Neural Processing Letters **20:** 1–10, 2004.

A Modified CMAC Algorithm Based on Credit Assignment

LEI ZHANG[1], QIXIN CAO[1], JAY LEE[2] and YANZHENG ZHAO[1]
[1]*Research Institute of Robotics, Shanghai Jiao Tong University, Shanghai, 200030, China. e-mail: zhanglei@sjtu.edu.cn*
[2]*Research Center of Intelligent Maintenance Systems, University of Wisconsin-Milwaukee, WI 53224, USA*

Abstract. A Credit-Assignment CMAC (CA-CMAC) algorithm is proposed to reduce learning interference in conventional CMAC. In the proposed CA-CMAC, the error of the training sample distributed to the addressed memory cell is proportional to the cell's credibility, which is the inverse of the cell's activated times. The learning process of CA-CMAC is analyzed and conventional CMAC is proved to be a special case of CA-CMAC. Furthermore, the convergence properties of CA-CMAC both in batch learning and in incremental learning are investigated; meanwhile, the convergence theorems in the two learning schemes are obtained, respectively. Finally, simulations are carried out to testify the theorems and compare the performance of CA-CMAC with that of CMAC. Simulation results prove that CA-CMAC converges faster than conventional CMAC.

Key words. CMAC, convergence property, credit assignment, learning interference

1. Introduction

Albus [1] first developed Cerebellar Model Articulation Controller (CMAC) neural network as a control method based on the principles of the cerebellum's behavior. The main advantages of CMAC against MLP, RBF, and other neural networks are its local generalization, extremely fast learning speed and easy implementation in software and hardware, so it has been widely used in many fields, especially in control fields [5, 7].

There are two basic learning schemes in CMAC. One is cyclic learning in which all training samples are repeatedly learned in many cycles. Parks [8] investigated five various cyclic algorithms in learning control systems, and recommended a 'maximum error' algorithm. The other is random learning in which the training samples are selected in a random way. Random training requires longer periods to achieve a desired performance level than cyclic learning [3]. Both cyclic learning and random learning have a problem named as learning interference, which means training of subsequent samples will destroy the precision of previous ones. Learning interference is due to local generalization built in CMAC, so it can't be avoided completely. In order to reduce learning interference, David [12] suggested a method termed as neighborhood sequential training. Sayil [9] developed a hybrid maximum error with

neighborhood sequential training algorithm. But in neighborhood sequential training, subsequent training points must lie outside of the neighborhood of the previous ones. It is usually impossible that these two methods are applied practically if the function to be learned is unknown.

To develop efficient CMAC training methods, it is essential to reduce learning interference and take advantage of CMAC's generalization ability. In [11], Su adopted a Credit-Assignment CMAC (CA-CMAC) algorithm to reduce learning interference, but he didn't make any analysis on the convergence of CA-CMAC. In this paper, we propose a novel CA-CMAC algorithm in which the credibility of the memory cell is determined according to its activated times. We further present the credit matrix and the credit correlation matrix. We also prove that the conventional CMAC algorithm is a special case of the proposed CA-CMAC algorithm. Then we investigate the convergence properties and obtain the convergence theorems of CA-CMAC in batch learning and incremental learning, respectively. Finally, simulations are carried out to compare the performance of CA-CMAC with that of conventional CMAC.

2. The CA-CMAC Algorithm

CMAC can be considered as an associative memory network, which performs two subsequent mappings $f : \mathbf{X} \to \mathbf{A}, h : \mathbf{A} \to \mathbf{P}$, where $\mathbf{X}$ is M-dimension input space. $\mathbf{A}$ is a N-dimension association cell vector which contains g nonzero elements. g is the generalization parameter. $\mathbf{P}$ is one-dimension output space. In the first mapping, the point $\mathbf{X}_k$ in the input space is mapped into a binary associative vector $\mathbf{A}_k$ whose elements are defined as (1). In the second mapping, the network output is calculated as the scalar product of $\mathbf{A}_k$ and the weight vector $\mathbf{W}$, as shown in (2). The update rule to the weights is shown in (3). For the simplicity of analysis on the convergence, hash coding isn't considered here.

