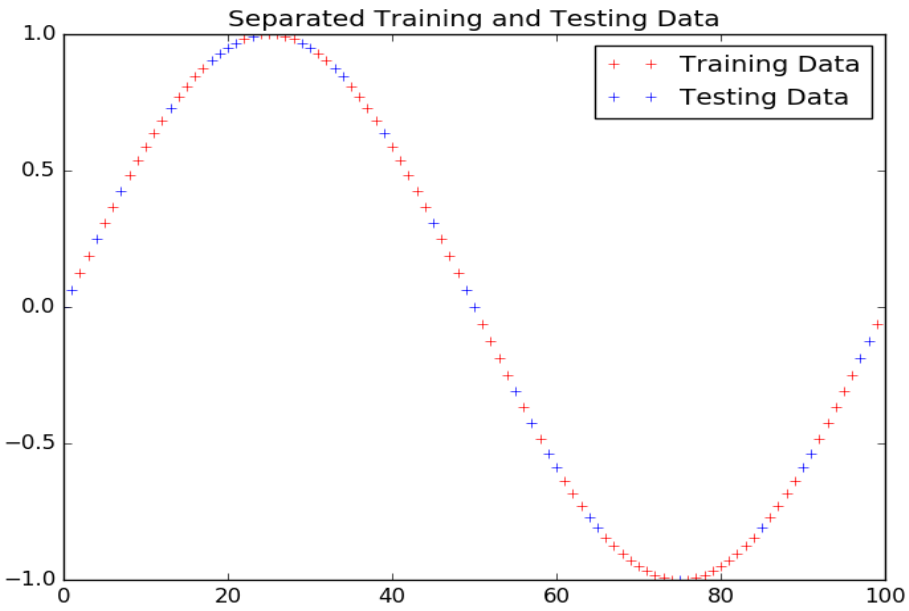

Figure 2: Separation of Train and Test data in 7:3 ratio

1.3 Training the CMAC

1.3.1 Mapping inputs to associative cells

This is one of the most primary steps in programming/training the CMAC. Each associative cell can be active/inactive based on which cells are triggered due to each input. Thus the relationship between the input vector and the memory cells can take the value of 0 or 1. The overlap between the associative cells excited by different input levels determines the generalization.

The programming of the CMAC done here has been tested for different levels of overlap. The **min possible is 1** which is almost like 1-1 mapping, and which usually gives the most accurate results but bad results for testing. This is like the case of overfitting. The **max possible value of overlap is 34** as the number of allowed weights is 35. As the overlap is increased though the accuracy decreases the convergence rate becomes faster. Hence choosing the amount of overlap is a design decision and a **tradeoff between accuracy and convergence.** The way I have chosen the associative cells has been shown below:

The generalization factor of β is made to vary from 1 to 34. An associative index is fetched for each of the input values. This forms a pointer. The associative cells above and below this pointer is selected as to be fired by the respective input value.

Fetching the associative index:

$$index\_pointer = floor(\beta/2) + ((35 - 2 * floor(\beta/2)) * input\_value)/100 \tag{2}$$

The value 35 is the number of weights to be taken for CMAC.

Once we get the above mid cell pointer, We then take equal no of cells to be activated above and below the pointer such that:

$$min\_cell = index\_pointer - \beta/2 \tag{3}$$

$$max\_cell = index\_pointer + \beta/2 \tag{4}$$

Thus in the discrete case of CMAC the associative cells selected for each input value are the cells between the **min cell** and the **max cell** values calculated above. Thus we have the mapping from the input vector space to the associative cells layer.

1.3.2 Determination of weights

Each association cell is assigned an address which is mapped via hash coding to a location in memory where the weights are stored. The aim of the training stage in CMAC programming is to determine these weights. The weight is continuously updated until the **no of iterations reaches 5000 or the difference between the previous and the current error reaches lesser than 0.00001. i.e converged.** This is repeated for an overlap ranging from 1 to 34.

The weights are updated using a **learning rate of 0.05**. The total weight is calculated for each input using Eqn(5). The error is then calculated as the difference between this value and the fixed output values of the sin function. This error divided by the overlap/generalisation factor gives the correction factor for the weights. This is shown in the code where the correction is then multiplied with the learning rate to get the final weight. The current error is then calculated using 6.

