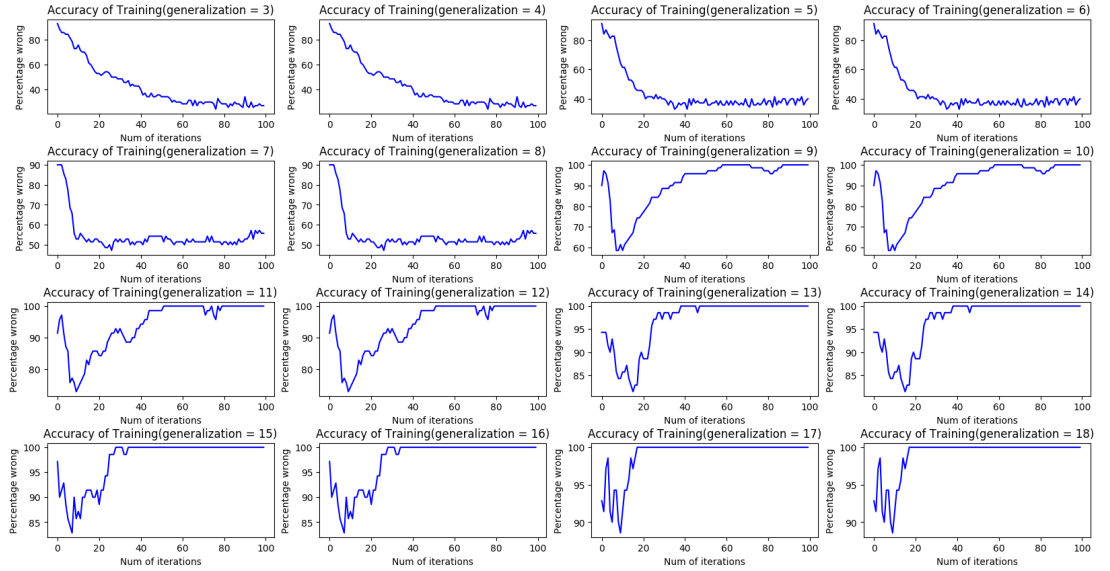


1 Discrete Cmac

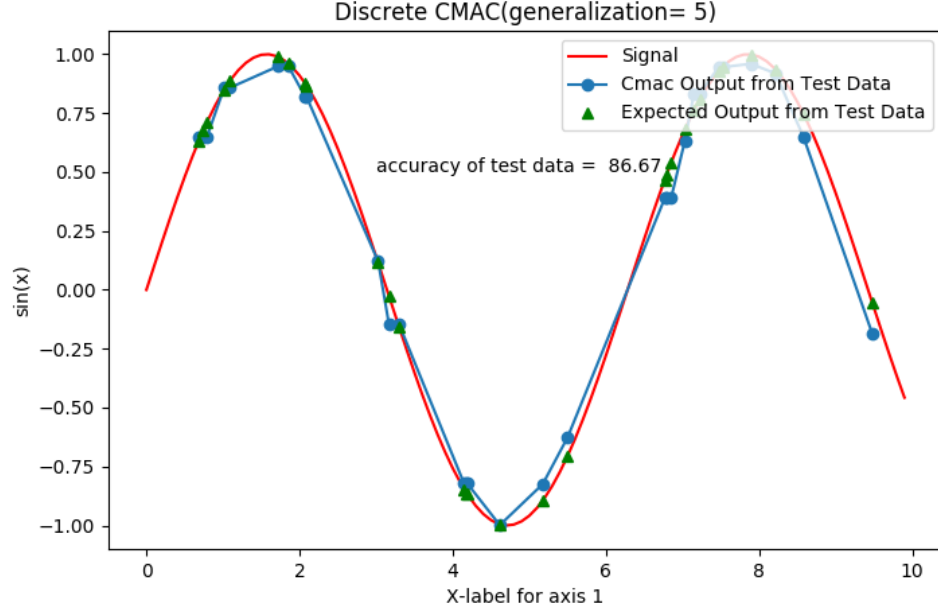
1.1 Description

1.2 Results

Generalization Vs Convergence: Clearly from the graph shown below, it can be noted that as the generalization number increases, the model has a hard time converging. This can be attributed to the fact that the generalization number, g indicates to a degree the similarity between tasks. As g increases, the model becomes weak (wrong model) as it tends to consistently learn the wrong thing by categorizing unrelated tasks as similar, hence a high bias. Ideally, g is chosen so that tasks which are related have similar values while tasks which are different have clear cut distinct values.