$$a_{k,j} = \begin{cases} 1 & \text{if the j-th element is activated by the k-th sample} \\ 0 & \text{otherwise} \end{cases} \qquad 1 \leq \mathrm{j} \leq \mathrm{n}, \tag{1}$$

$$Y_{\mathrm{r},k} = \mathbf{A}_k^{\mathrm{T}} \cdot \mathbf{W} = \sum_{j=1}^{N} a_{k,j} w_j, \tag{2}$$

$$w_j(t+1) = w_j(t) + \Delta w_j(t) = w_j(t) + \frac{\beta a_{k,j}}{g}\left(Y_{\mathrm{d},k} - \sum_{j=1}^{N} a_{k,j} w_j(t)\right), \tag{3}$$

where k is the k-th sample, $Y_{\mathrm{r},k}$ the real output at the k-th sample, w_j the j-th element in the weight vector $\mathbf{W}$, t the t-th cycle, β the learning rate, $Y_{\mathrm{d},k}$ is the desired output at the k-th sample.

From (3), it can be seen that all addressed memory cells get equal shares for error correction of the sample $(\mathbf{X}_k, Y_{\mathrm{d},k})$ in the t-th cycle. This will result in that previous learned information is corrupted due to learning interference. In fact, the memory cells activated by $(\mathbf{X}_k, Y_{\mathrm{d},k})$ may have different learning histories, thus have the different credibility. Based on the concept of credit assignment, we assign the credibility to each memory cell, which is the inverse of the cell's activated times. The error distribution is proportional to the credibility. The modified update rule to the weights can be written as

$$w_j(t+1) = w_j(t) + \beta * a_{k,j} * \frac{1/f(j)}{\sum_l (1/f_k(l))} \left(Y_{\mathrm{d},k} - \sum_{j=1}^{N} a_{k,j} w_j \right), \tag{4}$$

where $f(j)$ is the activated times of the j-th memory cell. $f_k(l)$ is the activated times of the memory cell activated by the k-th sample. The more times the memory cell has been activated, the more accurate the stored weight is. The equal share of error correcting as $1/g$ in (3) is replaced by $(1/f(j))/\sum_l(1/f_k(l))$ in (4). With this modification, the error of the training sample can be appropriately distributed into the activated memory cell based on its credibility. It is obvious the modified error distribution is more reasonable.

3. The Convergence Property of CA-CMAC

Wong[13] proposed another description of conventional CMAC algorithm to analyze the learning convergence. In this section, we will describe the learning process of CA-CMAC in the similar way. The convergence properties of CA-CMAC in batch learning and incremental learning are investigated, respectively.

Suppose the training samples are $\mathbf{X}_i$, $Y_{\mathrm{d},i}$ ($i = 1,2,\ldots,n$), n is the number of training samples. Consider the t-th cycle of the k-th training sample, the output error $\delta_k^{(t)}$ is $\delta_k^{(t)} = Y_{d,k} - \mathbf{A}_k^{\mathrm{T}}\mathbf{W}^{(t)}$ Let $g_k = \sum_l(1/f_k(l))$ and define unit correction is $\delta_k^{(t)} g_k$. From (4), correction for the weight in the memory cell activated by the k-th sample is $1/f_k(l)^* \delta_k^{(t)}/g_k$. These corrections may affect the outputs of other training samples. At this time, the output of the i-th training sample becomes $\mathbf{A}_i^{\mathrm{T}}\mathbf{W}^{(t)} + d_{ik} * \delta_k^{(t)}/g_k$, where $d_{ik} = \sum_l(1/f_{ik}(l)), f_{ik}(l)$ is the activated times of the memory cell which is activated by both the i-th sample and the k-th sample, evidently $d_{ik} = d_{ki}, d_{kk} = \sum_l(1/f_k(l)) = g_k$. If the initial weights are set to zeros, after t cycles, the accumulated unit correction of the k-th sample is $\Delta'_k = \sum_t \delta_k^{(t)} g_k$. Then the output of the i-th sample can be expressed as $Y_{r,i} = \sum_{k=1}^{n} d_{ik}\Delta'_k$. When the CA-CMAC learning converges, i.e., $\delta_k^{(t)}$ goes to zero, Δ'_k will converge to a constant. Therefore, the convergence of CA-CMAC algorithm is equivalent to the convergence of Δ'_k's. Let $\mathbf{D} = (d_{ik})(i,k = 1,2,\ldots,n)$, $\Delta' = [\Delta'_1, \Delta'_2, \ldots, \Delta'_n]^{\mathrm{T}}$. Expressed in a matrix form, the learning convergence is equivalent to find the solutions of the linear system