1.3.3 Determination of the output using weights

Once we train the CMAC which gives us the weights of the associative cells, the weights of all the active memory locations are summed to produce the output i.e

$$y^j = \sum_{iactive} w_i x_i^j \tag{5}$$

where x_i^j is 1 for i active and 0 otherwise.w_i is the weight of the ith memory location and the subscript j denotes the jth input pattern or level.

1.3.4 Error Calculation

The difference/error between the output calculated using weights and the pre-defined is calculated using the mean square error rule.

$$MSE = (Sum\_of\_weights - Output\_value)^2 \quad (6)$$

1.4 Testing the CMAC

The testing is performed by comparing the output received by doing a sum of weights and the the actual output of the sin function. The accuracy calculated as 100-error.

```
Accuracy of Discrete [92.60275798034147, 96.59056926223896, 96.58010522678089, 96.68091632499109, 96.71
25530959319, 96.7424268483404, 96.81531327302662, 96.64500675855317, 96.72020897109978, 94.784445892863
8, 94.70584549244374, 94.59983210428467, 94.60528941361436, 94.32856042976066, 94.32944315898392, 93.83
933841684262, 93.84209567246575, 93.63328209987115, 93.64263108401605, 89.2291565017278, 89.22184043843
187, 89.88821371690227, 89.86122717394083, 91.75486798277329, 91.71263272819657, 90.22902145965384, 90.
27354291121128, 85.38817881445485, 85.65333587108852, 75.55106135291992, 75.84488289425852, 81.75216290
558458, 81.73546176927172]
```

Figure 3: Accuracy readings for the 30 test points : Discrete

As seen from the accuracy readings above, **the highest accuracy for discrete was obtained for a overlap factor of 7 i.e 96.8. The lowest accuracy is obtained fro a overlap factor of 31 i.e 75.5. Thus the accuracy has decreased with the overlap.**

The performance and the rate of convergence of the error has been visualised for generalization or overlap factor of 5 as shown below:

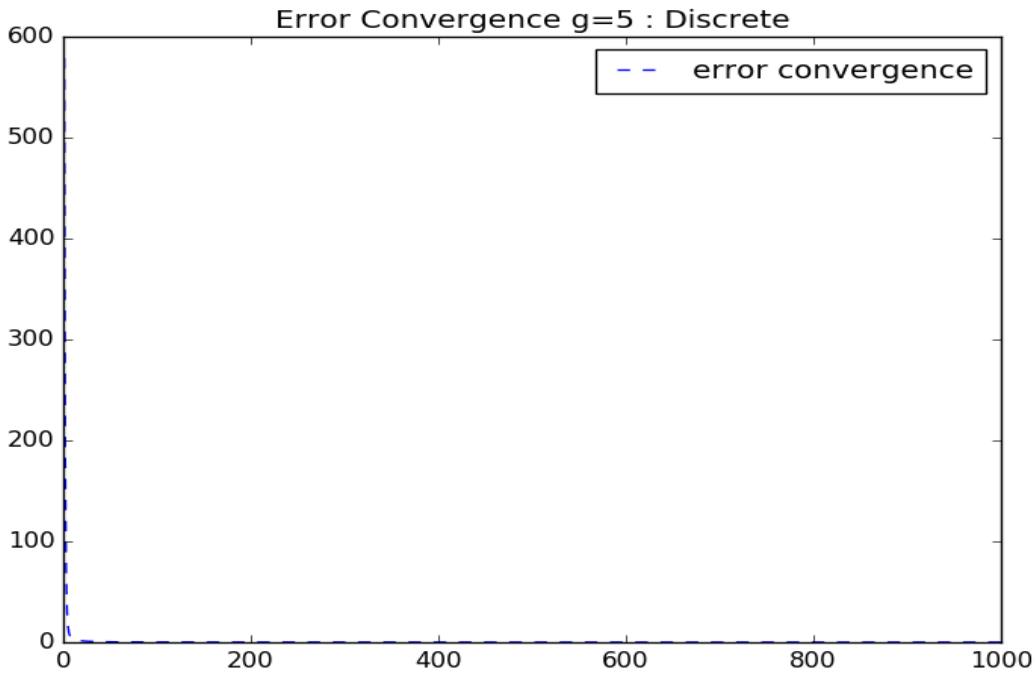

Figure 4: Error Convergence for g=5 : Discrete

Shown below is the CMAC predicted output for the the 30 test points. This graph as been shown for a feel of error / accuracy between the actual output and the predicted ones. The red dots are the predicted output for the corresponding blue "+" beside it. As seen in the graph the predicted ouputs are quite close to the actual outputs.

