ON LEARNING OF CMAC FOR MANIPULATOR CONTROL

Heon Hwang and Dong Y. Choi

Robotics Lab., Automation Eng. Dept.
Korea Institute of Machinery and Metals
Sangnam-dong 66, Changwon, Kyungnam, Korea

ABSTRACT

Cerebellar Model Arithmetic Controller (CMAC) has been introduced as an adaptive control function generator. CMAC computes control functions referring to a distributed memory table storing functional values rather than by solving equatioins analytically or numerically. CMAC has a unique mapping structure as a coarse coding and supervisory delta-rule learning property.

In this paper, learning aspects and a convergence of the CMAC were investigated. The efficient training algorithms were developed to overcome the limitations caused by the conventional maximum error correction training and to eliminate the accumulated learning error caused by a sequential node training.

A nonlinear function generator and a motion generator for a two d.o.f. manipulator were simulated. The efficiency of the various learning algorithms was demonstrated through the cpu time used and the convergence of the rms and maximum errors accumulated during a learning process. A generalization property and a learning effect due to the various gains were simulated. A uniform quantizing method was applied to cope with various ranges of input variables efficiently.

1. INTRODUCTION

On close examination of the simple or complex manipulating behavior such as those performed by biological organisms, the computational methods involved in engineering manipulator control problems have serious shortcomings. They fail to produce a truly sophisticated and adaptive motor behavior. The degree of diffculty experienced in obtaining mathematical solutions even for trivial actions performed by ordinary organisms is very high. However, it is almost certain that the biological organisms do not solve or model the complex mathematical formulation for such complicated motor behavior. Instead, it seems that biological organisms use some form of memory driven control system. Hence, many researchers have investigated the structural and functional properties of the brain.

Because of the general drawback to adaptive controllers for complicated robotic systems which require the computation of real time parameter identification based on some performance criteria and management of sensitivity on sensory inputs, the robust adaptive controller based on the biological structure and function have drawn a great attention recently. In the hope of achieving human-like highly adaptive and sophisticated manipulating behavior with perception of image and speech, a lot of researchers are involved in functional and structural modeling of information processing of the human brain. Artificial neural net models have been studied by scientists in various fields for many years. How to achieve a great degree of the robustness, adaptation, and easy learning is the major focus on this area which also requires high computation rates.

Anatomical and neurophisical studies of the cerebellum have led to a theory concerning the functional operations of the cerebellum. Some basic principles of how the cerebellum accomplishes motor behavior have been organized into a mathematical model, Cerebellar Model Articulation or Arithmatic Controller(CMAC) by Albus[1,2,3]. The CMAC is a schematical approximate modeling of the information processing characteristics of the cerebellum. Through a series of storages or learnings the CMAC works as a computational module generating weights in a distributed table look-up manner connected in parallel.

The CMAC has been applied to several control applications and revealed its usefulness at various levels in control problems. Albus[1,5] implemented CMAC module to control a seven degree of freedom master-slave arm. The CMAC was used in a closed loop control system to make the slave arm follow a specific trajectory. The trajectory was trained by the feedback from the movement of the master arm.

The CMAC module was used to control a two degree of freedom biped walking device by Camana[6].

Three degree of freedom planar manipulator was trained to follow a trajectory to avoid collisions under a fixed static obstacle by Rajadhyaksha[7]. Reference trajectories were generated manully using a stylus and trained iteratively.

An adaptive hierarchical model for two dimensional computer vision was simulated using the CMAC by Manglevdakar[8]. Three hierarchical levels based on the segmentation were implemented to analyze the image formed from a digitizer tablet.

Miller[9] applied the CMAC to position the manipulator with visual feedback by training on-line observations of the input-output relationship of the system being controlled. Learning parameters were chosen on an ad hoc basis. The CMAC based control of a two degree of freedom articulated robot arm was performed for the simulated repetitive and non-repetitive movements by Miller et.al[10]. The control activity of the CMAC in conjunction with a fixed-gain linear feedback controller was tested. A simulated learning of a manipulator was performed every one cycle of trajectory trace using on-line information based on the Albus's maximum error correction training[3].

Although the CMAC was applied to various applications as a control substitute or a reference input generator, a detailed analysis of the CMAC mapping and network with a learning capability was not performed yet. The convergence of the CMAC was anticipated from the experimental simulation.

The CMAC has a simple structured processing nature of generating output in response to any continuous or discrete state input. It requires, however, a design guide to specify control parameters and an efficient learning algorithm as a controller for unmodeled or modeled systems. To provide a design guide a through investigation on the convergence trend of learning and related control parameters should be done because of their nonlinear effects on trained results.

In this paper, we present the extended view on the CMAC such as mapping structure, learning process, memory requirements, and convergence and continuity (generalization) property, which has not been proved formally yet[3,10].

Three types of basic learning rules such as a batch type accumulated sequential error learning, on-line type direct sequential error learning, and a learning based on the uniformly distributed random errors were investigated in comparison with the conventional maximum error learning. A uniform quantizing method was applied to cope with the various ranges of input variables. Simulation results of a nonlinear function generator and a delta joint motion generator for a two d.o.f. manipulator were analyzed under the various training rules.

2. EXTENDED VIEW ON THE CMAC

This section briefly explains about the structure and function of the computational CMAC module, and presents the extended view on the CMAC. The CMAC accepts a continuous or discrete state input vector by converting a precise input into many discrete fuzzy inputs and produces an output vector by summing the distributed responses, called weights. From a functional point of view, learning can be stated as a sequential storage of the difference between the desired and the CMAC generated approximated discrete system responses in a distribute manner by forcing those differences to be near zero. In fact, the CMAC can be applied to any system which has input-output relations such as $P = H(S)$.

The CMAC maps the continuous or discrete input states into the structured discrete pattern vectors determined from the resolution and the number of quantizing blocks of each input variable and generates the approximated discrete responses. The hyperstructure of the combined quantizing block and the resolution of each input variable affects the memory size required and reveals two opposite features, the generalization and the interference.

In the case of non-fixed (real) node input state vectors, the CMAC can not maintain the unique input and output relation. However, the CMAC maps or discretizes successfully an infinite number of input states. An iterative learning scheme of the CMAC can be thought as a powerful substitute of LMS(Least Mean Square) error procedure.

