소뇌모델 선형조합 신경망의 구조 및 학습기능 연구(I) —분석 및 학습 알고리즘 개발—

On Learning and Structure of Cerebellum Model Linear Associator Network (I) —Analysis & Development of Learning Algorithm—

황 헌* 백 풍 기**
H. Hwang, P. K. Baek

적 요

인간 소뇌의 구조와 기능을 간략하게 수학적으로 모델링하여 입력에 따른 시스템의 적정 출력을 학습에 의한 적응 제어방식으로 추출해 내는 소뇌모델 대수제어기(CMAC : Cerebellar Model Arithmetic Controller)가 제안되었다. 본 논문에서는 연구개발된 기존 신경회로망과의 비교 분석에 의거하여, 소뇌모델 대수제어기 대신, 네트의 특성에 따라 소뇌모델 선형조합 신경망(CMLAN : Cerebellum Model Linear Associator Network)이라 하였다. 소뇌모델 선형조합 신경망은 시스템의 제어 함수치를 결정하는 데 있어, 기존의 제어방식이 시스템의 모델링을 기초로 하여 알고리즘에 의한 수치해석적 또는 분석적 기법으로 모델 해를 산출하는 것과 달리, 학습을 통하여 저장되는 분산기억 소자들의 함수치를 선형적으로 조합함으로써 시스템의 입출력을 결정한다. 분산기억 소자로의 함수치 산정 및 저장은 소뇌모델 선형조합 신경망이 갖는 고유의 구조적 상태공간 매핑(State Space Mapping)과 델타 규칙(Delta Rule)에 의거한 시스템의 입출력 상태함수의 학습으로써 수행된다.

본 논문을 통하여 소뇌모델 선형조합 신경망의 구조적 특성, 학습 성질과 상태공간 설정 및 시스템의 수렴성을 규명하였다. 또한 기존의 최대 편차수정 학습 알고리즘이 갖는 비능률성 및 적용 제한성을 극복한 효율적 학습 알고리즘들을 제시하였다. 언급한 신경망의 특성 및 제안된 학습 알고리즘들의 능률성을 다양한 학습이득(Learning Gain)하에서 비선형 함수를 컴퓨터로 모의 시험하여 예시하였다.

1. INTRODUCTION

On close examination of the simple or complex manipulating and perceiving behavior such as those performed by biological organisms, the computational methods involved in engineering manipulator control or image processing problems have serious shortcomings. They fail to produce a truly sophisticated and adaptive behavior. The degree of difficulty experienced in obtaining mathematical solutions even for trivial actions performed by ordinary organisms is very high. However, it is al-

* 成均館大學校 農業機械工學科
** 建國大學校 農業機械工學科

most certain that the biological organisms do not solve or model the complex mathematical formulation for such complicated 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 systems such as visual image processing for the part inspection or image understanding and sensor based robot control, which require the real time recognition of the extracted feature and the real time computation of the parameter identification based on some performance criteria with the proper management of sensitivity on sensory inputs, the robust adaptive controller based on the biological structure and function has 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 have been involved in functional and structural modeling of information processing of the human brain (1). Artificial neural net models have been studied by scientists in various fields for many years(2). How to achieve a great degree of the robustness, adaptation, and easy learning is the major focus in this area which also requires high computation rates.

Anatomical and neurophysical 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 Arithmetic Controller(CMAC) by Albus(3-6). CMAC is a schematical approximate modeling of the information processing characteristics of the cerebellum. Through a series of storages or learnings, CMAC works as a computational module generating weights in a distributed table look-up manner connected in parallel.

CMAC has been applied to several control applications and revealed its usefulness at various levels in control problems(1, 3 7-12). Although CMAC was applied to various applications as a control substitute or a reference input generator and visual image recognizer, a detailed analysis of CMAC mapping and network with a learning capability was not performed yet. The convergence of CMAC has been anticipated from the experimental simulation. From the analysis and comparison of CMAC net with the linear associator which is one of well known neural network models, CMAC is renamed as CMLAN (Cerebellum Model Linear Associator Net) in this paper.

CMLAN 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.

This paper presents the extended view and detailed analysis on CMLAN such as mapping structure, learning process, system memory requirements, property of interference and continuity(generalization), and convergence which has not been proved formally yet(3, 12). 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.

2. ANALYSIS ON CMLAN

This section briefly explains about the structure and function of the computational CMLAN module, and presents the extended view and detailed analysis on CMLAN.