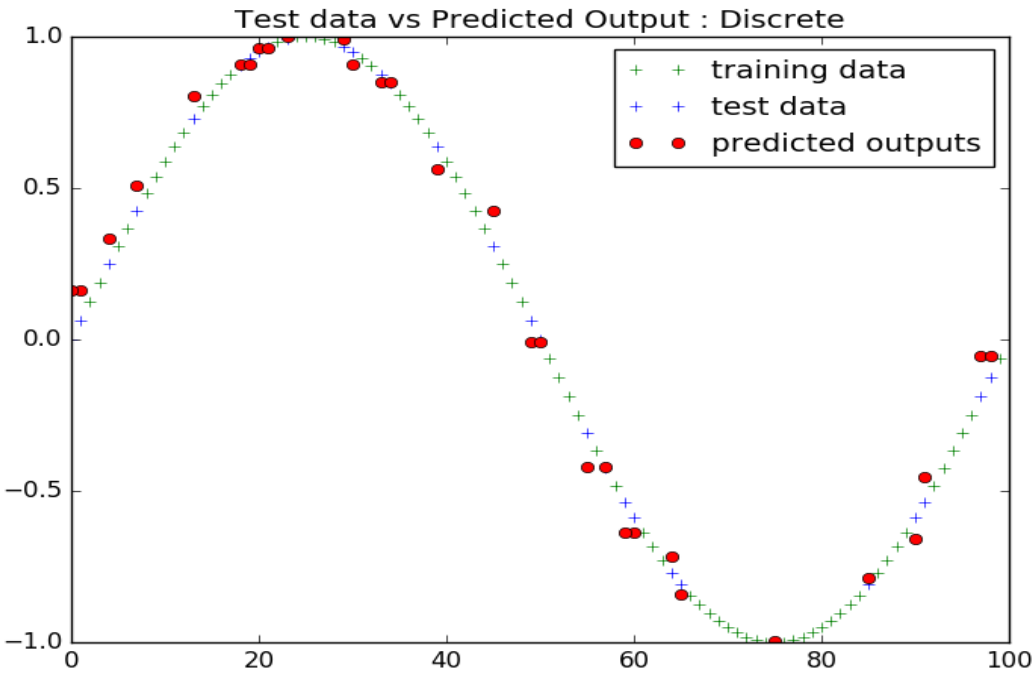

Figure 5: Test data vs Predicted Output for g=5 : Discrete

```
Time of convergence of Discrete [0.1434319019317627, 0.8778634071350098, 1.183518648147583, 1.884057044
9829102, 1.877746343612671, 2.5239009857177734, 2.4916160106658936, 3.108970880508423, 3.11781668663024
9, 3.7086379528045654, 3.7531306743621826, 4.340176343917847, 4.3308563232421875, 4.991700649261475, 4.
971181392669678, 4.5287253856658936, 4.728289365768433, 5.8949973583221436, 6.179940938949585, 6.291933
298110962, 6.589343786239624, 6.1030755043029785, 6.333025217056274, 7.034682512283325, 7.2005569934844
97, 6.682766914367676, 6.919987440109253, 6.3900697231292725, 6.4944353103637695, 7.693357467651367, 7.
735774993896484, 2.5351505279541016, 2.639143466949463]
```

Figure 6: Time of Convergence readings for 30 test points : Discrete

As seen from the time of convergence readings above, **the highest convergence for discrete was obtained for a overlap factor of 32 i.e 7.73. The lowest convergence is obtained fro a overlap factor of 1 i.e 0.143. Thus the convergence rate has increased with the overlap.**

2 Question 2

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.