Accuracy: Below is a graph that depicts the accuracy of the Discrete Cmac. After running a couple of times, it can be noted that the accuracy hovers between 70-85%



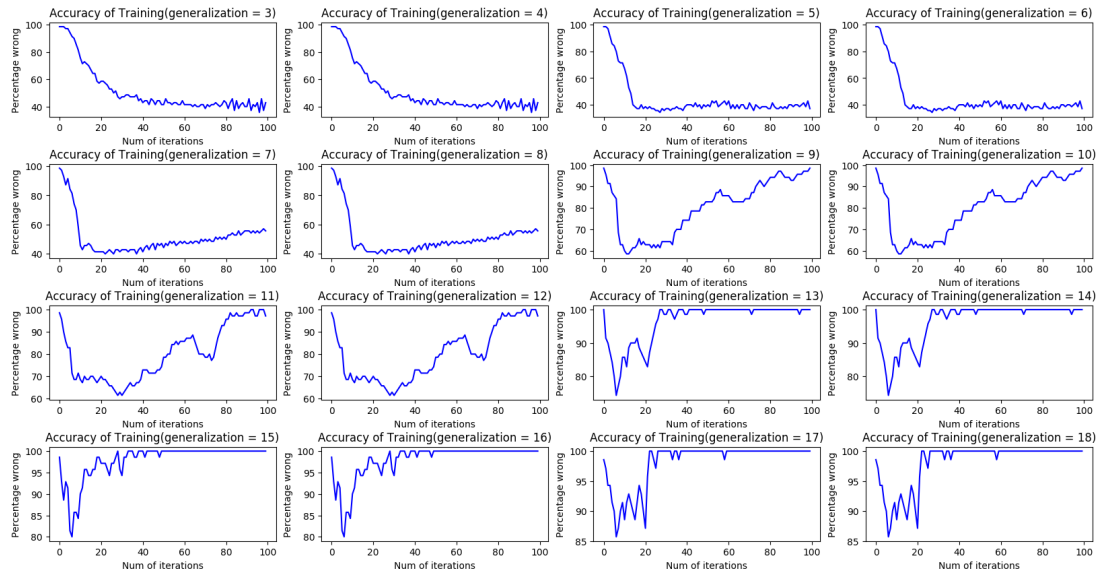
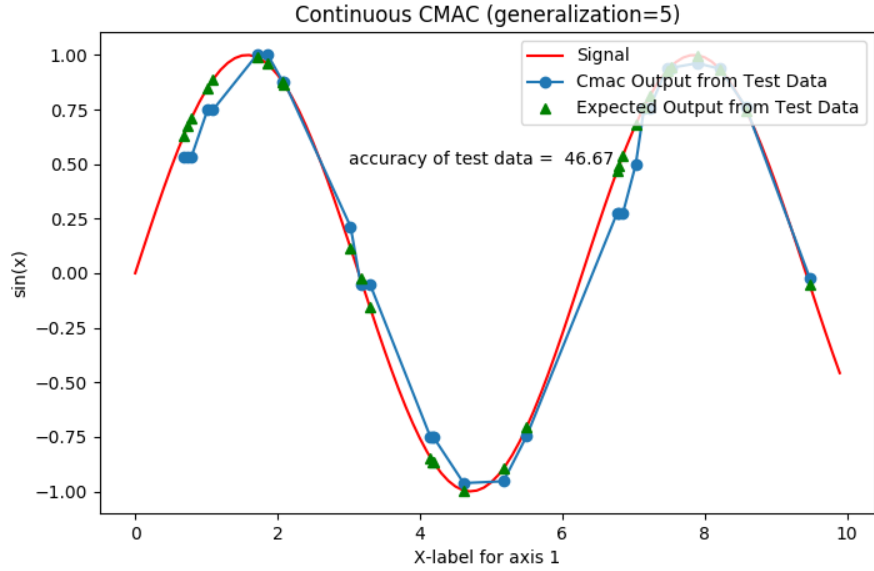
2 Continuous Cmac

