GENERALIZATION OF REINFORCEMENT LEARNING WITH CMAC

Sunggyu Kwon* and Kwang Y. Lee**

** Faculty of Mechanical and Automotive Engineering*
Keimyung University, Daegu, Korea.
(Email: cmack@kmu.ac.kr)
*** Department of Electrical Engineering*
The Pennsylvania State University, University Park, PA 16802, U.S.A.
(Email: kwanglee@psu.edu)

Abstract: To implement a generalization of value functions in Adaptive Search Element (ASE)-reinforcement learning, CMAC is integrated into ASE controller. ASE-reinforcement learning scheme is briefly studied to discuss how CMAC is integrated into ASE controller. Neighbourhood Sequential Training concept is utilized to establish the look-up table of CMAC and to produce discrete control outputs. In computer simulation, an ASE controller and a couple of ASE-CMAC neural network are trained to balance the inverted pendulum on a cart. The number of trials until the controllers are established and the learning performance of the controllers are evaluated to find that generalization ability of the CMAC improves the speed of the ASE-reinforcement learning enough to realize the cartpole control system. *Copyright© 2005 IFAC*

Keywords: CMAC, learning system, reinforcement learning, quantized states, neural networks

1. INTRODUCTION

Reinforcement learning is a method that, through interaction with its environment, learns by receiving feedback in the form of a numerical reward that is commensurate with the appropriateness of the response. In the last fifteen to twenty years, it has attracted an increasing interest in neural networks and intelligent control communities (Kaelbling, et al., 1996) and is considered as an important alternative to conventional methods to intelligent control. Reinforcement learning techniques are good especially when input-output training data are not available. However, reinforcement learning has some drawbacks such as many trials to establish a control strategy, slow convergence, the large state space problem and the large action space problem (Zhao and Liu, 1996).

There has been much effort to improve the performance of the reinforcement learning methods by using various intelligent control techniques. As a major development of reinforcement learning, the problem solving capacities of a reinforcement learning system with two adaptive neural elements, Adaptive Search Element (ASE) and Adaptive Critic Element (ACE), were illustrated (Barto, et al., 1983). A stochastic real-valued reinforcement learning algorithm for learning real-valued control outputs (Gullapalli, et al., 1994) and a stochastic competition learning based on the concept of genetic optimization were introduced to reduce the difficulties of the large action space problem (Zhao and Liu, 1996). An adaptive state space recruitment strategy (Kondo and Ito, 2002), and a reinforcement learning algorithm which allows generalization of learning by using previously learned knowledge (Ricordeau, 2003) are the works to implement generalization capability in reinforcement learning.

In some developments of reinforcement learning, CMACs were used. To improve implementation efficiency and performance, BOXES-ASE/ACE reinforcement learning algorithm (Michie and

Chambers, 1968) was modified with a state history queue and a dynamic link table which implements the CMAC state association (Hu and Fellman, 1994). To produce continuous outputs for controller trained by a stochastic reinforcement learning algorithm, a CMAC-based neural network was used (Han and Zhang, 1994). Both actor and critic were implemented by CMAC in examining the use of prior knowledge in the form of a stable controller that generates control inputs in parallel with a reinforcement learning system (Rosenstein and Barto, 2004).

In particular, there are many studies that use CMACs for generalization of value functions in reinforcement learning (Sutton and Barto, 1998). Most CMAC-based reinforcement learning algorithms use hash coding technique so that similar states in the input space will have similar outputs or value functions. In a self-learning control scheme, Lin and Kim (1991) integrated CMAC into the BOXES-ASE/ACE reinforcement learning algorithm where CMACs were used for storing learning parameters). However, generalization of the ASE weights by CMAC affects the control performance indirectly. To deal with tasks with a continuous state space, Sutton (1996) combined CMAC with a reinforcement learning based on the *sarsa* algorithm. Unfortunately, how the CMACs were combined with the reinforcement learning algorithm and how CMACs were trained was not described. To take the possibility of having a similar output value function for non-adjoining input areas, Xu et al., (2002) modified CMAC structure where the modification requires a priori knowledge of learning control problems. In this case, the CMAC structure should be dependent on the characteristics of the learning system. Wei and Zhao (2003) introduced variable resolution discretization of input space to improve the generalization capability of the CMAC-based reinforcement learning. However, discretizing the input space with variable resolution is another issue to be resolved.

Some previous works are application specific while others require certain conditions for combining CMAC with reinforcement learning methods. However, CMAC itself is a good parametrized function approximator for reinforcement learning without modification of its structure or conditioning of its parameters. Thus, CMAC can be generally combined with any reinforcement learning algorithm only by considering its training.

In this paper, CMAC is integrated into the ASE where the ASE output is used as learning examples for the CMAC that is trained by the Neighbourhood Sequential Training (NST) method (Thompson and Kwon, 1995; Sayil and Lee, 2002). In Sections 2 and 3, the ASE-reinforcement learning scheme and the NST training aspect of CMAC is briefly studied. And the ASE/CMAC neural network is trained to balance the inverted pendulum on a cart in Section 4. The learning performance of the ASE controller and ASE/CMAC neural network was compared to find that generalization ability of CMAC improves the ASE-reinforcement learning in terms of learning speed.