Structure and Function

CMLAN 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 CMLAN generated approximate discrete system responses in a distributed manner by forcing those differences to be near zero. In fact, to generate system output or relationship of the interior system parameters CMLAN can be applied to any system which has input-output relations such as $P=H(S)$, where P and S are output and input state vectors respectively and H is a mapping function.

CMLAN 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, CMLAN can not maintain the unique input and output relation. CMLAN maps or discretizes, however, successfully an infinite number of input states. An iterative learning scheme of CMLAN can be thought as a powerful substitute of LMS(Least Mean Square) error procedure.

In a word, main characteristics of the CMLAN 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 a non-linear activation function, CMLAN can be applied to a case of the continuous valued inputs and the discrete binary output pattern vectors.

CMLAN 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 is composed of N variables which can be continuous or discrete.

A range of each S_i is segmented by K_i, the prespecified quantizing block for input variable i, resulting in quantizing functions ${}^iQ_j$ for $j=1$ to K_i, which represents the number of the quantized blocks at layer j of input 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, ${}^iN_T$ can be obtained such that.

$${}^iN_T = S_{Ri} + K_i$$

The number of elements in association vector A, denoted by $|A|$ and concatenated from N input variables can be obtained by assuming $K=K_i$ for $i=1,2,3\cdots, N$ such that

$$|A| = \sum_{j=1}^{k} {}^1Q_j\, {}^2Q_j \cdots {}^NQ_j,$$

where $|A|$ also denotes the required CMLAN system memory. CMLAN 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 vetor, 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 the 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 CMLAN 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 CMLAN 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 CMLAN, every input variable is actually considered to be nondimensional and the only factors to be considered for a structured mapping are the mapped CMLAN 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 CMLAN 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 between the desired function value and the CMLAN generated approximate value is computed. Comparing it with the predefined tolerance and choosing an integer node at which a maximum error occurs, the CMLAN 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.

$$\triangle = G\left(\frac{F(S)_m - CMLAN_m}{|A^*|}\right)$$

where G = training gain,

S_m = input node vector where the maximum error occurs,

$F(S_m)$ = desired function value at Sm,

$CMLAN_m$ = CMLAN generated function value at S_m,

$|A^*|$ = 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. 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.

CMLAN has an inherent training difficulty for functions having discontinuities. A function having a discontinuity should be handled by using the separate CMLANs or by taking trained results apart in the CMLAN input space. A function for CMLAN 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 CMLAN, generalization and interference.

A propety of generalization or continuity is simply explained as similar inputs produce similar responses because of the overlapping nature of the CMLAN 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.

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 CMLAN structured mapping is a kind of coarse coding but has a unique coding characteristics. As mentioned earlier, the CMLAN 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 state space is same as the number of the CMLAN 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 linearly independent binary pattern vectors are formed by the CMLAN 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 CMLAN resembles exactly a linear associator with a delta learning rule as shown in fig. 1. Since the mapped input binary pattern vectors are linearly independent each other, CMLAN satisfies the requirements of the delta rule learning. the CMLAN weight connection can be expressed in a matrix form as a system of linear equations such that