$$\mathbf{D}\Delta' = \mathbf{Y}, \tag{5}$$

where $\mathbf{Y} = [Y_{d,1}, Y_{d,2}, ..., Y_{d,n}]^{\mathrm{T}}$, let $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_n]$ and define a matrix $\mathbf{F}$

$$\mathbf{F} = \begin{bmatrix} 1/f(1) & 0 & 0 & ... & 0 & 0 \\ 0 & 1/f(2) & 0 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & ... & 0 & 1/f(N-2) & 0 & 0 \\ 0 & ... & 0 & 0 & 1/f(N-1) & 0 \\ 0 & ... & 0 & 0 & 0 & 1/f(N) \end{bmatrix}_{N\times N}.$$

The diagonal elements of $\mathbf{F}$ represent the credibility of memory cells, so we call $\mathbf{F}$ the credit matrix. From the definition of $\mathbf{D}$, we can get

$$\mathbf{D} = \mathbf{A}^{\mathrm{T}}\mathbf{F}\mathbf{A}. \tag{6}$$

We call $\mathbf{D}$ the credit correlation matrix. $\mathbf{D}$ has many good properties, which can be described in Theorem 3.1.

THEOREM 3.1. *The credit correlation matrix* $\mathbf{D}$ *has following properties*: (a) $\mathbf{D}$ *is a real symmetric matrix. All of its elements are non-negative integers, and the diagonal elements are* g_k $k = 1, 2, \ldots, n$. (b) $\mathbf{D}$ *is a positive semidefinite matrix.*

Proof. Property (a) can be obtained easily from the definition of $\mathbf{D}$. Property (b) is proved as follows:

$\exists \mathbf{X} \neq 0$, the quadratic form $f(\mathbf{X}) = \mathbf{X}^{\mathrm{T}}\mathbf{D}\mathbf{X} = \mathbf{X}^{\mathrm{T}}(\mathbf{A}^{\mathrm{T}}\mathbf{F}\mathbf{A})\mathbf{X} = (\mathbf{A}\mathbf{X})^{\mathrm{T}}\mathbf{F}(\mathbf{A}\mathbf{X}) \geq \mathbf{0}$, therefore, D is positive semidefinite.Then (5) can be written as

$$\mathbf{A}^{\mathrm{T}}\mathbf{F}\mathbf{A}\Delta' = \mathbf{Y}. \tag{7}$$

If $f(i) = 1 (i = 1, 2, \ldots, N)$, $\mathbf{F}$ will become into the unit matrix and $g_k (k = 1, 2, \ldots, n)$ equals to g. Then (7) becomes

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\Delta' = \mathbf{Y}, \tag{8}$$

where $\Delta = [\Delta_1, \Delta_2, \ldots, \Delta_n]^{\mathrm{T}}, \Delta_k = \sum_t \delta_k^{(t)}/g (k = 1, 2, \ldots, n)$. Equation (8) is no other than the equation of the linear system described by Wong in conventional CMAC algorithm [13]. Therefore, conventional CMAC algorithm is just a special case of CA-CMAC, in which all the memory cells have the same activated times, thus have the same credibility. □

