Proceedings of the 6th WSEAS Int. Conf. on NEURAL NETWORKS, Lisbon, Portugal, June 16-18, 2005 (pp14-19)

An Algorithm for Saving the Memory Utilization in the 1-D Cerebellar Model Controller

Wang Chiang and Cheng-Chih Chien
Department of Electrical Engineering ,Tamkang University,Taipei,Taiwan

Abstract

It is very difficult to establish a mathematical model of a complicated higher-order nonlinear system; therefore, the neural network with the nonlinear-mapping capability (ability) is widely adopted to solve the control problem. However, it takes a very long time for the learning of conventional neural network, so the cerebellar model with the merits of simple algebraic operations and local update of weighting number (value) can replace the neural network (with the shortcomings of long-time learning). In this paper, a judging method by the function's slope is adopted to save the district value of average in the same memory when the variation of the output is not great, so the memory utilization can be saved effectively. Hence, the learning effect can be improved and the practical hardware cost can be saved.

Keywords : Cerebellar Model Articulation Controller; CMAC

1. Introduction

The frame of the cerebellar model controller is shown in Fig.1. It imitated the frame of a human cortex's storage by a series of mapping methods to reach the function of repeated learning. The operation method of learning is shown as follows:

First, a learning space that provides CMAC for obtaining the training sample must be specified. Then, the space is quantified in as many discrete pieces (S=($s_1, s_2, \ldots\ldots, s_k$)). By mapping the index memories, the physical (real) memories which are mapped can be determined. The numbers stored in the physical (real) memories are called weights (W=($w_1, w_2, \ldots\ldots, w_n$)). The responding output to the input state can be obtained by summing the contents of the mapped memories. When CMAC has not yet finished the training, the corresponding CMAC output values are somewhat different from the expected values of the samples. Hence, the error between the expected value of the sample and the CMAC real output is averagely distributed to the physical (real) memories mapped by the index memories. The containing value is then modified according to the errors, thus the CMAC output values is expected to be closer to the expected value in the next time.

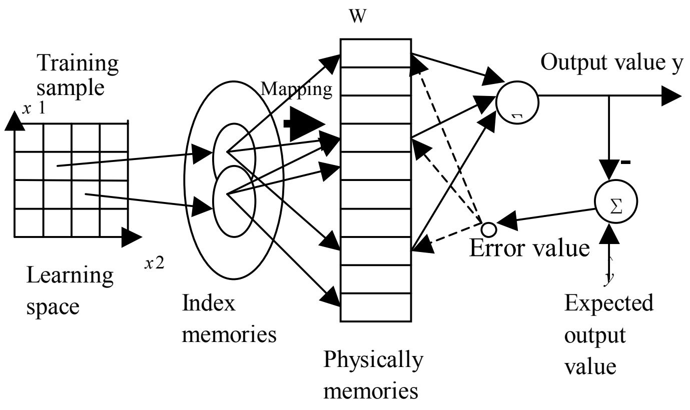

Fig. 1 The basic frame of the cerebellar model controller

2. Memory Division

In the cerebellar model controller, every variable was quantified, so the state space is separated (divided) into many discrete pieces. Arbitrary quantified input state can be mapped to a set of physical (real) memory, and the output can be obtained by this set of memory. Hence, the output signal in every state is distributed and saved in some physical (real) memories.

2.1 1-D CMAC Memory Division

The divided way of the memory units in the 1-D cerebellar model controller is shown in Fig. 2. It is the most general and easiest to be understood. The distance between the neighboring sample states is called resolution. The parameters of the memory-unit number and the resolution mapped by every sample state can be defined by us.

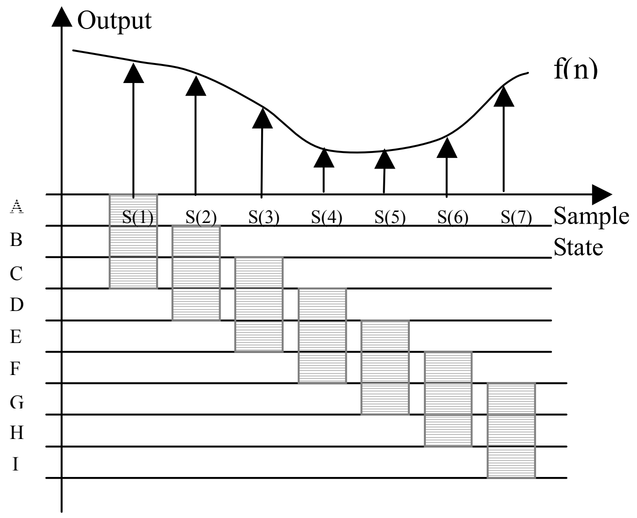

Fig. 2 The divided way of the memory units in the 1-D cerebellar model controller