$$[X_{p1}\ X_{p2} \cdots X_{pM}]\ [W_1\ W_2 \cdots W_M]^T = T_P$$

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

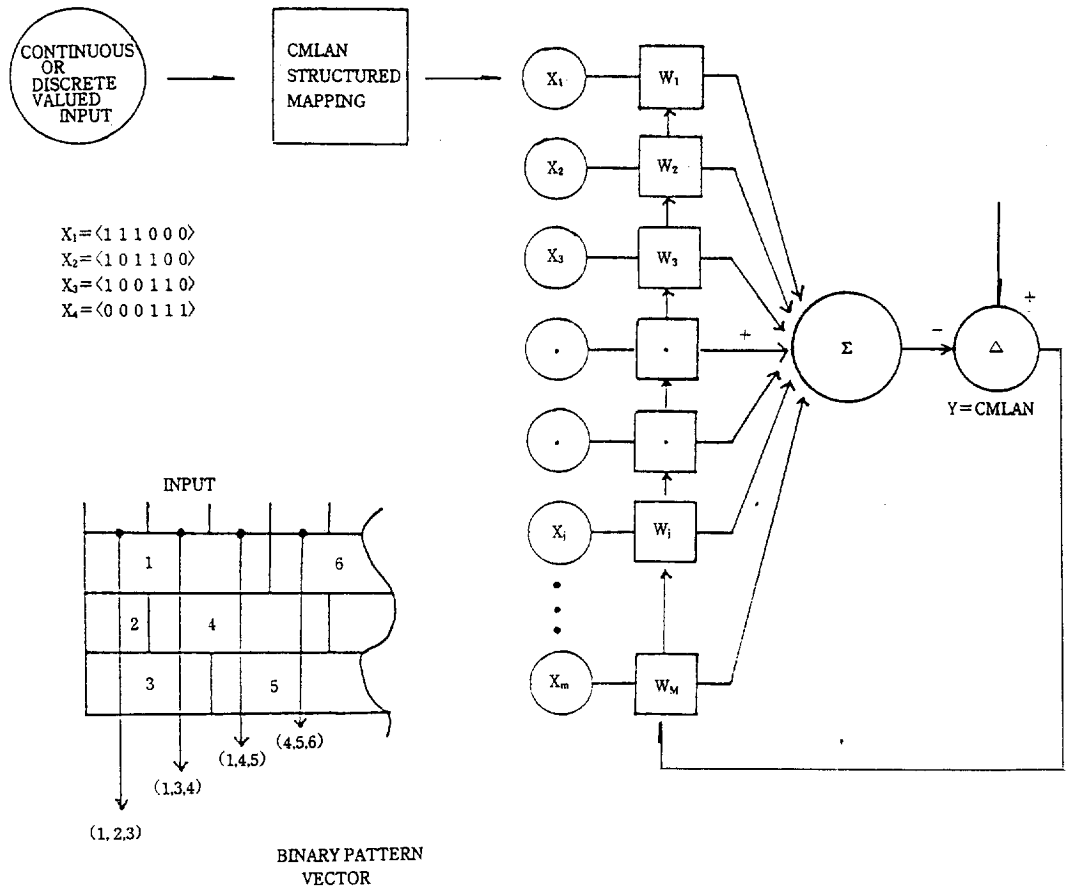


Fig. 1. CMLAN network.

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 minmum, 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 the steepest descent on the error surface[13].

The convergence of the CMLAN can be proven similary as a case of multi-layer LMS learning network. CMLAN can be thought as one layer network being connected whose 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[14]. 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 = CMLAN generated output.

The overall error is then,

$$E = \sum_{P=1}^{N} E_p$$

For a specific case,

$y_i = x_i$ for $i = 1, 2, \cdots, M$

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

Since CMLAN 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,

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

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 oscialating 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 casese of similar inputs with different outputs. Because of the generalization property of CMLAN, the interference is rather occurred when the discontinuity or serious functional change occurs in the output values within the neighborbood of generalization. Also the 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 CMLAN 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. CMLAN 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. Since linear interpolated results(dot line) are almost overlapped to the CMLAN generated errors, it can not be seen clearly. From this we can see the CMLAN 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 practice, same as other neural networks the most serious problem of CMLAN is the speed of convergence. How long and how much memory it might take for a system to learn is the main concern. In the next section, three basic learning algoithms are presented and the performance of these are compared with the MEC learning. Learning features which are application dependent are also defined.

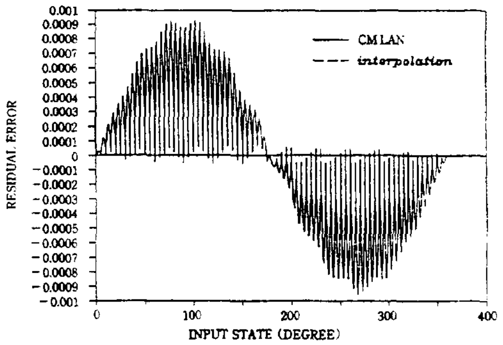


Fig. 2. Error distribution of the trained CMLAN net for *sin(X)* (on-line type SEC learning : G=0.8, KI=30, Offset=1, Sampled interval=5 deg, Epoch=300).

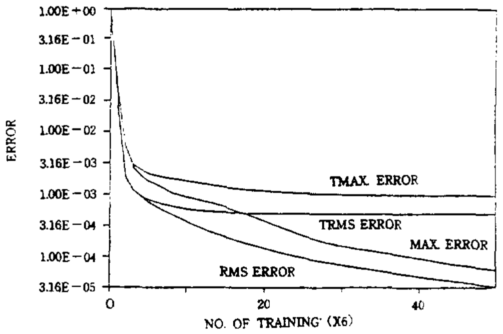


Fig. 3. Convergence trends of rms and max error over sampled (5deg) and total(1deg) input nodes with respect to the trained epochs.