3.1. THE CONVERGENCE OF CA-CMAC IN BATCH LEARNING

In batch learning, the update rule of the i-th sample in the t-th cycle can be written as

$$\Delta_i'^{(t+1)} = \Delta_i'^{(t)} + \beta(Y_{\mathrm{d},i} - \sum_{j=1}^{n} d_{ij}\Delta_j'^{(t)})/g_i, \tag{9}$$

Define a matrix

$$\mathbf{G} = \begin{bmatrix} g_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & g_2 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & g_{n-2} & 0 & 0 \\ 0 & \dots & 0 & 0 & g_{n-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & g_n \end{bmatrix}_{n\times n},$$

Equation (9) can be written as

$$\mathbf{G}\Delta'^{(t+1)} = \mathbf{G}\Delta'^{(t)} + \beta(\mathbf{Y} - \mathbf{D}\Delta'^{(t)}), \tag{10}$$

i.e.,

$$\Delta'^{(t+1)} = (\mathbf{I} - \beta\mathbf{G}^{-1}\mathbf{D})\Delta'^{(t)} + \beta\mathbf{G}^{-1}\mathbf{Y}. \tag{11}$$

Equation (11) is actually equivalent to the iteration of (5). From the convergence theorems of the linear equations and matrix theories [10,14], we have following lemmas.

LEMMA 3.1. *For any initial vector* $\mathbf{X}^{(0)}$ *and* $\mathbf{q}$, *the sufficient and necessary convergent condition of the vector sequence* $\{\mathbf{X}^{(k)}\}$ *generated by iterative scheme* $\mathbf{x}^{(k+1)} = \mathbf{M}\mathbf{x}^{(k)} + \mathbf{q}$ $(k = 0,1,\dots)$ *is that, the spectrum radius of the iterative matrix* $\mathbf{M}$ *is less than* 1, *i.e.*, $\rho(\mathbf{M}), 1$.

LEMMA 3.2. *If* $\mathbf{A}$ *is a Hermite matrix*, $\mathbf{A}$*'s Rayleigh quotient* $\mathrm{R}(\boldsymbol{x})$ *is defined as* $R(\mathbf{x}) = \mathbf{x}^{\mathrm{H}}\mathbf{A}\mathbf{x}/\mathbf{x}^{\mathrm{H}}\mathbf{x}(\mathbf{x} \in \mathbf{C}^n, \mathbf{x} \neq 0)$, *where* $\mathbf{C}^n$ *denotes n-dimension complex vector set. Suppose the maximum and minimum eigenvalues of* $\mathbf{A}$ *are* $\lambda_{\max}$ *and* $\lambda_{\min}$, *then* $\mathrm{R}(\mathbf{x})$ *has the following property*: $\lambda_{\min} \leq R(\mathbf{x}) \leq \lambda_{\max}$.

From Lemmas 3.1 and 3.2, the convergent condition of CA-CMAC in batch learning is obtained.

THEOREM 3.2. *The sufficient and necessary convergent condition of CA-CMAC in batch learning is that, the learning rate* β fulfils $0 < \beta < 2/\lambda_{\max}$, where $\lambda_{\max} > 0$ *is the maximum eigenvalue of the matrix* $\mathbf{G}^{-1}$ $\mathbf{D}$.

Proof. From Lemma 3.1, the sufficient and necessary convergent condition of CA-CMAC in batch learning expressed by (11) is

$$\rho(\mathbf{I} - \beta\mathbf{G}^{-1}\mathbf{D}) < 1. \tag{12}$$

Suppose λ is any one eigenvalue of the iterative matrix $\mathbf{I} - \beta\mathbf{G}^{-1}\mathbf{D}$, and $\boldsymbol{x}$ is its corresponding eigenvector, then

$$(\mathbf{I} - \beta\mathbf{G}^{-1}\mathbf{D})\mathbf{x} = \lambda\mathbf{x}. \tag{13}$$