The 1-D function that has been used for the Continuous CMAC is same as the case of Discrete. The training and testing set used is also the same as already shown earlier.

2.1 Training the CMAC

2.1.1 Mapping inputs to associative cells

This basics and the importance of this step has already been explained in the discrete case. The primary difference between discrete and continuous CMAC is the way the mapping from the input vector to the associative cells is performed. This has been explained below. An index or mid cell pointer is fetched for each input as shown in Eqn(7)i.e

$$index\_pointer = floor(\beta/2) + ((35 - 2 * floor(\beta/2)) * input\_value)/100 \tag{7}$$

Once this pointer is obtained, the middle cells are assigned the weightage of 1, and the top and the bottom cells are assigned weightage as shown below:

For top cells:

Following are the index of the associative cells.

$$index = floor(pointer - \beta/2) \tag{8}$$

The corresponding weights:

$$Weightage = ceil(pointer - \beta/2) - (pointer - \beta/2) \tag{9}$$

For middle cells:

The associative cells index is made to vary in the range:

$$min = ceil(pointer - \beta/2) \tag{10}$$

$$max = floor(pointer + \beta/2 + 1) \tag{11}$$

The weights assigned to all of the above cells is 1.

For bottom cells:

The index of the associative cell:

$$index = floor(pointer + \beta/2) \tag{12}$$

The weightage assigned:

$$Weightage = (pointer + \beta/2) - floor(pointer + \beta/2) \tag{13}$$

2.2 Determination of weights

In the continuous case too the weight is continuously updated until the **no of iterations reaches 5000 or the difference between the previous and the current error reaches lesser than 0.00001. i.e converged.** This is repeated for an overlap ranging from **1 to 34**.

The weights are updated using a **learning rate of 0.05**. The total weight is calculated for each input using 5. The error is then calculated as the difference between this value and the fixed output values of the sin function. This error divided by the overlap/generalisation factor gives the correction factor for the weights. This is shown in the code where the **correction is then multiplied with the learning rate and also the weightage of each cell to get the final weight.** The current error is then calculated using 6.

2.3 Testing the CMAC

The rest of the procedure to calculate the error and the output remains the same for continuous as shown in the case of discrete.

```
Accuracy of Continuous [92.60275798034147, 96.59056926223896, 96.58010522678089, 96.68091632499109, 96.
7125530959319, 96.7424268483404, 96.81531327302662, 96.64500675855317, 96.72020897109978, 94.7844458928
638, 94.70584549244374, 94.59983210428467, 94.60528941361436, 94.32856042976066, 94.32944315898392, 93.
83933841684262, 93.84209567246575, 93.63328209987115, 93.64263108401605, 89.2291565017278, 89.221840438
43187, 89.88821371690227, 89.86122717394083, 91.75486798277329, 91.71263272819657, 90.22902145965384, 9
0.27354291121128, 85.38817881445485, 85.65333587108852, 75.55106135291992, 75.84488289425852, 81.752162
90558458, 81.73546176927172]
```

Figure 7: Accuracy readings for 30 test points : Continuous

As seen from the accuracy readings above, **the highest accuracy for continuous was obtained for a overlap factor of 7 i.e 96.81. The lowest accuracy is obtained fro a overlap factor of 31 i.e 75.5. Thus the accuracy has decreased with the overlap.**