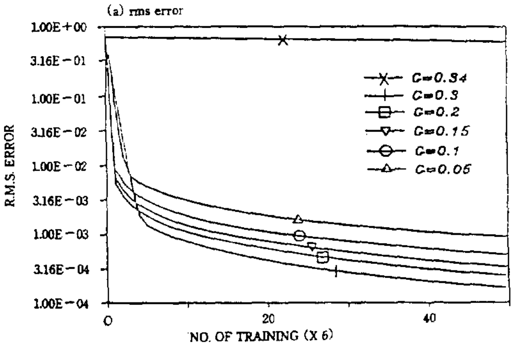
Training Algorithms

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

Previously the convergence of CMLAN 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 ranger of $0 \leq x \leq 360$ (deg) and K=30 for the equivalent cpu learning time. The cpu time elapsed was measured using VAX 11/750 run-time library routine. It took around 0.33 seconds per learning epoch for both batch and on-line type SEC learnings. It took 0.26 seconds per epoch and 0.0055 seconds per each random learning for MEC and REC respectively. Sampled input patterns were selected at every 5 degree interval resulting 73 sampled input nodes. The resulting performance of the rms and maxmum 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 CMLAN batch type SEC learning are shown in fig. 4. At G=0.34 the system diverges and it does not learn. To aviod the divergence, the training performance is checked at the first epoch searching the best gain which gives the fastest convergence and then the gain slightly above the best is chosen. 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.



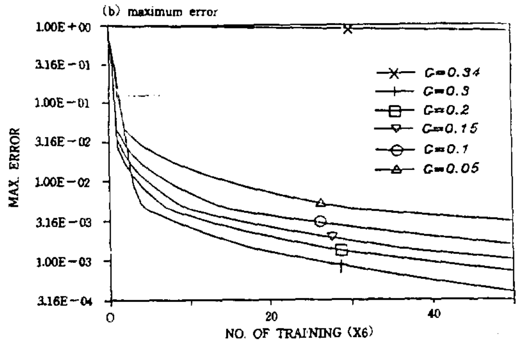


Fig. 4. Batch type SEC learning for $P=sin(X)$ with K=30 (a : rms error, b : maximum error).

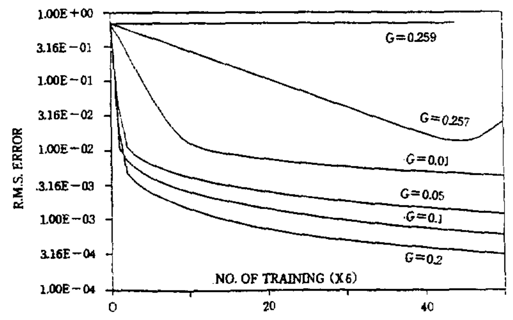
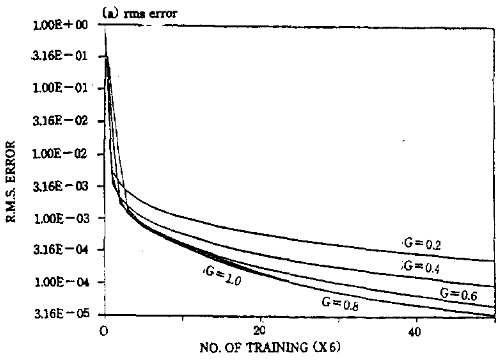


Fig. 5. Batch type SEC learning for $P=sin(X)$ with K=40.

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. The parallel trend of slopes for the various gains is maintained until it reaches its global minimum. It is not plotted but converged learned rms errors from the unlimited learning epoch with G=0.5 and G=0.01 were 1.395131E−6 at the epoch of 19000 and 7.898105E−6 at the epoch of 55500 respectively.

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 is similar to that of the batch type SEC learning except the diverging behavior as shown in Fig. 6.

However, it should be noted that the smaller gain catches up with larger gain as learning progresses. This is contrary to the result of the batch



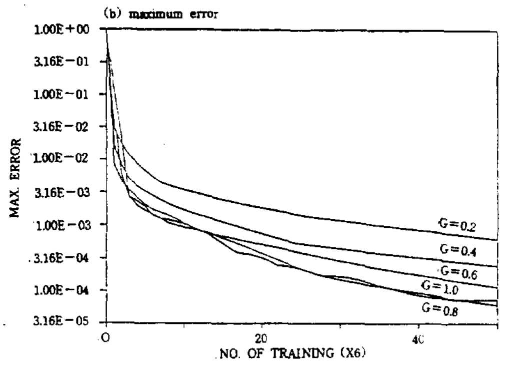


Fig. 6. On-line type SEC learning for *P=sin(X)* with K=30 (a : rms error, b : maximum error).

