Hierarchical Clustering for Efficient Memory Allocation in CMAC Neural Network

Sintiani D. Teddy[1] and Edmund M.-K. Lai[1]

School of Computer Engineering,
Nanyang Technological University, Singapore, 639798
sdt@pmail.ntu.edu.sg, asmklai@ntu.edu.sg

Abstract. CMAC Neural Network is a popular choice for control applications. One of the main problems with CMAC is that the memory needed for the network grows exponentially with each addition of input variable. In this paper, we present a new CMAC architecture with more effective allocation of the available memory space. The proposed architecture employs hierarchical clustering to perform adaptive quantization of the input space by capturing the degree of variation in the output target function to be learned. We showed through a car maneuvering control application that using this new architecture, the memory requirement can be reduced significantly compared with conventional CMAC while maintaining the desired performance quality.

1 Introduction

The Cerebellar Model Articulation Controller (CMAC) neural network was proposed by Albus [1] as an associative memory neural network that models the mechanisms of the human cerebellum. Since then, CMAC has become a popular choice for real-time control and optimization [2] such as the modeling and control of robotic manipulators [3]. It has also been applied to various signal processing and pattern-recognition applications [4,5].

CMAC learning is based on the principle that similar inputs should produce similar outputs, while inputs that are located distantly in the input space should produce nearly independent outputs. CMAC is an associative memory which stores information locally and behaves as a dynamic look-up table, in which its contents are indexed by the inputs to the network. The advantages of CMAC are simple computation, fast training, local generalization and ease of hardware implementation.

Unfortunately, the look-up table behavior of CMAC also implies that the size of the network increases exponentially as the number of input variables. This causes problems, especially when there are uneven degree of variations in the target function to be learned, where uniform quantization of input space will result in suboptimal space utilization.

It is therefore necessary to find a mechanism for efficient memory space allocation by allocating more storage space in the range of input space which holds more information. Some previously published works have tackled this problem by

W. Duch et al. (Eds.): ICANN 2005, LNCS 3697, pp. 473–478, 2005.

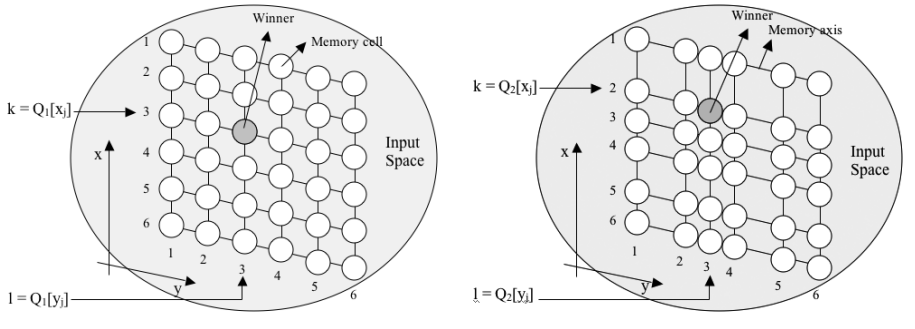
introducing non-uniform quantization of the input space to CMAC [6,7,8]. However, the examples used to illustrate the performance are single input variable cases. Moreover, there is a compromise between the computational complexity and the required memory space.

In this paper, we propose a CMAC architecture for reducing the memory requirements. It makes use of adaptive quantization based on hierarchical clustering technique, which we refer to as Hierarchical-Clustering based Adaptive Quantization Cerebellar Model Arithmetic Computer (HCAQ-CMAC). The proposed architecture is tested on an automated car maneuver control application. The experimental results show a significant improvement on the memory utilization.

2 CMAC Network

The CMAC behaves like a memory, where a particular input to output mapping acts as the address decoder. Each possible input vector selects a unique set of cells, the weighted sum of which is the output of the network for that particular input combinations.

An example of the cell memory allocation is depicted in Figure 2 for a 2-dimensional input. From this point of view, CMAC can be considered a memory in which the memory cells are uniformly distributed along the input dimensions. Each of the input dimension can be considered as being *quantized* into discrete steps or quantization levels. The input value will first be quantized into one of the levels, and the result will be the index which is used to access the memory locations. The idea of HCAQ-CMAC is based on this observation.



(a) 2D CMAC Memory Cells (b) 2D HCAQ CMAC Memory Cells

Fig. 1. Comparison of CMAC and HCAQ-CMAC Memory Surface

3 The HCAQ-CMAC

Figure 2 shows an example of a two-dimensional HCAQ-CMAC network. In HCAQ-CMAC, the memory cells are distributed in a non-uniform way according to the degree of variations of the target function to be learned. This is in contrast to Figure 2 where the cells are distributed uniformly.

The idea is to perform a non-uniform quantization of the input variables to obtain a more efficient coverage of the overall input space. The more changes observed in the region of an input variable, the more memory space will be allocated to that particular region along that axis. This results in a finer quantization level inside the input range for which "high frequency of activities" are observed. This implies that more memory cells are allocated to the range of input which holds more information than the rest of the input space.