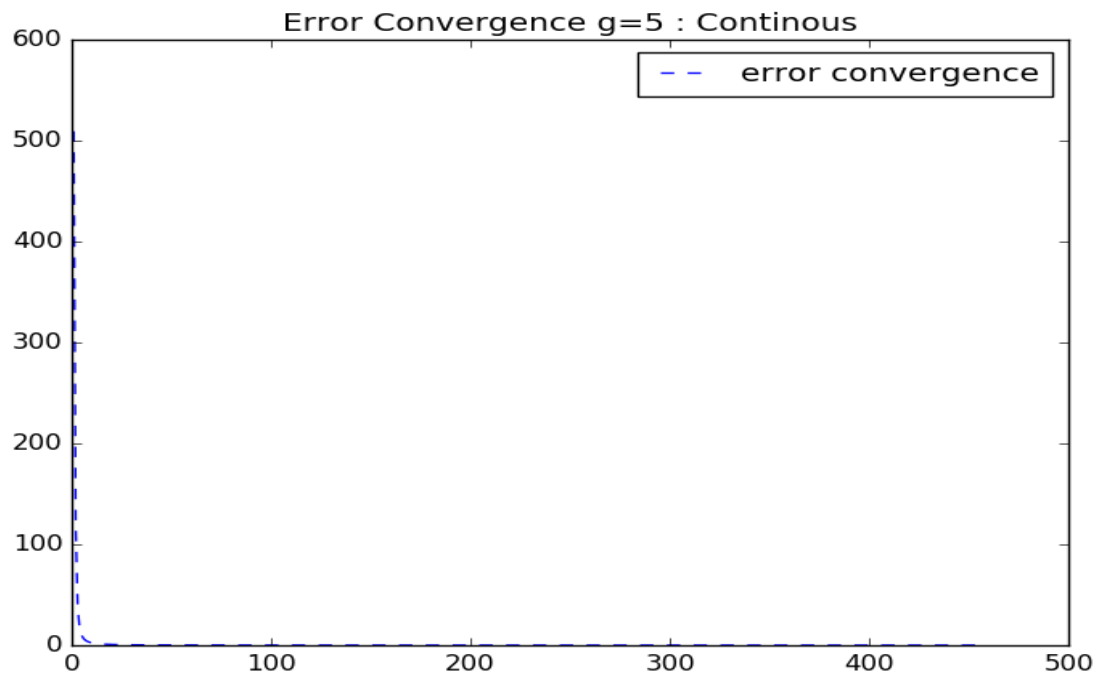

Figure 8: Error Convergence for g=5 : Continuous

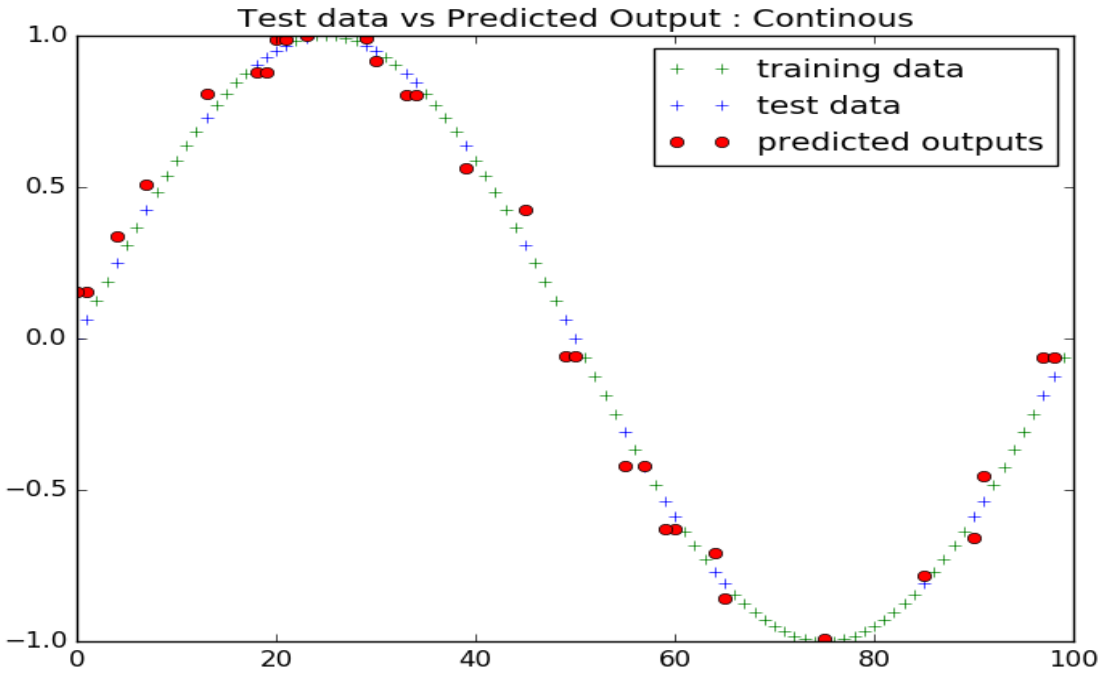

Figure 9: Test data vs Predicted Output for g=5 : Continuous

```
Time of convergence of Continuous [0.1434319019317627, 0.8778634071350098, 1.183518648147583, 1.8840570
449829102, 1.877746343612671, 2.5239009857177734, 2.4916160106658936, 3.108970880508423, 3.117816686630
249, 3.7086379528045654, 3.7531306743621826, 4.340176343917847, 4.3308563232421875, 4.991700649261475,
4.971181392669678, 4.5287253856658936, 4.728289365768433, 5.8949973583221436, 6.179940938949585, 6.2919
33298110962, 6.589343786239624, 6.1030755043029785, 6.333025217056274, 7.034682512283325, 7.20055699348
4497, 6.682766914367676, 6.919987440109253, 6.3900697231292725, 6.4944353103637695, 7.693357467651367,
7.735774993896484, 2.5351505279541016, 2.639143466949463]
```

Figure 10: Time of Convergence readings for 30 test points : Continuous