Let two sides of (13) left-multiply the conjugate transpose vector $\mathbf{x}^{\mathrm{H}}$ of $\mathbf{x}$, then

$$\mathbf{x}^{\mathrm{H}}(\mathbf{I} - \beta\mathbf{G}^{-1}\mathbf{D})\mathbf{x} = \mathbf{x}^{\mathrm{H}}\lambda\mathbf{x}.$$

i.e.,

$$\lambda = 1 - \beta \frac{\mathbf{x}^{\mathrm{H}}\mathbf{G}^{-1}\mathbf{D}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}}. \tag{14}$$

Because $\mathbf{G}^{-1}\mathbf{D}$ is the real symmetric matrix and $x \neq 0, (\mathbf{x}^{\mathrm{H}}\mathbf{G}^{-1}\mathbf{D}\mathbf{x})/\mathbf{x}^{\mathrm{H}}\mathbf{x}$ is the Rayleigh quotient of the matrix $\mathbf{G}^{-1}\mathbf{D}$ and can be marked as $R(\boldsymbol{x})$. $\mathbf{G}$ is the positive diagonal matrix, so $\mathbf{G}^{-1}\mathbf{D}$ is also positive semidefinite. From Lemma 3.2, we know, where $0 \leq \lambda_{\min} \leq R(\mathbf{x}) \leq \lambda_{\max}$, where $\lambda_{\max}$ and $\lambda_{\min}$are the maximum and minimum eigenvalues of the matrix $\mathbf{G}^{-1}\mathbf{D}$. From Theorem 3.1, $\mathbf{G}^{-1}\mathbf{D}$ is impossibly similar to a zero-matrix, so $\lambda_{\max} > 0$. Then (14) becomes $\lambda = 1 - \beta R(\mathbf{x})$ and (12) becomes

$$\max|1 - \beta R(\mathbf{x})| < 1 \quad \Leftrightarrow \quad |1 - \beta R(\mathbf{x})| < 1, \forall R(\mathbf{x}) \quad \Leftrightarrow \quad 0 < \beta < 2/R(\mathbf{x}), \forall R(\mathbf{x}). \tag{15}$$

Equation (15) means that the sufficient and necessary convergent condition of CA-CMAC in batch learning is the learning rate β simultaneously fulfils the requirements of $\forall R(\mathbf{x})$, i.e.,

$$\beta \in \bigcap_{\forall R(x)} \{\beta | 0 < \beta < 2/R(\mathbf{x})\}. \text{ That is } 0 < \beta < 2/\lambda_{\max}.$$

3.2. THE CONVERGENCE OF CA-CMAC IN INCREMENTAL LEARNING

In incremental learning, the update rule of the i-th sample in the t-th cycle is

$$\Delta_i'^{(t+1)} = \Delta_i'^{(t)} + \beta(Y_{\mathrm{d},i} - \sum_{j=1}^{i=1} d_{ij}\Delta_j'^{(t+1)} - \sum_{j=1}^{n} d_{ij}\Delta_j'^{(t)})/g_i. \tag{16}$$

$\mathbf{D}$ can be written as $\mathbf{D} = \mathbf{L} + \mathbf{G} + \mathbf{U}$, where $\mathbf{L}$ and $\mathbf{U}$ are the lower and upper off-diagonal parts of $\mathbf{D}$, respectively. $\mathbf{G}$ is the diagonal part of $\mathbf{D}$ and defined in Section 3.1. Then (16) becomes

$$\mathbf{G}\Delta'^{(t+1)} = \mathbf{G}\Delta'^{(t)} + \beta(\mathbf{Y} - \mathbf{L}\Delta'^{(t+1)} - (\mathbf{G} + \mathbf{U})\Delta'^{(t)}), \tag{17}$$

i.e.,

$$\Delta'^{(t+1)} = (\mathbf{G} + \beta\mathbf{L})^{-1}[(1 - \beta)\mathbf{G} - \beta\mathbf{U}]\Delta'^{(t)} + \beta(\mathbf{G} + \beta\mathbf{L})^{-1}\mathbf{Y}. \tag{18}$$

Equation (18) is no other than the Successive Over Relaxation SOR scheme of the linear system (5) with β being the relaxation factor. From the iteration theories of the linear equations [14], we get Lemmas 3.3 and 3.4.