3.1 Adaptive Quantization

The idea of adaptive quantization is to capture the input distribution and output variation. For this purpose, hierarchical clustering technique is employed. The clustering method is applied separately on each individual input dimension. For each input dimension, we start off by having each individual training data sample as a cluster. In each iteration, the two nearest clusters with the smallest merging cost function are merged to form a single cluster. The cost function is defined as the distance between the mean output value of the two clusters, expressed mathematically as

$$f(M,N) = \frac{\sum_{i \in M} X_i}{n_M} + \frac{\sum_{j \in N} X_j}{n_N} \tag{1}$$

where M and N are two different clusters, and X_i is the i^{th} output value contained in a cluster.

The clusters-merging iteration is continued until the number of clusters in that input dimension reaches the predefined memory size. This step is effectively clustering the nearest data points having similar output together, and allocating more storage cells into those densely populated areas which contain a high degree of variation in the target output.

3.2 Memory Allocation

Following the adaptive quantization, is the memory allocation, in which each of the individual cluster is allocated a memory axes along its particular dimension. The result of the memory allocation is an adaptively quantized CMAC associative neural network, as in the example depicted in Fig. 1 for 2D input case.

3.3 Network 1-Point Training and Neighborhood Retrieval

The learning equation employed is the Widrow-Hoff learning equation, modified for 1-point update and neighborhood retrieval HCAQ-CMAC:

$$\mathbf{Z}^i_{x_j,y_j} = \frac{1}{S_N}\left[\sum_{k \in K, l \in L} \mathbf{W}^i_{k,l}\right] \tag{2}$$

$$K = \{Q[x_j] - NR_x \leq k \leq Q[x_j] + NR_x\} \tag{3}$$

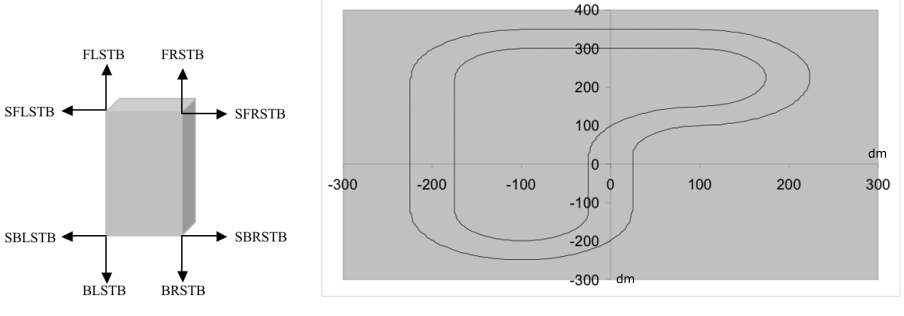
$$L = \{Q[y_j] - NR_y \leq k \leq Q[y_j] + NR_y\} \tag{4}$$

$$\mathbf{W}^{i+1}_{Q[x_j],Q[y_j]} = \mathbf{W}^{i}_{Q[x_j],Q[y_j]} + \alpha \left[\mathbf{W}^{i}_{Q[x_j],Q[y_j]} - \mathbf{D}_{x_j,y_j}\right] \tag{5}$$

Here, i is the iteration number, $\mathbf{V_j} = (x_j, y_j)$ is the two dimensional input to a 2D HCAQ-CMAC, $Q[\cdot]$ is the quantization function, $\mathbf{Z}^i_{x_j,y_j}$ is the output of the network for input $\mathbf{V}_j$, S_N is the number of elements inside the neighborhood of the current input, N is the neighborhood constant, R_x and R_y are both the input space range for input dimension x and y respectively, and $\mathbf{W}_{k,l}$ is the HCAQ-CMAC memory cell at index (k, l). Neighborhood retrieval is employed to smoothen the output of HCAQ-CMAC so that fluctuations of the retrieved output are reduced.

4 Experiments and Results

We demonstrate the performance of the proposed HCAQ-CMAC for a multi-input experiment. In particular, the HCAQ-CMAC network is used as a car a automatic steering controller. The car simulator was developed in [9] and [10]. It consists of a vehicle model together with a 3D virtual driving environment. The simulated car is equipped with 8 directional sensors in the 8 different directions of the car, as shown in Figure 4. The sensor readings were taken at every simulation time interval. The inputs to the HCAQ-CMAC network are the 4 front sensor values: FLSTB (Front Left Sensor to Barrier), FRSTB (Front Right Sensor to Barrier), SFLSTB (Side Front Left Sensor to Barrier), SFTSTB (Side Front Right Sensor to Barrier). The output of the network controls the steering angle. The car is driven along a path in a multi-lane circuit shown in Figure 2. Training data are obtained by sampling human drivers' steering control actions for the specified track. A 4-dimensional HCAQ-CMAC is trained on the steering angle response to data from the four front car sensors. Over 100 seconds of driving data are collected for training. The auto-driving performances are compared with those obtained using a standard CMAC network, employing the same learning