As seen from the time of convergence readings above, **the highest convergence for continuous was obtained for a overlap factor of 32 i.e 7.73. The lowest convergence is obtained fro a overlap factor of 1 i.e 0.143. Thus the convergence rate has increased with the overlap.**

3 Comparison - Discrete CMAC vs Continuous CMAC

The performance graphs of both the discrete and the continuous case has been shown below:

3.1 Accuracy

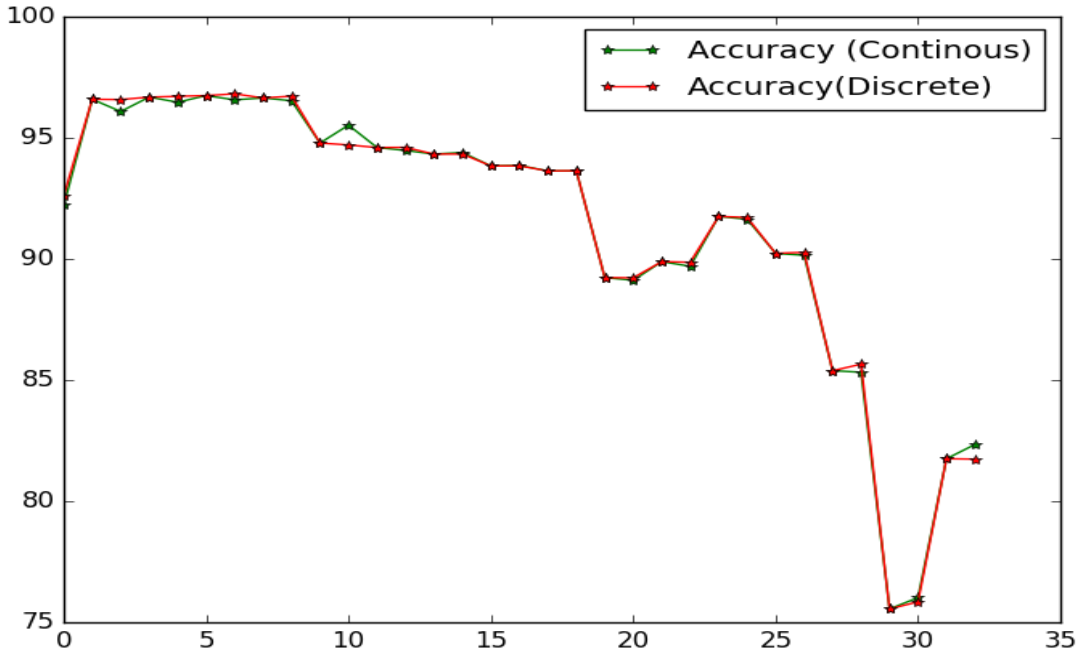

Figure 11: Comparison of Accuracy : Discrete vs Continuous

Discrete:

- As overlap area is increased, accuracy has decreased.
- This shows that as the generalization increases or the overlap increases, change in any single weight shows an effect even on the other weights.

- This is because for higher overlap change due to new training example will in-cure noise in the other weights which will reduce accuracy.