According to the division method, the indices of all the sample states of the mapped physical (real) memory are set as “1”, and the sample states that have not been mapped to the memories are set as 0 (shown in Table 1)

State \ The Mapped Memories	A	B	C	D	E	F	G	H	
S(1)	1	1	1	0	0	0	0	0	0
S(2)	0	1	1	1	0	0	0	0	0
S(3)	0	0	1	1	1	0	0	0	0
S(4)	0	0	0	1	1	1	0	0	0
S(5)	0	0	0	0	1	1	1	0	0
S(6)	0	0	0	0	0	1	1	1	0
S(7)	0	0	0	0	0	0	1	1	1

Table 1 The memory-unit indices table

From the foregoing table, if the input states were quantified as k states (k=7), which are represented as S(1), S(2),…, S(7). Every state use m weights (m=3), then there will be n memory units (n=k+m-1). Equation (1) is used to represent the stored data in S(k)

$$y_{S(k)} = C_{S(k)}W = \sum_{j}^{n} C_{S(k,j)}W_j \ldots\ldots\ldots\ldots\ldots(1)$$

2.2 2-D CMAC Memory Division

A general memory-unit dividing way of the 2-D cerebellar model controller is shown in Fig. 2. In the 2-D learning space, every input variable axis is quantified as nine discontinuous units, which are called elements. The width of every element is called resolution. The small squares surrounded by the small discontinuous units of these two input-variable axes are called input states. Fig. 3 shows that the learning space can be quantified as 81 discontinuous input states.

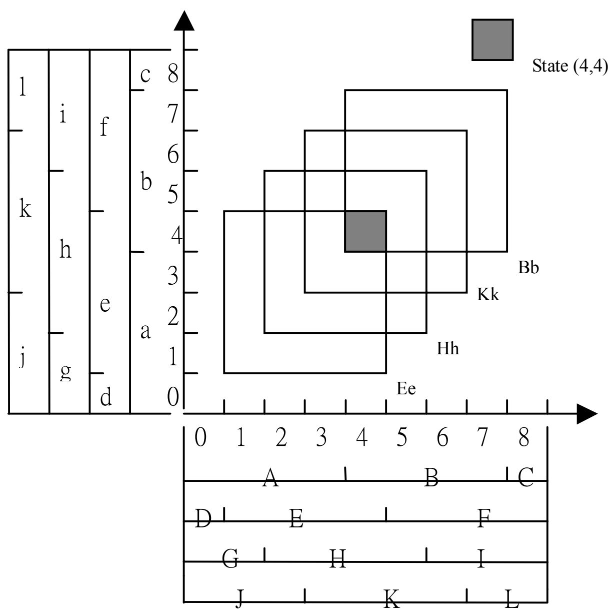

Fig. 3 The dividing way of the 2-D cerebellar model's memory units

After every quantified layer has been established, only the divisions on the same layers can form cubes according to the general rule. There are 4 quantified layers, 9 different size super-cubes in each layer for a total of 36 super-cubes. All of the super-cubes according to the definition in Fig. 3 are shown in Table 2.

Layer Name	Names of the Super-Cubes								
LayerQ1	Aa	Ab	Ac	Ba	Bb	Bc	Ca	Cb	Cc
LayerQ2	Dd	De	Df	Ed	Ee	Ef	Fd	Fe	Ff
LayerQ3	Gg	Gh	Gi	Hg	Hh	Hi	Ig	Ih	Ii
LayerQ4	Jj	Jk	Jl	Kj	Kk	Kl	Lj	Lk	

Table 2 The cubes' names generated by the quantification layers

3. Saving Memory Size Algorism

After the training samples have been inputed to CMAC, they will be addressed to several physical memory addresses for storing the sample information by a series of quantification and mapping. Then, the information can be retrieved from these physical-memories positions by using equation (1), and the CMAC output can be obtained by summing these data. This algorithm can make the method of conventional memories-utilization settings flexible, hence the demands of function accuracy and the needs of the memories can be totally evaluated according to our needs.