In a word, main characteristics of the CMAC is placed on its structured mapping which decomposes continuous or discrete vauled input state vectors into a set of linear independent binary valued input pattern vectors. Other processes beyond this conversion is the same as those of a linear associator model of the neural net except a slightly different learning scheme. With a little modification by introducing non-linear activation function, the CMAC can be applied to a case of the continuous valued inputs and the discrete binary output pattern vectors.

The CMAC is composed of two main mappings in addressing the corresponding response memories for a given input.

$$S \rightarrow A \text{ and } A \rightarrow P,$$

where S = sensory or command input state vector
A = association or address vector
P = response vector

The first mapping determines the active address vector for storing and retrieving trained results from any given input and the second generates the corresponding response, which is the arithmetic sum of the values stored in the active address vector. $S \rightarrow A$ is broken down into two sub-mappings such as $S \rightarrow M$ and $M \rightarrow A$, where M is an intermediate vector. Each input vector S composed of N variables which can be continuous or discrete.

A range of each S_i is seperated by K_i, the quantizing block, resulting in quantizing functions, ${}^iQ_1, {}^iQ_2, \cdots, {}^iQ_{Ki}$, where i and K represent the input variable i and the Kth quantized layer of variable i. The quantized block is one bounded to assure the equivalent quantization. Since the quantized blocks of each layer are indexed and offset by one unit between the adjacent layers, the number of quantizing layers of variable i is equivalent to K_i. The interval of S_i is usually converted to one unit. Since the resolution of S_i stated by Albus may lead to the restriction of the input state vectors, the interval of S_i was used instead. In fact, the resolution of the input state vector is determined from the offset of quantized blocks. Input state variables are usually continuous but sometimes discrete with the resolution determined from the characteristics of the system components or discrete sampling time.

A set M_i is composed of ${}^iQ_1, {}^iQ_2, \cdots, {}^iQ_{Ki}$. Each ${}^iQ_j$ has $INT(S_{Ri}/K_i)+1$ or $INT(S_{Ri}/K_i)+2$

number of the quantized blocks, where S_{Ri} is a range of input variable i. For given S_i a set of elements M^*_i is composed of the quantized blocks selected from each ${}^iQ_j$, where $j = 1, \cdots, K_i$. For variable i, the total number of quantized blocks, ${}^iN_T$ and the number of quantized blocks in layer j, ${}^iN_j$, can be obtained such that

$$ {}^iN_T = S_{Ri} + K_i $$
$$ {}^iN_j = \mathrm{INT}(S_{Ri} / K_i) + 1 \text{ for } j=1,\cdots,K_i $$

If ${}^iN_T$ is greater than $\sum_{j=1}^{K} {}^iN_j$, one is added from the first layer to the next until they are equal. $|A|$, the number of elements in association vector A, concatenated from N input variables can be obtained such that

$$ |A| = \sum_{j=1}^{K} {}^1Q_j \cdot {}^2Q_j \cdots {}^KQ_j, $$

where $|A|$ denotes the required number of the CMAC system memory. The CMAC reduces the infinite memory required for the continuous control function to $|A|$ number of memories by discretizing the control function while maintaining a certain accuracy. A set of concatenated block of M^*_i for a corresponding layer of every input variable produces the association vector, A^*(address decoder), which denotes a distributed memory address where trained results are stored or retrieved.

As the result of the structured mapping of $S \rightarrow A$, the data storage at any point alters the values stored at neighboring points. The size and shape of a neighborhood obviously depend on a quantizing interval K_i and the offset of input variables. The hypercube generated by the input variable offsets should be scaled to equal to the precision required for each variable. Thus a unit step along any input variable axis will always be within the desired precision. Although this required precision should be set by the designer considering the resolution of the control system components, the affordable system memory and desired accuracy of system response should be considered as mentioned earlier.

Since for most CMAC control applications it is hard to determine the sensitivity of the resolution of each input variable to a system response because of the combined effect with K_i, input variable ranges, and the characteristics of the control function, a uniform scheme for the CMAC quantization has been devised as following.

Based on the estimated precision for each input the range of each input variable is scaled and modified by extending or contracting its range to achieve a unit step interval. In the CMAC, every input variable is actually considered to be nondimensional and the only factors to be considered for a structured mapping are the mapped CMAC input ranges, variable offset, and quantizing intervals. After selecting the maximum range of all input variables, other input variable ranges are set to the number of unit steps of the selected maximum range. This procedure allows various offsets of input variables and the uniform mapping through the use of a common quantizing interval such that $K = K_i$. this scheme eliminates the restrictions of the magnitude of a quantizing interval caused by relatively small input range allowing the efficient mapping for the data storage and improves the CMAC system response because of the offset adaptive to input variable range. However, it should be noted that the scaled offset should be above the resolution of the control system component.

For a given fixed node input, the conventional maximum error correction(MEC) training proposed by Albus does the following data storage process. Initially all sets of association vector are set to zero. At every discrete input node a difference bettween the desired function value and the CMAC generated approximated value is computed. Comparing it with the predefined tolerance and choosing an integer node at which a maximum error occurs, the CMAC mapping of the input is performed to generate A^*,which is the active set of association vector. The error is stored at each element of A^* in a distributed manner such that

$$ \Delta = G \left[\frac{F(S_m) - CMAC_m}{|A^*|} \right] $$

where G	=	training gain
S_m	=	input node vector where the maximum error occurs
$F(S_m)$	=	CMAC generated function value at S_m
$\lvert A^* \rvert$	=	number of elements in A^* which is equivalent to K

The value G is less than or equal to one and represents the learning speed, denoting G=1 is equivalent to one step learning. As the value of G is smaller, the learning gets slower and the resulting weight difference has lesser effect on the neighborhoods defined by K.

Although the MEC training has the inherent disadvantages of long cpu time, oscillating behavior, and the inadequacy of handling continuous or variable discrete input state vectors, it has the greatest learning performance per each trial of training. With the fixed input node space, learning on sampled node input generates a linear interpolating effect on other untrained nodes. In section 4, we present a non-oscillating learning algorithm guaranteed to converge, a fast converging algorithm with a little oscillating feature, and an algorithm for non-fixed input nodes to overcome the limitations of the MEC learning.