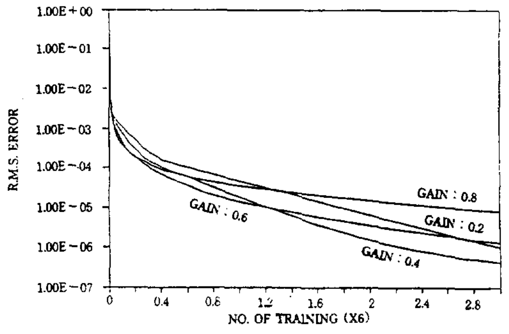


Fig. 7. On-line type SEC learning for *P=sin(X)* with K=40.

type SEC. Fig. 7 shows trends of catching in learned performance under various gains with a fixed K of 40 when learning epoch is extended up to 3000. Note. however, practically the learning period is also critical to the system performance as well. For this reason, it is not recommended to reduce the gain value very small based on the results of fig. 7 when applying on-line SEC learning.

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 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. 8. As the learning gain increases, it learns relatively faster but oscillation starts earlier.

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 CMLAN mapping integrates the trained errors in handling very large input space.

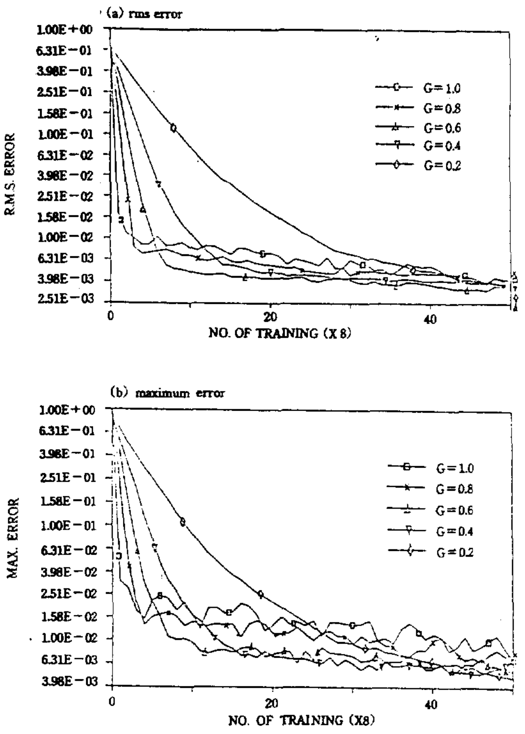


Fig. 8. MEC learning for $P=sin(X)$ with K=30 (a : rms error, b : max error)

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 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 large input-output pairs and when the desired function values are not measured on-line. The advantages of REC learning might be adopted by the SEC learning.

A random number generator was modified to allow a uniform random deviation between 0 and 1 by adding an additional shuffle on the number generated by a VAX 11/750 system supplied routine. This routine is effectively free of the sequential correlation[15]. Trends of the rms and maximum errors resulted from the REC learning performed on the sampled input nodes by 5 degree 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 and REC 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 the learned history of the whole input space.

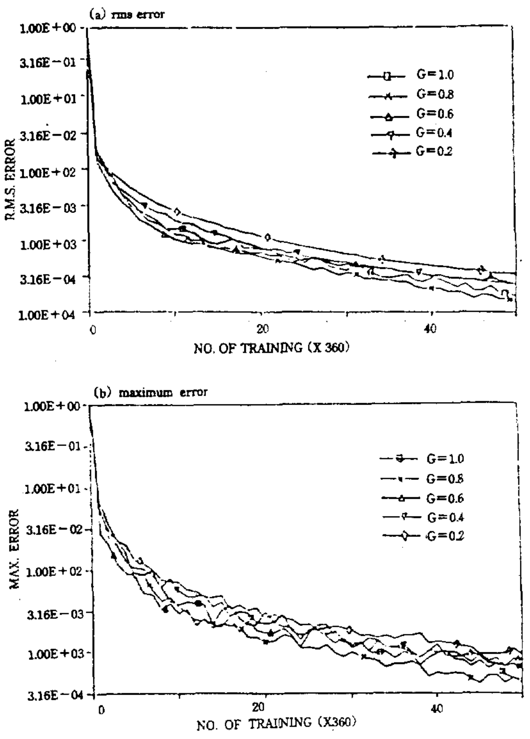


Fig. 9. REC learning for $P=sin(X)$ with K=30 (a : rms error, b : maximum error).