2.1 Description

2.2 Results

Accuracy: Below is a graph that depicts the accuracy of the Continuous Cmac. After running the code a couple of times, it can be noted that the accuracy hovers between 40-60%

Discrete Vs Continuous Cmac: The "convergence Vs generalization" graphs for both continuous cmac and discrete cmac are relatively similar in particular for generalization numbers $g = 3$ to 6 . However, the accuracy of the test results are clearly different. For the same parameters used in training the Discrete Cmac (generalization number, number of iterations in training, accuracy used in training, learning rate), it can be noted that the continuous Cmac has a significant drop in accuracy as the model clearly seems to overfit. This is because the continuous cmac slides over more weights. Therefore, although a generalization number of 5 is used, more than 5 weights are changed during each update and this means that more inputs are categorized as similar and that is undesirable. To rectify this the generalization number can be reduced and number of iterations increased to improve performance. Also, the type of sliding window utilized has an impact on the results.



3 Recurrent Connections with Cmac

Problem: Cmac can be particular useful in modelling connections that are only state dependent. Most recurrent connections such as recurrent neural networks employed in speech recognition, music generation etc. are temporally

dependent. So if a given task is paused midway, all the relevant information is lost as the history of the information seen previously matters. Although changes can be made to such networks by storing certain information, this is usually intractable as it means a different logic to determine where the current function is located at. Therefore, a model that is only state dependent becomes necessary. Cmac can be used to model a state dependent recurrent connections and this is shown below.

Modelling Cmac to output desired trajectory: The output of cmac is now fed as an input to P (logic depends on the particular problem), then P decodes the appropriate weights to excite. This is illustrated by the diagram below.

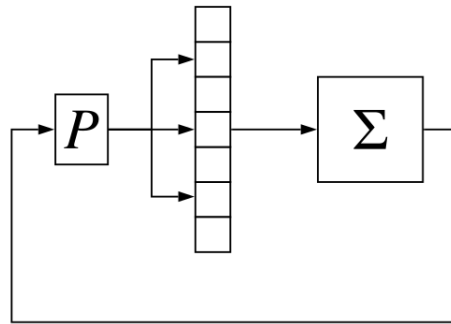


Figure 1: Recurrent Cmac

What if two different inputs have the same output value. How do we know which input to excite next?

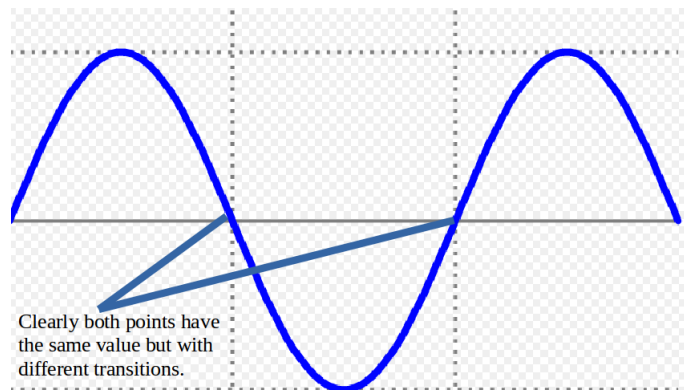


Figure 2: Trajectory

The decoding function \mathbf{P} must take this into account. Therefore, \mathbf{P} is modified to be $P(\Sigma, weights)$. Therefore, different inputs with same output weight sum are treated as different. The diagram is now modified to be the following.

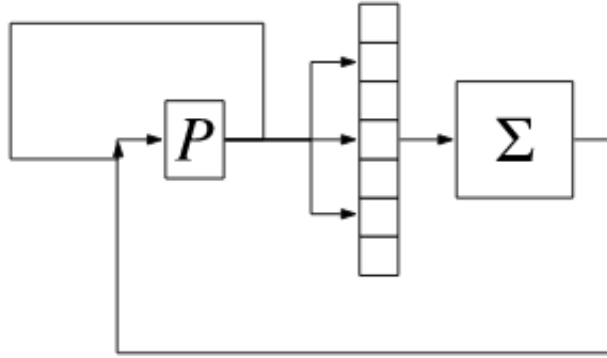


Figure 3: Modified model