The CMAC has an inherent training difficulty for functions having discontinuities. A function having a discontinuity should be handled by using the separete CMACs or by taking trained results apart in the CMAC input space. A function for the CMAC to be trained should be smooth enough to get a good performance. More careful attention on the control parameters should be made for functions varying sharply because of the inherent correlative effects of the two opposite properties of the CMAC, generalization and interference. A property of generalization or continuity is simply explained as similar inputs produce similar responses because of the overlapping nature of the CMAC structured mapping. Learning interference occurs when the property of generalization is not desirable such that quite different responses are required for similar inputs. Although Albus stated the learning interference can be overcome by the repeated iterations of data storages on similar inputs, this can not be achieved successfully because of the linear characteristics of learning procedure because it uses a unity linear activation function.

3. CONVERGENCE, GENERALIZATION AND LIMITATION

Consider a coarse coding, which divides the space into large overlapping uniform sizes of zones and assigns a unit to each zone. The key fact in a coarse coding is how accurately an input feature is encoded. The CMAC structured mapping is a kind of coarse coding but has a unique coding characteristics. As mentioned earlier, the CMAC mapping scheme decomposes input state vectors into a set of linear independent binary pattern vectors whose values of elements are one for the active elements and zero for the inactive elements.

The number of generated binary pattern vectors for the input space state is same as the number of the CMAC fixed node inputs determined by the offset of input variables and is regardless of K. K defines the number of active units and the total number of binary input elements m, used for mapping. Given input space with an offset specified, the total number of decoded binary elements is decreased as K increases. The feasible number of different input pattern vectors is the number of combination K from m. From these n number of linear independent binary pattern vectors are formed by the CMAC mapping. Every binary element is connected to its own weight and the value of an output unit is determined from the linear sum of each weighed input binary values.

Once binary pattern vectors are formed, the network of the CMAC resembles exactly a linear associator with a delta learning rule as shown in fig. 1. Since the mapped input binary pattern vectors are linear independent each other, the CMAC satisfies the requirements of the delta rule learning. The CMAC weight connection can be expressed in a matrix form as a system of linear equations such that

$$[\, X_{p1}\ X_{p2}\ \ldots .\ X_{pm}\,]\ [W_1\ W_2\ \ldots ..W_m\,]^T = T_p$$

where p indicates the various input patterns and is indexed as 1,2,...,n. The weight matrix can be obtained as $\mathbb{W} = \mathbb{X}^* \ \mathbb{T}$ and $\mathbb{X}^*$ is a pseudo-inverse. As far as the system input and output has one to one corresponding relation, $\mathbb{W}$ is unique. If not, $\mathbb{W}$ is a least square solution to the system of minimum norm. An easy way to compute $\mathbb{W}$ specially for a large system is to use an iterative error correcting storage procedure called LMS(Least Mean Square) learning procedure.

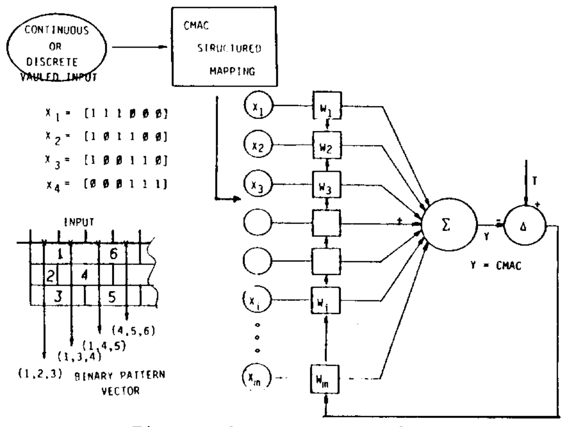


Fig. 1 The CMAC network

Since every decomposed binary input pattern vector is not orthogonal, with a given network and a given set of associations, it is desired to produce a set of weights that minimizes some sensible measure of errors. The error surface is formed from the error measure as a height in weight space whose dimension is composed of each weight in a network. The shape of the error surface is critical in the speed of learning. For a network with linear output elements, the error surface forms a bowl shape. Since the bowl has only one global minimum, a steepest descent on the error surface is guaranteed to find it. If the derivative of the error surface is proportional to the weight change by the delta learning rule, this corresponds to performing steepest descent on the error surface.

The convergence of the CMAC can be proven similary as a case of multi-layer LMS learning network[11]. The CMAC can be thought as one layer network being connected the input and output directly. Since with a linear activation function muti-layer LMS network can be converted to one layer network, the number of processing layers is not really matter while an activation function is kept linear[12]. If there is a fixed finite set of input-output cases, the error measure for a specific input-output case is

$$E_p = 1/2\ (T_p - Y_p)^2$$

p = index over input-output pairs
T_p= desired output
Y_p= CMAC generated output

The overall error is then,

$$E = \sum_{p=1}^{n} E_p$$

For a specific case,

$$y_i = x_i \text{ for } i=1,2,\ldots,m$$

$$X = \sum_{i=1}^{m} W_i\ y_i$$

Since the CMAC has a linear output which is identical to the total input,

$$Y = X$$

The partial derivative of E_p with respect to each weight is obtained from the chain rule,

$$\frac{\partial E_p}{\partial W_i} = \frac{\partial E_p}{\partial Y}\frac{dY}{dX}\frac{\partial X}{\partial W_i}$$
$$= -(T-Y)\ y_i$$
$$= -\ \delta\ y_i$$

From the requirement of steepest descent,

$$\Delta\ W_i \ \propto\ -\frac{\partial E_p}{\partial W_i}$$
$$= \zeta\ \delta\ y_i \qquad : \text{Delta Rule}$$

After one complete sweep of all pattern presentations,

$$\frac{\partial E}{\partial W_i} = -\sum_{p=1}^{n} \delta_p\ y_{pi}$$

This is strictly true for a batch type sequential error correction(SEC) learning such that the values of the weights are not changed during the epoch of whole pattern presentations. All input states are presented sequentially and an error for each pattern is multiplied by learning gain and accumulated. The accumulated errors are fed back to each weight at every epoch. It is guaranteed to move in the direction of the steepest descent. The learning gain should be small enough for a system to converge because of the accumulated effects.