LEMMA 3.3. *The necessary condition for the convergence of SOR scheme of the linear system is the relaxation factor*ω *fulfils* $o < \omega < 2$.

LEMMA 3.4. *If* $\mathbf{A}$ *is a positive definite matrix, when the relaxation factor* ω *fulfils* $o < \omega < 2$, *SOR scheme of the linear system* $\mathbf{Ax} = \mathbf{b}$ *converges permanently.*

Then, the convergent condition of CA-CMAC in incremental learning is obtained.

THEOREM 3.3. *The necessary convergent condition of* CA-CMAC *in incremental learning is that, the learning rate β fulfils $0 < \beta < 2$. Specially, when* $\mathbf{D}$ *is a positive definite matrix, $0 < \beta < 2$ becomes the sufficient and necessary condition.*

Proof. It can be easily obtained from Lemmas 3.3 and 3.4.

4. Simulation Results

To testify Theorems 3.2 and 3.3, we carried out a simulation using CA-CMAC to approximate the complex one-dimension function

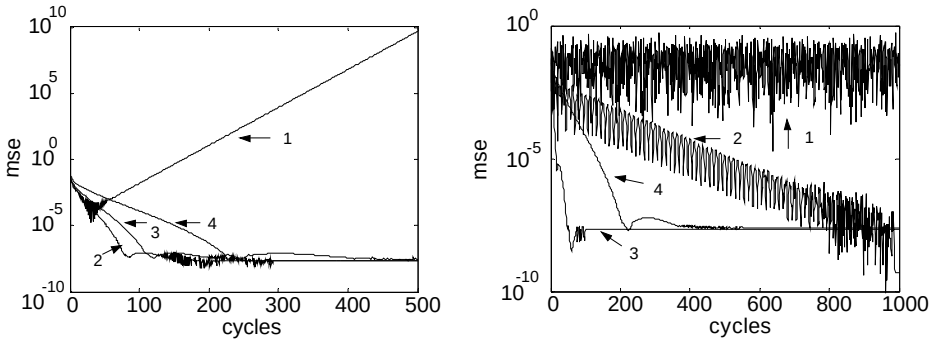
$$f(x) = \sin x + 2\cos(2x) + \mathrm{e}^{-x}, \quad 0 \le x \le 9. \tag{19}$$

Simulation parameters are as follows: the training sample step and the quantizing interval are both 0.5; the generalization parameter g is 8; the permitted error of each training sample is 0.0001; the maximum cycles in batch and incremental learning are 500 and 1000.

It can be calculated that the matrix $\mathbf{G}^{-1}\mathbf{D}$ is positive definite and its maximum eigenvalue is 8 under above simulation parameters. From Theorems 3.2 and 3.3, the ranges of the learning rate for the convergence of CA-CMAC in batch learning and incremental learning are (0, 0.25) and (0, 2), respectively. Figure 1(a) and (b) show the mean square error (mse) of all training samples versus each cycle in the two learning schemes. It can be seen that the CA-CMAC algorithm isn't convergent when the learning rate is beyond the range (0, 0.25) in (a) or (0, 2.0) in (b). Simulation results prove the correctness of Theorems 3.2 and 3.3.

To compare the convergence performance of CA-CMAC with that of conventional CMAC, another simulation is carried out to solve an inverse kinematics problem of 2-DOF planar robot arm. Let (x, y) denote the coordinates of the gripper of the arm. The joint angles are given as (20) in [15].

$$\begin{aligned} \theta_1 &= \arctan(y/x) + \arctan[l_2 \sin\theta_2/(l_1 - l_2\cos\theta_2)], \\ \theta_2 &= \arccos[l_1^2 + l_2^2 - x^2 - y^2)/(2l_1 l_2)], \end{aligned} \tag{20}$$



1: $\beta = 0.25$, 2: $\beta = 0.2$, 3: $\beta = 0.15$, 4: $\beta = 0.1$
(a) batch learning

1: $\beta = 2.0$, 2: $\beta = 1.9$, 3: $\beta = 0.5$, 4: $\beta = 0.1$
(a) incremental learning