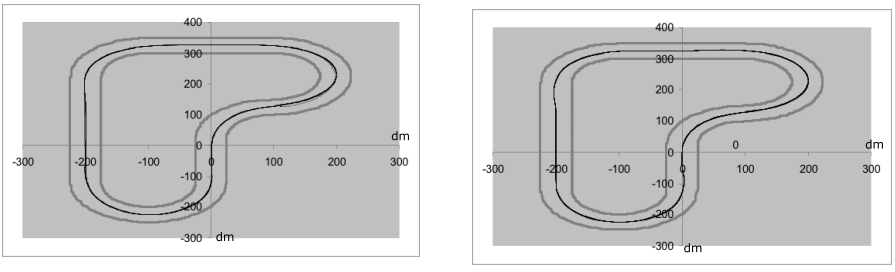


(a) Car sensors' placement (b) Driving Track

Fig. 2. Simulation Environment

Table 1. Comparison of Results from CMAC and HCAQ-CMAC Testing

	CMAC		**HCAQ-CMAC**	
Memory size per dimension	8	10	5	6
Neighborhood size	0.2	0.2	0.1	0.1
Training				
Learning constant	0.1	0.1	0.1	0.1
Final epoch training error	28.5546	30.104	22.6618	22.6607
Training time	9031 ms	14453 ms	3438 ms	4000 ms
Testing				
Average deviation from centre line	0.3499 m	0.2068 m	0.2216 m	0.2058 m
Average deviation of car orientation	0.7272 rads	0.7594 rads	0.6891 rads	0.6969 rads



(a) HCAQ CMAC Driving Path (size = 6) (b) CMAC Driving Path (size = 10)

Fig. 3. Driving paths comparison

function and parameters. The results are tabulated in Table 1. Figure 3 gives a visualization of the tack path obtained using $6 \times 6 \times 6 \times 6$ HCAQ-CMAC as compared to the path obtained using $10 \times 10 \times 10 \times 10$ CMAC. It is observed that using a HCAQ-CMAC whose size is only 60% of the original CMAC network (in each dimension), driving qualities are very similar. This significant improvement on memory allocation will not only reduce memory requirement of a CMAC network, but will also reduce the network training time.

5 Conclusions

We have presented the HCAQ-CMAC as an enhancement to the original CMAC architecture. HCAQ-CMAC improves memory utilization of CMAC by allocating more memory cells in the region where rapid changes in the output of the target function are observed. The performance has been evaluated on multiple-input application – an automated car maneuver control. Simulation results show that significant reduction in memory size can be achieved in HCAQ-CMAC, while still maintaining comparable quality of performance compared to the standard CMAC network. Further research in this direction will include a more detailed

study of the computational complexity of the proposed approach and apply it to other application areas.

References

1. Albus, J.S.: A new approach to manipulator control: The cerebellar model articulation controller (CMAC). J. Dynamic Syst., Measurement, Contr., Trans. ASME (1975) 220–227
2. Yamamoto, T., Kaneda, M.: Intelligent controller using CMACs with self-organized structure and its application for a process system. IEICE Trans. Fundamentals **E82-A** (1999) 856–860
3. Commuri, S., Jagannathan, S., Lewis, F.L.: CMAC neural network control of robot manipulators. J. Robot Syst. **14** (1997) 465–482
4. Wahab, A., Tan, E.C., Abut, H.: HCMAC amplitude spectral subtraction for noise cancellation. Intl. Conf. Neural Inform. Processing (2001)
5. Huang, K.L., Hsieh, S.C., Fu, H.C.: Cascade-CMAC neural network applications on the color scanner to printer calibration. Intl. Conf. Neural Networks **1** (1997) 10–15
6. Moody, J.: Fast-learning in multi-resolution hierarchies. In: Adv. Neural Infor. Processing Syst. Volume 14. Morgan Kauffman Publishers (1989) 29–38
7. Menozzi, A., Chow, M.: On the training of a multi-resolution CMAC neural network. 23rd. Intl. Conf. Ind. Electron. Contr. Instrum. **3** (1997) 1130–1135
8. Yeh, M.F., Lu, H.C.: On-line adaptive quantization input space in cmac neural network. IEEE Intl. Conf. Syst., Man, Cybern. **4** (2002)
9. Pasquier, M., Quek, C., Toh, M.: Fuzzylot: A self-organizing fuzzy neural rule-based pilot system for automated vehicle. Neural Networks **14** (2001) 1099–1112
10. Ang, K.K., Quek, C.: An improved mcmac with momentum neighborhood and average trapezoidal output. IEEE Transactions on Systems, Man and Cybernetics Part B **30** (2000) 491–500