3.1 1-D Saving Memory Size Algorithm

When the learning target is a 1-D function, the first thing is to obtain the parameters of variation degree between the neighboring target samples (shown in equation (2)) in order to have a basis for saving the memories positions.

$$m_n = \left| \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \right| \quad ....(2)$$

(n=1,2,……,k)

The parameters for comparison is first found out, and compared with the variation-degree parameters. Then, determine which target samples can be left out. The formula is shown in equation (2), where α is selected by ourselves. If we want to decrease the memory utilization, we must choose a smaller α.

$$\rho = \left(\frac{f(x_a)_{\max} - f(x_b)_{\min}}{x_a - x_b} \right) \div \alpha \quad ......(3)$$

Compare the magnitude relation of m and ρ. If $m_n \geq \rho$, then map to a set memory and continue to compare m_{n+1} and ρ. When m_n is less than ρ, then take the average value of the (n-1)th and nth target sample. The averaged value will be the new common target value of the (n-1)th and nth sample states. During the mapping process, only a memory position is mapped, and the comparison process of m_{n+1} and ρ was left out, and perform directly the comparison of m_{n+2} and ρ.

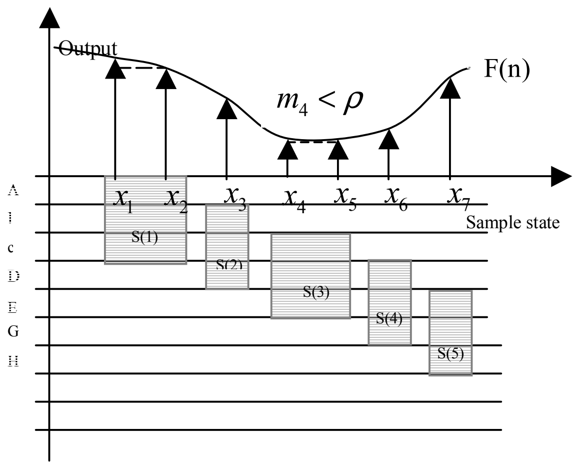

Fig. 4 The divided way of leaving out the memory units for a 1-D cerebellar model controller.

Compare Fig. 2 and Fig. 4, it is found that the precursor action can increase apparently the quantification state and memory utilization.

4. 1-D CMAC Simulation Example

The foregoing algorithm is verified by an actual example

1. Learning Function: y=tanh(0.5*x)
2. Learning Space: -10~10
3. The states' number of the quantified learning space: 101
4. The memory-unit number mapped by the input state
5. Learning rate: 0.5
6. Learning performance index: 0.5
7. The number of the training samples: 101
8. Performance evaluation: adopt the sum of square of error.
9. The value of α :4 and 8