Figure 1. The convergence property of CA-CMAC.

where θ_1 and θ_2 are the first and second joint angle, $0 \leq \theta_1, \theta_2 \leq \pi$. Only θ_1 is learned. l_1 and l_2 are the link lengths. Suppose $l_1 = l_2 = 10$ and $0 < x, y < 10$. Simulation parameters are as follows: the intervals of x and y of the training sample are both 1, i.e., $x, y = [0.5, 1.5, \ldots, 9.5]$; the quantizing intervals of x and y are both 0.5; the generalization parameter is 8; the permitted error of each training sample is 0.001; the maximum cycle is 100.

Because a larger fixed learning rate will result in unstable phenomenon while a smaller one will cause slower convergence speed, it has been proved that adjusting learning rate dynamically can improve the convergence performance of CMAC algorithm effectively [4,6]. Figure 2 shows the comparison of two algorithms with the learning rate decreased dynamically in incremental learning. The learning rate is adjusted as follows:

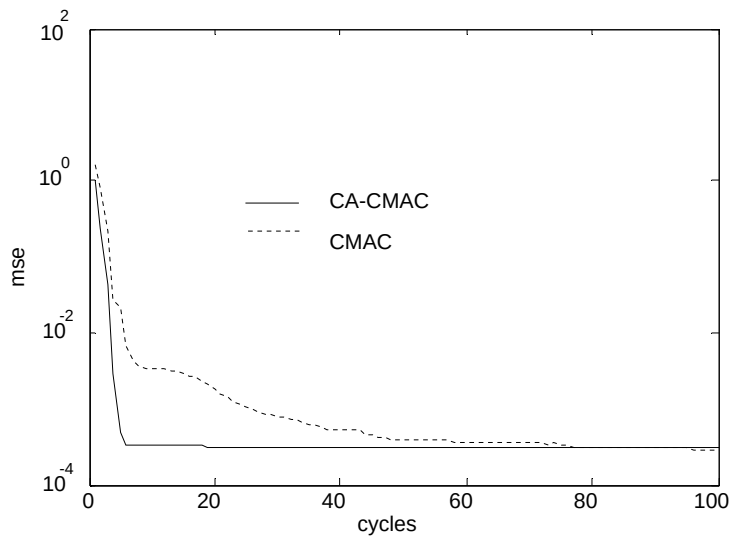


Figure 2. The convergence of two algorithms.

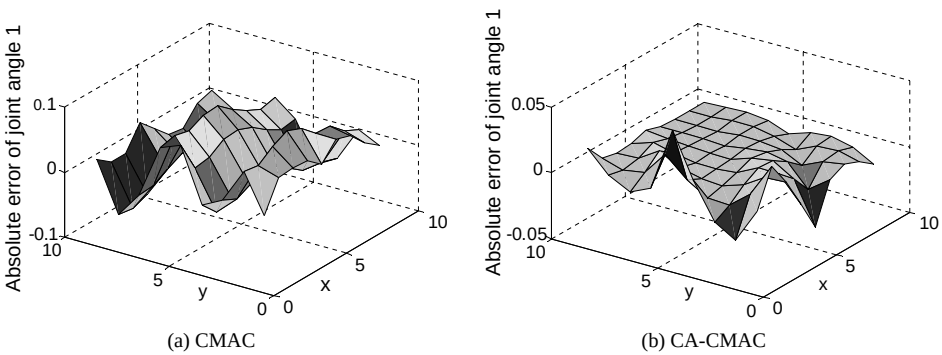


Figure 3. The absolute error of untrained samples.

$$\beta_t = \begin{cases} \beta_{t-1} & \textit{if } \mathrm{mse}_t < \mathrm{mse}_{t-1}, \\ 0.8\beta_{t-1} & \textit{otherwise}, \end{cases} \tag{21}$$

where mse_t means the mean square error of all training samples after the t-th cycle. It can be seen from Figure 2 that CA-CMAC converges faster than conventional CMAC .The mean square error of CA-CMAC is less than 0.001 after only four cycles. This fast convergent speed is especially important for many real-time control applications. To compare the generalization abilities of two algorithms, the absolute errors of untrained samples (i.e., $x, y = [1, 2, \ldots, 9]$) after 10 cycles in two algorithms are shown in Figure 3. It is obvious that the generalization ability to untrained samples of CA-CMAC is also better than that of CMAC.