For on-line type SEC learning by changing the weights after each pattern is presented, the process is apart to some extent from a true gradient descent in E. This may sometimes force the oscillating of E to occur but by making learning gain sufficiently small, the steepest descent is approximated arbitrarily closely. With a relatively high gain, although a little oscillating of E occurs during the whole sweep, the fast convergence of E per epoch is obtained. The MEC learning can be thought as a modified version of the on-line type SEC learning.

This type of the simple LMS learning procedure has its limitation to the cases of similar inputs with different outputs. Because of the generalization property of the CMAC, the interference is rather occurred when the discontinuity or serious functional change occurrs in the output values within the neighborhood of generalization. Also steepest descent will be slow at points in the weight space where the error surface forms a long ravine with steep sides and a very low gradient along the ravine. In this case, the gradient at most points in the space is almost perpendicular to the direction towards the minimum. If the learning gain is large, there are divergent oscillations across the ravine, and if it is small the progress along the ravine is very slow. This effect is shown in the next section.

The generalization property of the CMAC is shown in fig.2 using a simple trigonometric function, *sin(x)*. The input range was 0 to 360 degree and the interval of the sampled input is 5 degree. The CMAC net was trained with 73 sampled input-output pairs and the trained net was used to obtain the response at every one degree over the whole input space. Resulting errors were compared with linear interpolated values of the results of trained sample nodes. From this we can see the CMAC trained net automatically generates the linear interpolating behavior. The on-line SEC learning was executed for 300 epochs. The convergence of the rms and maximum error over the sampled and the total input nodes is shown in fig.3 with respect to the number of learning epochs.

In pratice, same as other neural networks the most serious problem of the CMAC is the speed of convergence. How long and how much memory it might take a system to learn is the main concern. In the next section, three basic learning algorithms are presented and the performance of these are compared with the MEC learning. Learning features which are application dependent are also defined.

4. Training Algorithms

Training algorithms can be classified into a sequential error correction(SEC) and a random

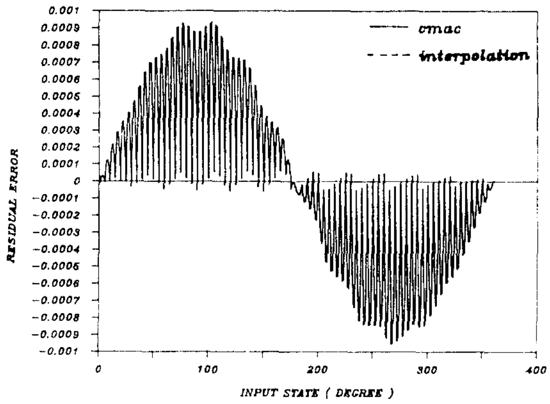


Fig.2 Error distribution of the trained CMAC net for *sin(x)*(on-line type SEC learning: G=0.8, K=30, Offset=1, Sampled interval=5 deg, Epoch=300).

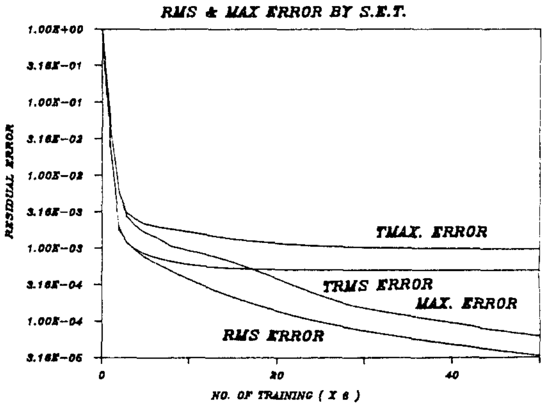


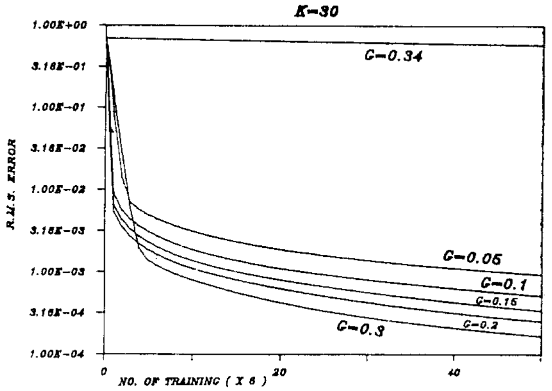
Fig.3 Convergence trends of rms and max error over sampled(5deg) and total(1deg) input nodes with respect to the trained epochs.

error correction(REC) according to the selection of the input nodes to be trained. Choosing one of the two is application dependent. In general, the SEC learning requires error measures over all sets of input and output pairs for one learning epoch. Since the computing overhead required for obtaining errors over the input space is very heavy when a system function to be trained is computationally complex, the SEC learning is proper to the application whose errors are measured and kept on-line. This is especially true for a function evaluated by the numerical iteration and for a large number of input-output pairs.

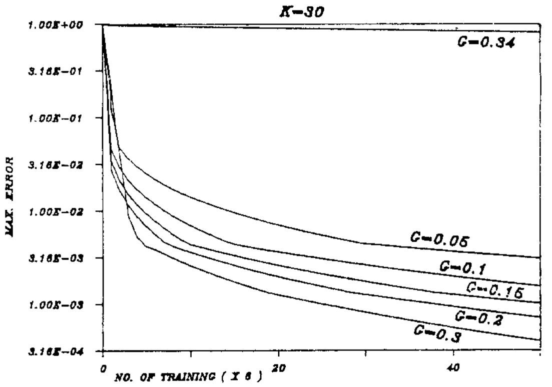
In the previous section, the convergence of the CMAC was proven with the batch and on-line type SEC learning. Here, three basic learning and the MEC learning algorithms were executed using various learning gains for *P=sin(x)* with the range of $0 \leq x \leq 360$(deg) and K=30 for the equivalent cpu learning time. Sampled input patterns were selected at every 5 degree interval resulting 73 sampled input nodes. The resulting performance of the rms and maximum errors are obtained over sampled input nodes and compared each other.

The batch type SEC learning guarantees the convergence of the accumulated rms error without forcing any oscillation per epoch once the learning gain is properly selected. Since training is done only once per epoch with the accumulated errors, it converges relatively slow compared to the on-line type SEC learning. Trained results of the CMAC batch type SEC learning are shown in fig.4. At G=0.34 the system diverges and it does not learn.