The simulated results are shown as follows

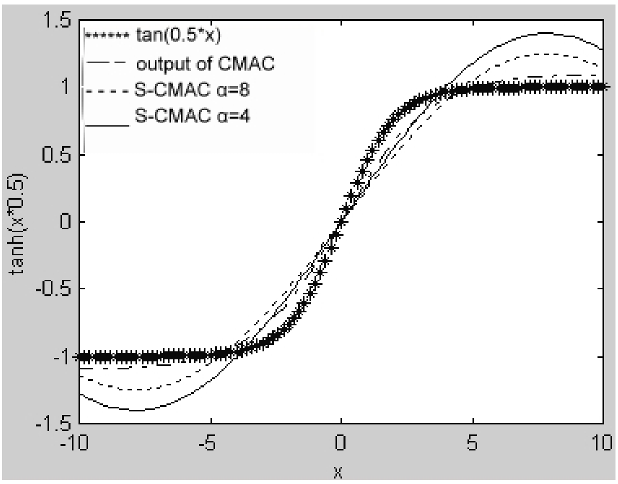

Fig. 5 The first learning period

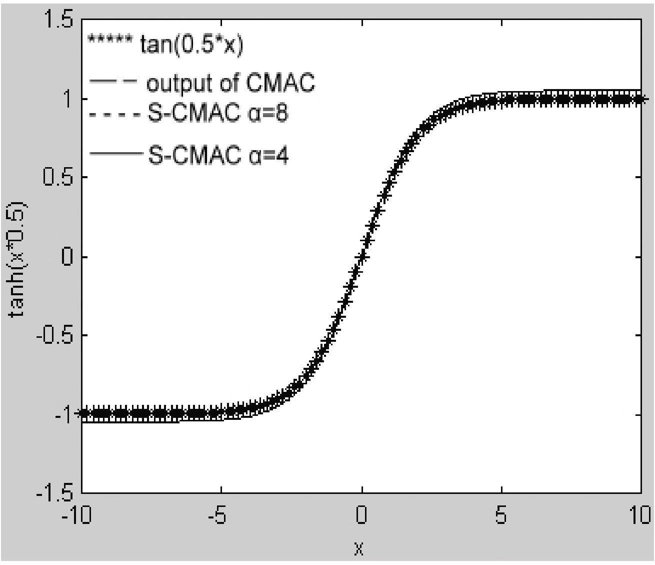

Fig. 6 The twentieth learning period

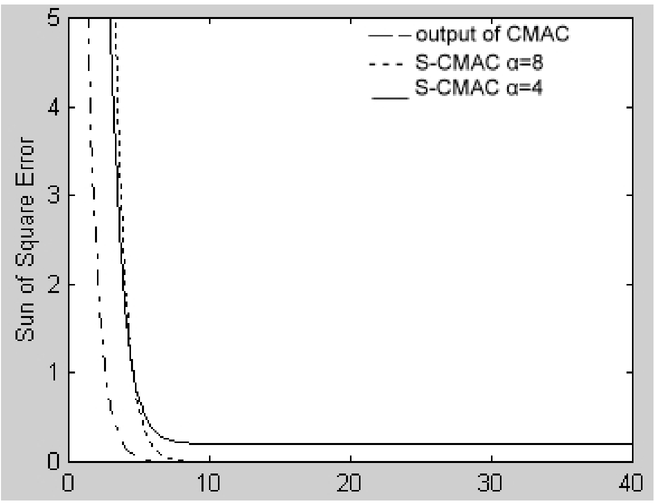

Fig. 7 The comparison of the learning errors

From this example (see Fig. 5), the learning effect to the target function of S-CMAC (saving memory's CMAC) is less than the conventional CMAC during the initial stage of the learning period. However, when the learning period increases, (see Fig. 6 and Fig. 7, S-CNAC, α=8), the output value is very close to the output of the target function. Hence, it takes a longer learning period for S-CMAC to reach the ideal output effect.

Although it takes a shorter learning period for a conventional CMAC frame to reach the convergent effect, it needs a longer learning period to converge. However, it doesn't mean that the learning effect of S-CMAC is worse. Since in the precursor, the number of the samples to be learned has been decreased, the calculation quantity in every period decreases, the needed time for every period decreases. Although the period for convergence is longer, the total time has not great difference with the initial CMAC frame.

Furthermore, the choice of α value is related to the output error and the saving quantity of the memories. The smaller the α value is, the smaller the memory utilization quantity, hence the error becomes greater, it even can't be converged to the acceptable expected output value. We should choose the suitable α value to meet our needs, and obtain an equilibrium point between the hardware cost and output quality

Conclusions

The main essence of the learning algorithm during the CMAC training process is the average distribution. That is to say, after the training samples have been inputed to CMAC, they will be addressed to several physical memory addresses for storing the sample information by a series of quantification and mapping. Then, the information can be retrieved from these physical-memories positions by using equation (1), and the CMAC output can be obtained by summing these data.

Every state of the conventional 1-D CMAC needs so multifarious calculation quantities. If the states' number to be divided is too large, then the calculation quantities will be very large. The improved algorithm in this paper will help us to efficiently decrease the calculation quantities and memory utilization under the presupposition that the error value is not too great and choose carefully the value of α. Compared with the conventional CMAC output, it needs more learning period to reach the convergent effect. If the CMAC output of the controlled body is not requested to be very accurate, but is requested to converge quickly, using the algorithm will save the hardware cost and learning time.

References

[1] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers,"ASME *Journal of Dynamic System,Meaurement,and Control*,vol.64,pp.759-768,November 1942.

[2] J.S.Albus,"A new approach to manipulator control : The cerebellar model articulation controller(CMAC),"*ASME Journal of Dynamic System,Meaurement, and Control*,vol.97,pp.220-227,September 1975.

[3] C. S. Lin and H. Kim,"Selection of learning parameters for CMAC-Based adaptive critic learning,"*IEEE Transactions on Neural Networks*,vol.6,no.3,pp.642-647,May 1995.

[4] J. S. Albert, "Data storage in the cerebellar model articulation controller (CMAC) *ASME Journal of Dynamic Systems, Meaurement, and Control*, vol.97,pp.228-233,September 1975.

[5] A. Thammano and C. H. Dagli, "A comparison of *FAM and CMAC for nonlinear control, "in IEEE World Congress on Computational Intelligence, Proceedings of*

the third IEEE Conference, 1994, vol.3, pp.1549-1553.

[6] W.T.Miller, R. P. Hewes, F. H. Glanz, and L. G. Kraft, “Real-time dynamic control of an industrial manipulator using a neural-network-based learning control,“ *IEEE Transactions on Robots and Automation*, vol. 6,no.1, pp. 1-9,February 1990.