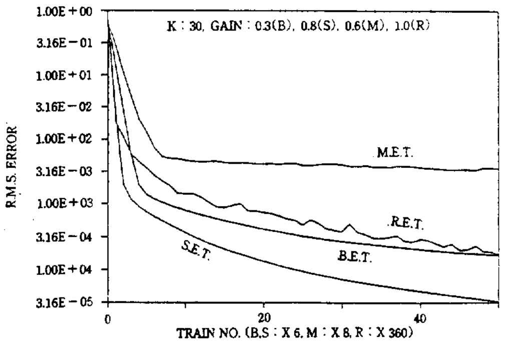


Fig. 10. Performance of various learning algorithms on the rms errors over the sampled input nodes $P=sin(X)$, K=30).

3. CONCLUSTION

The convergence of CMLAN has been proved identifying it 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 CMLAN 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 CMLAN network using the sampled input pattern vectors automatically generates linear interpolating results for the untrained input nodes located among sampled nodes. With proper number of sampled input node inputs, CMLAN can learn the desired system behavior arbitrarily close .

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

The performance of the proposed learning algorithms 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 enormously small compared to normal table look-up type storage.

The presented results of the CMLAN analysis on learning will accellerate and extend its engineering application to the various fields.

REFERENCES

1. DARPA Neural Network Study. 1987. AFCEA Int. Press : 123−133, 445−450.
2. Anderson, J.A. and E. Rosenfeld. 1988. Neurocomputing : Foundations of research. MIT Press, Cambridge, MA.
3. Albus, J.S. Dec 1972. Theortical and experimental aspects of a cerebellar model. Ph. D. Thesis, Univ. of Maryland.
4. Albus, J.S. Sept. 1975. A new approach to manipulator control : The cerebellar model articulation controller(CMAC). Journal of Dynamic Systems, Measurement, and Control, Trans. of the ASME, Vol. 97, No. 3 : 220−227.
5. Albus, J.S. Sept. 1975. Data storage in the cerebella model articulation controller(CMAC). Journal of Dynamic Systems, Measurement, and Control, Trans. of the ASME, Vol. 97, N. 3 : 228−233.
6. Albus, J.S. 1979. Mechanisms of planning and problem solving in the brain, Mathematical Biosciences, 45 : 247−291.
7. Albus, J.S. June 1979. A model of the brain for robot control. BYTE Magazine : 54−95.

8. Camana, P.C. 1977. A study of physiologically motivated mathematical model for human postural control. Ph. D. Thesis, Dept. of Electrical Eng., Ohio State Univ.
9. Rajadhyaksha, J. 1985. A 2－D adaptive multi-link CMAC manipulator simulation. MSME Thesis, Dept. of Mechanical Eng., Louisiana State Univ.
10. Manglevhedakar, S. 1986. An adaptive hierarchical model for computer vision. MSME Thesis, Dept. of Mechanical Eng., Louisiana State Univ. 11. Miller III, W.T. April 1987. Sensor-based control of robotic manipulators using a general learning algorithm. IEEE Journal of Robotics and Automation, Vol. RA－3, No. 2 : 62－75.
12. Miller III, W.T., F.H. Glantz and L.G. Kraft. Summer 1987. Application of a general learning algorithm to the control of robotic manipulators. Int. Journal of Robotics Research Vol. 6, No. 2 : 84－98.
13. Rumelhart, D.E. and J.L. McClelland. 1987. Parallel distributed processing : Explorations in the microstructure of cognition. Vol. 1, MIT Press, Cambridge, MA : 318－362.
14. Rumelhart, D.E. and J.L. McClelland. 1987. Parallel distributed processing : Explorations in the microstructure of cognition. Vol. 1, MIT Press, Cambridge, MA : 365－422.
15. Press W.H. et. al. 1986. Numerical recipes : The art of scientific computing. Cambridge Univ. Press, 191－199.

祝

學位取得

姓　　名	김　영　중
生年月日	1954年 9月 26日
勤務處	연암공업전문대학
取得學位名	공학박사
學位取得年月日	1989年 12月 15日
學位授與大學	Texas A&M University
學位論文	Clostridial Fermentation of High Energy Sorghum