At the first epoch, the training performance is checked searching the best gain to avoid



a:rms error



b:maximum error

Fig.4 Batch type SEC learning for $P=sin(X)$ with K=30 (a:rms error b:maximum error).

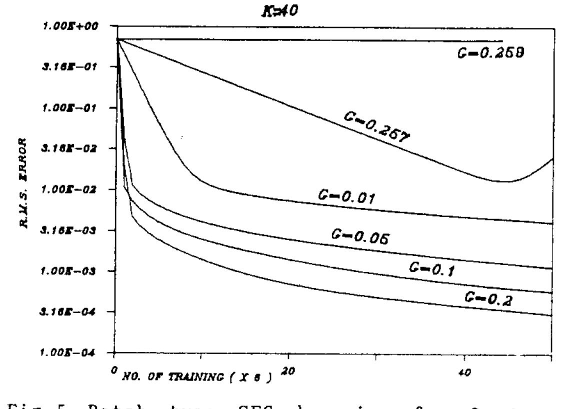
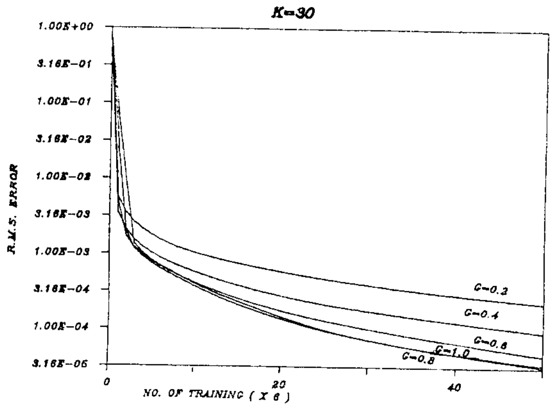


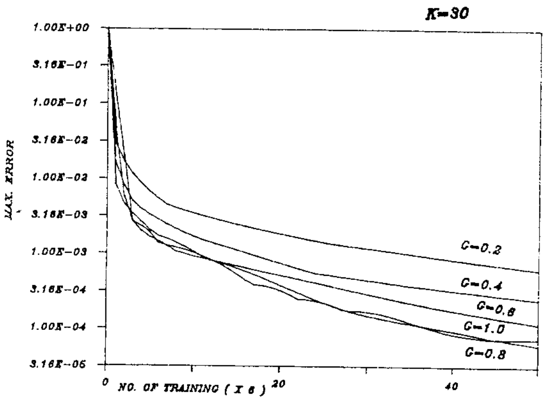
Fig.5 Batch type SEC learning for $P=sin(X)$ with K=40.

the divergence. In practice, this process is good enough to guarantee the system convergence. Initially the high and low end gains exhibit rather a slow converging behavior because of the interference and generalization properties respectively. However, as far as the system convergence is preserved, the large gain has the best performance after all. Trained results with K=40 in fig.5 show the diverging and converging effects mentioned in the previous section clearly.

In a case of on-line type SEC learning the connected weights are changed at each pattern presentation. Although the performance measure of the accumulated rms error oscillates at each pair presentation, it converges fast with little oscillation at every epoch. The trend of learning results are similar to that of the batch type SEC learning except the diverging behavior as shown in fig.6.



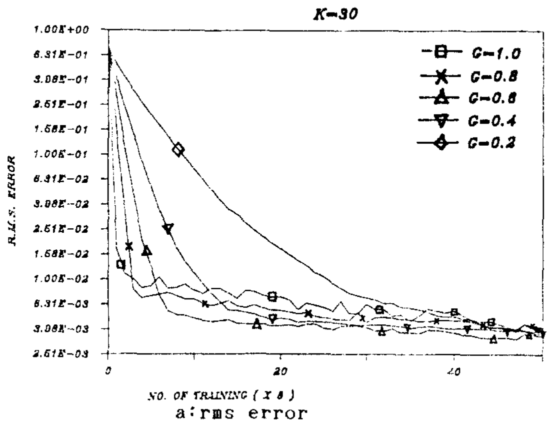
a:rms error



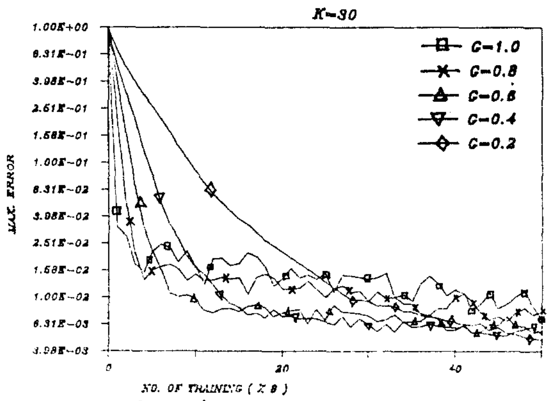
b:maximum error

Fig.6 On-line type SEC learning for $P=sin(X)$ with K=30 (a:rms error b:maximum error).

The MEC learning proposed by Albus can be thought as a modified version of the on-line sequential type LMS learning. Although the correction effect of an individual pattern presentation is the greatest of all, it requires almost the same computational load because of the searching effect for the maximum error node over all input nodes presented. The learning progress is quite slow and resulting system performance is poor



a:rms error



b:maximum error

Fig.7 MEC learning for $P=sin(X)$ with K=30 (a:rms error b:max error).

compared to the SEC learning because learning is actually done once per epoch. Furthermore, the MEC learning can not avoid the oscillating features as the number of training increases. The trend of learning is shown in fig.7. As the learning gain increases, it learns relatively faster but oscillation starts ealier.

Three learning algorithms mentioned above have restriction of learning for the fixed node input-output pairs. The REC learning which is one of quite natural mechanisms is not restricted to the fixed input nodes but can be applied to the continuous real input nodes. Since it does not restrict the input space nodes, the CMAC mapping acting as an analog to digital converter intgrates the trained errors. It can train the nonlinear behavior among the fixed input nodes.

Two types of generating random inputs can be considered. One is generating a random sequence input node from the sampled input patterns and the other is from the total input patterns. The input space which is converted to the binary input pattern space has an infinite number of state vectors. If the supervised output is obtained directly from the input state vectors, the learned error is integrated. Since the computing overhead is not significant in this case, the initial performance of learning is quite good but it can not avoid the oscillating behavior similarly as the MEC learning. However, this learning is appropriate for a large input-output pairs and when the desired function values are not measured on-line. The advantages of REC learning can be adopted by the SEC learning in the case of on-line type learning.