5. Conclusion

In this paper, a CA-CMAC algorithm is proposed to reduce learning interference in conventional CMAC. The equal distribution of the training sample error into addressed memory cells in CMAC is replaced by proportional distribution based on the credibility in CA-CMAC. Then the learning process of CA-CMAC is analyzed. The convergence properties of CA-CMAC in batch learning and incremental learning are both investigated, meanwhile, the convergence theorems in the two different learning schemes are obtained, respectively, which are significant in acting as the guidance of choosing the learning rate properly in practical applications. Finally, simulation results of function approximating prove the correctness of the convergence theorems. And simulation results of the inverse kinematics of 2-DOF robot arm show CA-CMAC has a faster convergent speed and a better generalization ability than CMAC.

Acknowledgements

This paper is supported by National Natural Science Foundation of China (Grant number 50128504).

References

1. Albus, J. S.: A New approach to manipulator control; The Cerebellar Model Articulation Controller (CMAC), *Journal of Dynamic System, Measurement and Control*, **97**(3) (1975), 220–233.
2. Cembrano, G., Wells, G. and Sarda, J.: Dynamic control of a robot arm using CMAC neural networks, *Control Engineering Practice*, **5** (1999), 485–492.
3. Kwon, S.: An adaptive control system for biological and robotic simulation, Ph. D. dissertation, Department of Mechanical Engineering. Louisiana State University, Dec. 1990.
4. Lin, C. S. and Chiang, C. T.: Learning convergence of CMAC techniques, *IEEE Transaction on Neural Networks*, **8**(6) (1997), 1281–1292.
5. Lin, C. S. and Kim, H.: CMAC-based adaptive critic learning control, *IEEE Transaction Neural Network*, **2**(1991), 530–535.

6. Lu, H. C. and Chang, J.C.: Enhance the performance of CMAC neural network via fuzzy theory and credit apportionment, *Proceedings of the 2002 International Joint Conference on Neural networks*, **1** (2002), 715–720.
7. Miller, W. T., Hewes, P. R. and Glanz, F. H.: Real-time dynamic control of an industrial manipulators using a neural network-based learning controllers, *IEEE Transaction Robot. Automat.*, **6**(1) (1990), 1–9.
8. Parks, P. C. and Militzer, J.: A comparison of five algorithms for the training of CMAC learning control systems, *Automatica*, **28**(5) (1992), 1027–1035.
9. Sayil, S. and Lee, K. Y.: A hybrid maximum error algorithm with neighborhood training for CMAC, *Proceedings of the 2002 International Joint Conference on Neural Networks*, **1** (2002), 165–170.
10. Shi, R. C.: *Matrix Analysis*, Beijing Institute of Technology Publisher: Beijing, China, 1996.
11. Su, S. F., Ted, T. and Hung, T. H.: Credit Assigned CMAC and it's application to online learning robust controllers, *IEEE Transaction on Systems, Man, and Cybernetics-part B: Cybernetics*, **33**(2) (2003), 202–213.
12. Thompson, D. E.: Neighborhood sequential and random training techniques for CAMC, *IEEE Trans Neural Network,* **6**(1) (1995), 196–202.
13. Wong Y. and Sideris, A.: Learning convergence in the cerebellar model articulation controller, *IEEE Transaction on Neural Networks*, **3** (1992), 115–121.
14. Xi, M. C.: *Numerical Analysis Methods*, Chinese Science and Technology Publisher: Hefei, China, 1995.
15. Yao, S. and Zhang, B.: The learning convergence of CMAC in cyclic learning, *Journal of Computer Science and Technology*, **9**(4) (1994), 320–328.