Continuous:

- In continuous, the accuracy is seen to decrease with an increase in generalization or the overlap area.
- Similar to discrete accuracy decreases due to high degree of generalization.

Comparison:

- As seen from the graph there is decrease in accuracy but accuracy of continuous is greater than accuracy of discrete.
- This is seen because when we learn with continuous function we consider even small intervals around.
- For Example, if we have a function with higher slope (higher degree) it can be better learned by continuous than by discrete.

3.2 Time of Convergence

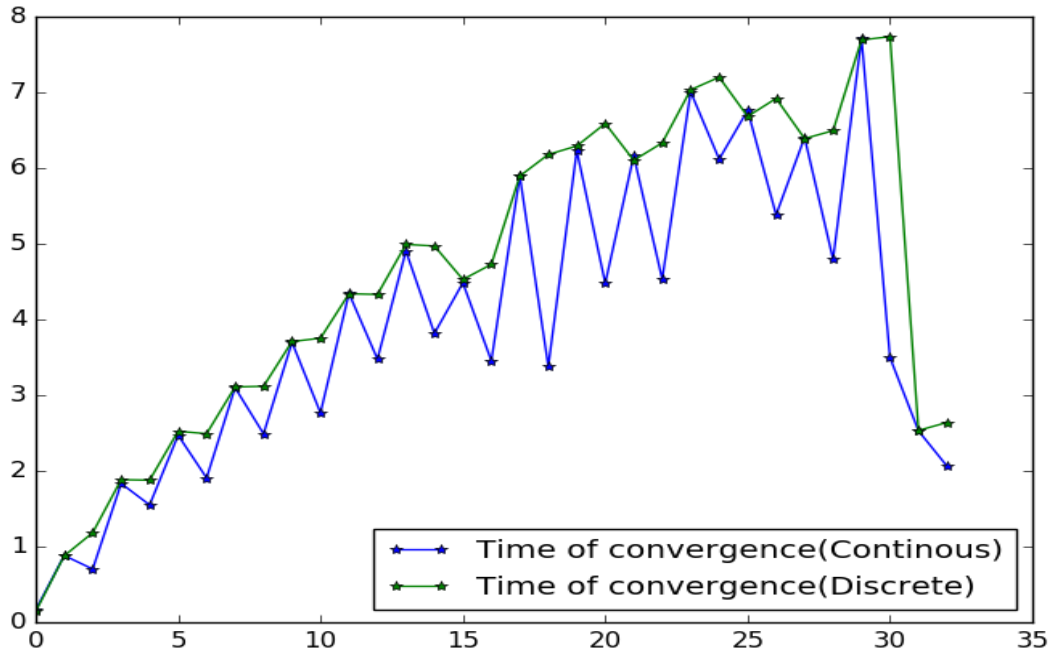

Figure 12: Comparison of Time of Convergence : Discrete vs Continuous

Discrete:

- As there is increase in overlap time of convergence increases due to many weights changing at every instance iteration.

- This incures noise in the readings and it is difficult for model to converge.

Continuous:

- In continuous , similar to discrete again we see increase in time due to same reason as above.
- Overlap incures noise in the readings and it is difficult for model to converge.

Comparison: As seen though the time of convergence for discrete and continuous is more or less the same, the time of convergence for continuous is slightly greater than that for discrete.

4 Question 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).

- Recurrent connections or RNN is basically artificial neural networks where each recurrent neuron receives 2 inputs - current and the past input. The past output is similar to having memory in humans.
- In a general feedforward neural network, the activations flow only in one direction, from the input layer to the output layer. A recurrent neural network looks very much like a feedforward neural network, except it also has connections pointing backward.
- RNN suffers from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones.
- During back propagation, recurrent neural networks suffer from the vanishing gradient problem. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. If a gradient value becomes extremely small, it doesn't contribute too much learning.
- LSTMs or Long Short Term Memory and GRUs was proposed to avoid the above problems.
- They have internal mechanisms called gates that can regulate the flow of information.
- These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. Almost all state of the art results based on recurrent neural networks are achieved with these two networks.

Following is the link to the github repository where you can find the code to the project:

`https://github.com/koyalbhartia/Robot-Learning---CMAC`