A random number generator was modified to allow a uniform random deviation between 0 and 1 by adding an additional shuffle on the numbers generated by a VAX 11/750 system-supplied routine. This routine is effectively free of the sequential correlation[13]. The generalizing effect is shown in fig.8 when

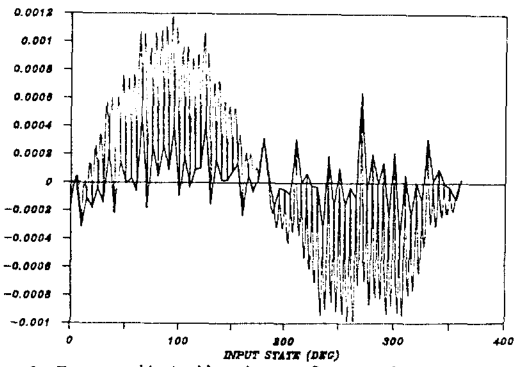
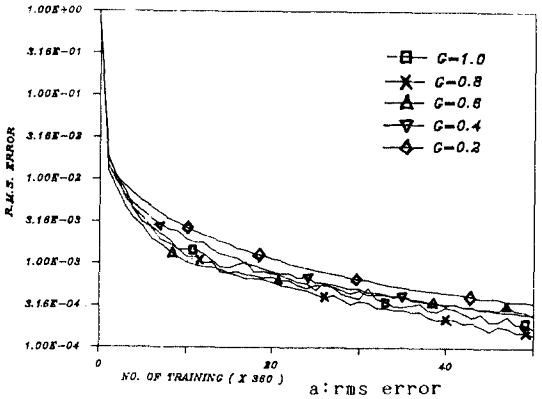
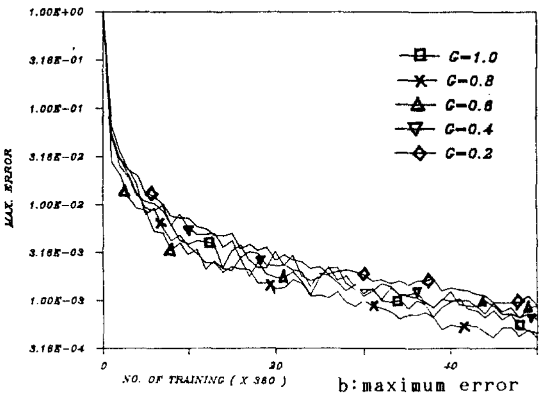


Fig.8 Error distribution of REC learning for $P=sin(X)$ (K=30,G=0.8 Offset=1,Sampled interval=5 deg,number of training=18000)



a:rms error



b:maximum error

Fig.9 REC learning for $P=sin(X)$ with K=30 (a:rms error b:maximum error).

a linear interpolating results for the untrained input nodes located among sampled nodes. With proper number of sampled input node inputs, the CMAC can learn the desired system behavior arbitrarily closely.

Two types of learning, REC and SEC, were presented and their learning performances were compared with the MEC learning. The MEC learning had the poorest performance of all because of its learning charateristics. The SEC learning provided the basic tool to improve or develop the better learning algorithm. The SEC and MEC learning have their restrictions such that they accept the prespecified fixed node inputs. The REC learning, however, overcomes this fact and can also accept the non-fixed real node inputs. A uniform quantizing method was devised to cope with the various ranges of input variables and corresponding resolutions efficiently.

The performance of the proposed learnings was quite good enough to implement the memory driven control system according to the simulation results performed. The required system memory for distributed trained data storage was enourmously small compared to normal table look-up type storage.

The presented results of the CMAC analysis on learning will accellerate and extend its application to robotics field such as motion planning and control, calibration, sensor fusion including vision, and obstacle avoidance. Application of the hybrid type of the CMAC plus conventional control and the hierarchical CMAC network for information processing are also recommended.

Since the effects of the control parameters on the CMAC learning algorithm are critical, interrelations between these parameters and the shapes of the system function to be trained should be investigated with its learning performance. In the case that the unknown desired relations between the input and output are to be trained, it is desirable to set a certain guideline for the application of the CMAC.

7. ACKNOWLEDGMENT

The authors gratefully acknowledge the Korea Institute of Machinery and Metals' Basic and Advanced Research Assistance Program.

8. REFERENCES

[1] Albus, J.S.,'Theoretical and Experimental Aspects of a Cerebellar Model,' Ph.D. Thesis, University of Maryland,Dec. 1972.

[2] Albus, J.S.,'A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller(CMAC),' Journal of Dynamic Systems, Measurement, and Control, Tran. of the ASME, Vol.97, No.3, Sept. 1975, pp.220-227

[3] Albus, J.S.,'Data Storage in the Cerebella Model Articulation Controller(CMAC) ,'Journal of Dynamic Syatems, Measurement, and Control, Tran. of the ASME, Vol. 97, No.3, Sept. 1975, pp.228-233.

[4] Albus, J.S.,'Mechanisms of Planning and Problem Solving in The Brain,' Mathematical Biosciences, 45, 1979, pp.247-291.

[5] Albus J.S.,'A Model of The Brain for Robot Control,' BYTE Magazine, July 1979, pp.54-95.

[6] Camana, P.C.,'A Study of Physiologically Motivated Mathematical Models for Human Postural Control,' Ph.D. Thesis, Dept. of Electrical Engineering, Ohio State University, 1977.

[7] Rajadhyaksha, J.,'A 2-D Adaptive Multilink CMAC Manipulator Simulation,' MSME Thesis, Dept. of Mechanical Engineering, Louisiana State University, 1985.

[8] Manglevhedakar, S.,'An Adaptive Hierarchical Model for Computer Vision,' MSME Thesis, Dept. of Mechanical Engineering, Louisiana State University, 1986.

[9] Miller III, W.T.,'Sensor-Based Control of Robotic Manipulators Using A General Learning Algorithm,' IEEE Journal of Robotics and Automation, Vol. RA-3,No.2,April 1987, pp.62-75.

[10] Miller III, W.T., F.H. Glantz, and L.G. Kraft, 'Application of A General Learning Algorithm to The Control of Robotic Manipulators,' Int. Journal of Robotics Research Vol.6, No.2, Summer 1987, pp.84-98.

[11] Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1, D.E. Rumelhart and J.L. McClelland(Eds), Cambridge, MA:MIT Press, 1987, pp.318-362.

[12] Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1, D.E. Rumelhart and J.L. McClelland(Eds), Cambridge, MA:MIT Press, 1987, pp.365-422.

[13] Numerical Recipes: The Art of Scientific Computing, Press W.H. et. al.(Eds), Cambridge Univ. Press, 1986, pp.191-199.

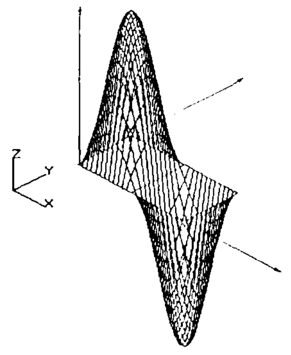
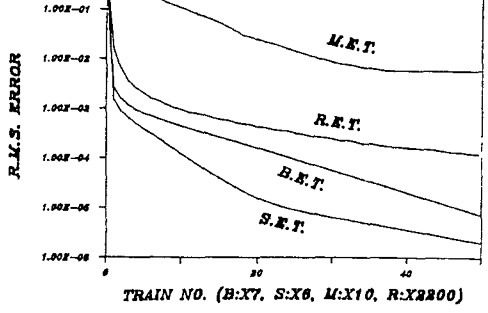


Fig.11 A nonlinear function to be trained $P=sin(x)sin(y)$ with $0 \leq x \leq 360$ $0 \leq y \leq 180$ (deg).



using the REC learning performed on the sampled input nodes by 5 degree. Trends of the rms and maximum errors are shown in fig.9. It is seen that as the number of training increases, the effect of the gain gets smaller.

The rms performance of all four algorithms mentioned so far has been compared in fig.10 for the equivalent training period. The SEC learnings show the very good converging trend and system performance. The REC learns fast at the early stage but it has a limitation caused by the randomly distributed inputs without considering learned history of the whole input space.

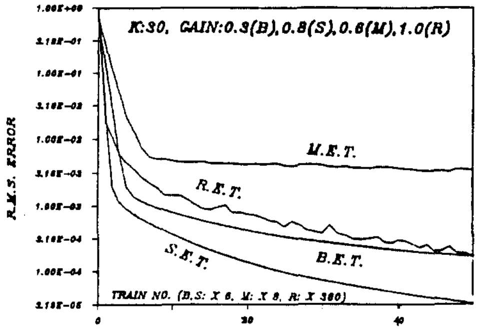


Fig.10 Performance of various learning algorithms on the rms errors over the sampled input nodes($P=sin(x)$,K=30).

5. LEARNING SIMULATIONS

Using the previously mentioned learning algorithms, a nonlinear function geneartor and a motion generator for a two d.o.f. manipulator were simulated. Trained results were compared using the rms and maximum errors over the sampled input nodes for the equivalent learning period. The rms and maximum errors over the extended input nodes were also compared to show the generalizing effect of each learning. The uniform quantizing scheme mentioned in section 2 was applied.

5.1 Nonlinear Function Generator

A nonlinear function, $P=sin(x)sin(y)$(fig.11) was trained with input ranges of $0 \leq x \leq 360$ and $0 \leq y \leq 180$ using the batch type SEC(G=0.5), on-line type SEC(G=0.8), MEC(G=0.8), and REC(G=0.4) learnings. The interval of sampled input nodes was 15 degrees resulting 325 pairs of inputs. The offset of the CMAC input space was 1 degree and K was equal to 30. The learning gain of the batch type SEC was selected from the scheme presented in section 4.

The rms and maximum errors were compared over the sampled input nodes and the extended input nodes of 3 degree interval and those are shown in fig.12 and fig.13 respectively. The SEC and REC learning shows the excellent error fitting on the sampled input nodes. However, they show rather poor results on the overall node space sampled by 3 degree. This reveals their linear interpolating effect on the intermediate untrained nodes, which are connected as nonlinear surface. Maximum and rms error have almost same trends as the number of training epoch is increased. The rms error oscillates in the REC and MEC learnings as learning number is increased although it is not shown clearly in fig.12. The REC learning shows a rather large oscillating behavior in maximum error because of its randomly selected correction without considering the slope of overall error surface. The MEC learning shows the worst learned rms error performance and the oscillating behavior in maximum error even it corrects the selected node of maximum error at each epoch.

5.2 Joint Movement of Two D.O.F. Manipulator

Two d.o.f. planar manipulator was trained to move from the specified initial arm state to the target point in the work space by training the required joint movements directly. The initial arm configuration was set such that $\theta_1=\pi/4$ and $\theta_2=-\pi/4$. Since two different arm configurations exist for every target point, the final arm configuration and the entire joint movements were determined from the initial joint states and the location of the target. The CMAC input space is the work space of the manipulator and was defined by two input variables using polar variables, $r \in [300,700]$ in mm unit and $\phi \in [-150,150]$ in degree unit. Fig.14 shows configurations of two seperate desired joint movement and aspects of discontinuity caused by the desired arm pattern.

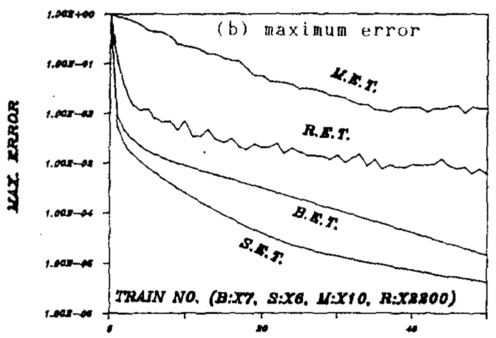
Two seperate sets of the CMAC weights with a common input space were trained to handle discontinuity caused by two different target configurations. Each CMAC set has two sub-CMAC joint movement controllers resulting in one CMAC input and two CMAC trained outputs. Four differnt learnings such as batch SEC(G=0.05), on-line SEC(G=0.8), MEC(G=0.6), and REC(G=0.6) were applied for the equivalent training period. The quantizing value K and the offset of the CMAC input space were set to be 150 and 1 respectively. The interval of the sampled input nodes were 20 mm and 20 degrees resulting in 336 input pairs.

The rms and maximum errors were obtained from the joint and world space and are shown in fig.15 and fig.16 respectively. The oscillating behavior of the rms error occurs in the BET in the world space at the early stage although it is not shown clearly in fig.16. The reason is mainly due to the unmatched relation of converging directions between the joint space and the world space. The SEC and REC learnings exhibit the excellent trained performance while the MEC shows a poor result as expected. However, if the input pairs are increased, the required training cpu time will be the problem as mentioned earlier as a restriction of the SEC learnings.

6. CONCLUSION

The convergence of the CMAC has been proved identifying the CMAC network as a kind of one-layer linear associator having a linear activation function. Trained results from simulating various functions showed the coincident converging features. The CMAC structured mapping has its merit of converting the continuous or discrete input state vectors into the linear independent binary pattern vectors, which is required for the delta learning rule.

The trained CMAC network using the sampled input pattern vectors automatically generates



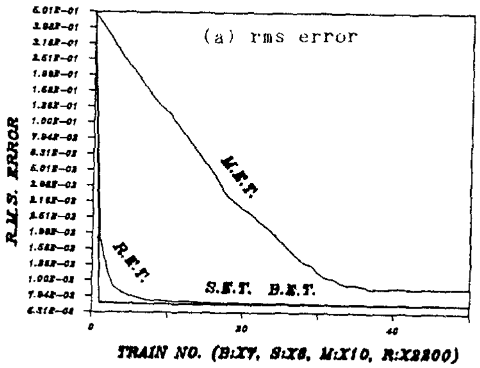


Fig.12 The performace using various learning over sampled input nodes: (a) rms error (b) maximum error (Sampled interval= 15 deg, Batch SEC :G=0.5, On-line SEC :G=0.8, MEC :G=0.8, REC :G=0.4).

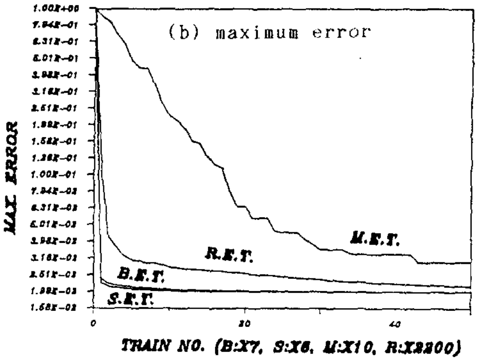


Fig.13 The performace using various learning over extended input nodes: (a) rms error (b) maximum error (Sampled interval= 3 deg, Batch SEC :G=0.5, On-line SEC :G=0.8, MEC :G=0.8, REC :G=0.4).

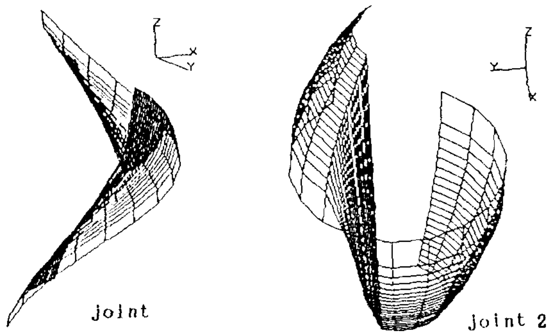
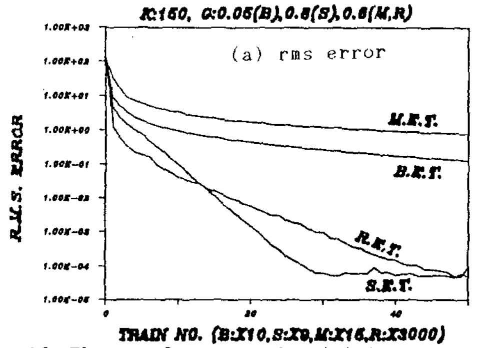


Fig.14 Joint movements to be trained with $300 \leq r \leq 700$(mm) and $-150 \leq \phi \leq 150$ (deg).



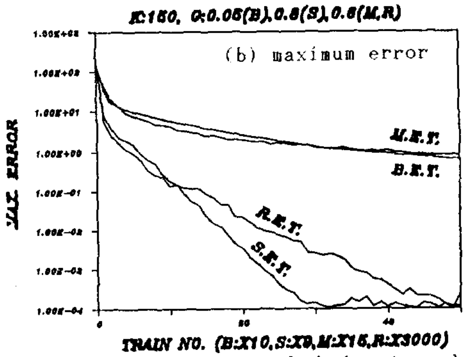
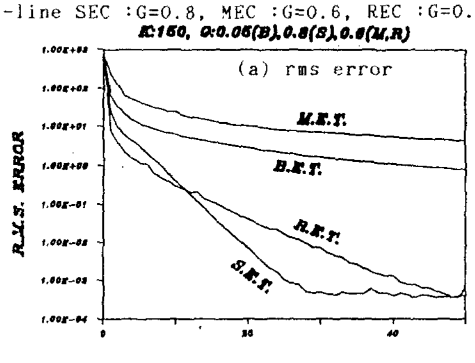


Fig.15 The performance in joint space using various learnings over sampled input nodes: (a) rms error (b) maximum error (Sampled interval= 20 deg and 20 mm, Batch SEC :G=0.05, On-line SEC :G=0.8, MEC :G=0.6, REC :G=0.6).



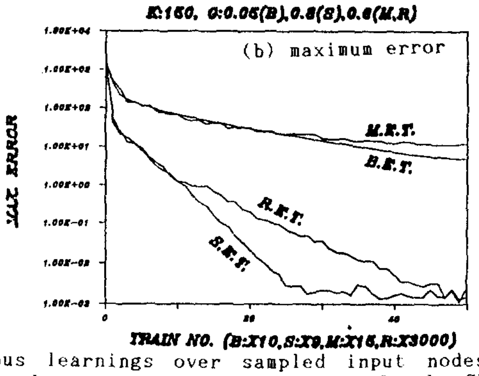


Fig.16 The performance in world space using various learnings over sampled input nodes: (a) rms error (b) maximum error (Sampled interval= 20 deg, and 20 mm, Batch SEC :G=0.05, On-line SEC :G=0.8, MEC :G=0.6, REC :G=0.6).