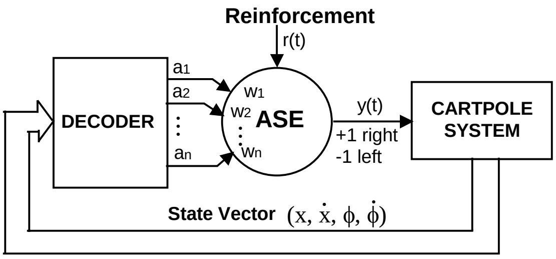

Fig. 1. Scheme of ASE controller (Barto, 1983).

2. ASE-REINFORCEMENT LEARNING

In this section, for the purpose of presenting the idea of integrating CMAC into the ASE controller, ASE reinforcement learning scheme for cartpole control system is briefly discussed.

The ASE controller consists of a decoder and a neuron like adaptive element, ASE (Fig. 1). The decoder, at each time step, receives a state vector giving the state of the cartpole system. It transforms each state vector into a n -component binary vector $\mathbf{A}(t) = (a_1, a_2, \ldots, a_n)$ whose components are all zeros except for a single one a_i in the position corresponding to the pathway that represents the system state to decoder. The value of a_i is 1.0.

For the transformation by the decoder, each state variable should be partitioned into certain number of intervals. Suppose that there are 4, 4, 7, and 4 intervals, respectively, for the four state variables as following:

$$x : [-2.4, -0.96, 0.48, 1.92, 2.4] \,,$$

$$\dot{x} : [-2.0, -0.8, 0.4, 1.6, 2.0] \,,$$

$$\phi : [-12.0, -8.19, -4.41, -0.63, 3.15, 6.93, 10.71, 12.0] \,,$$

$$\dot{\phi} : [-150.0, -60.0, 30.0, 120.0, 150.0]$$

where x (m) and $\dot{x}$ (m/s) are the position and velocity of the cart on the track, and ϕ (deg) and $\dot{\phi}$ (deg/s) the angle and angular velocity of the pole with respect to the vertical axis of the cart.

Then, the intervals are combined to form $448 = 4 * 4 * 7 * 4$ regions. These regions correspond to the output pathways of the decoder. Thus, the decoder has $n = 448$ binary valued output pathways. These pathways are connected to ASE.

The ASE consists of n weights, $w_1, w_2, \cdots, w_n$. The element has n non-reinforcement input pathways from the decoder, a single reinforcement input pathway, and a single output pathway. It must be trained so that its weights are established and its output based on the weights can be used to determine the direction of applying the force of fixed magnitude. However, in the training process, the equations of motion for the cartpole system are not known to ASE, and since the learning is unsupervised, there are no control laws to imitate either.

Here, it is noted that the weights of ASE and the eligibility of the pathways are only related to the n non-reinforcement input pathways, but not directly to the state vectors. Also, through the decoder, some state vectors correspond to a common pathway. Then, it is possible to specify a "special system state" among many states that are decoded into the pathway. Whenever a specific pathway is invoked, it is possible to consider that the system is described by the special system state even though actual system state is one of those states that correspond to the specific pathway. This strategy does not disturb the original ASE-reinforcement learning scheme, including the weights and eligibility update rules. Let's call the special state the *pathway state*. Then, every weight in ASE connected to a pathway of the decoder is related to a pathway state in the continuous state space.

Same reasoning can be applied to each state variable. Since the pathway state is defined by 4 components, those values for the components are the special values for individual state variable. For example, the special values of 4 state variables for the pathway states are as following, where the selection of these values are based on the NST consideration:

x : -2.16, -0.72, 0.72, 2.16
$\dot{x}$: -1.8, -0.6, 0.6, 1.8
ϕ : -11.34, -7.56, -3.78, 0.0, 3.78, 7.56, 11.34
$\dot{\phi}$: -135.0, -45.0, 45.0, 135.0

Let's call these special values the *pathway values* of the variables. Then, the pathway states for the very first and last pathways could be (-2.16, -1.8, -11.34, -135.0) and (2.16, 1.8, 11.34, 135.0), respectively.

Based on the ASE output, a fixed magnitude of force is applied to cart in either direction. And, the operation of cart is evaluated as failure when the cart hits the track limits or the pole falls beyond the specified angles. Then, the evaluation signal is fed to ASE as a reinforcement signal. Refer to the paper by Barto, at al. (1983) for the details about the ASE definition and the ASE-reinforcement learning scheme.

Applying the fixed magnitude of force is a way of producing output by ASE, since the ASE only learns to decide the direction of the force to apply. However, it is known that applying variable force produces generally better control. Therefore, it is worthwhile to consider the capability of control forces between 0 and a fixed magnitude as the ASE output.

The variable control forces can be generated, for example, between two extreme fixed magnitudes of control forces by assuming an unknown approximating function that would relate the states close to pathway states to those variable control forces. The approximating function can be assumed by CMAC since CMAC is very good at generalizing learned information.

3. INTEGRATING CMAC INTO ASE CONTROLLER

The CMAC (Albus, 1975) is described by a series of mappings: *S-to-M*, *M-to-A*, and *A-to-p* mapping, where *S* denotes the continuous state space, *M* the intermediate state space, and *A* the quantized state space. In the space *A*, there are a certain number of look-up tables. The output of the CMAC, *p*, is a scalar value that is a summation of values of one entry from each look-up table.

In the *S-to-M* mapping for balancing the inverted pendulum, each of the 4 state variables is quantized into K intermediate variables. The number K is a parameter that determines the range of generalization in storing the learned information in look-up tables. And each intermediate variable is at different level of quantization. Suppose, $K = 3$. Then, there are 3 intermediate variables at 3 levels of quantization. By using the same partitions of state variables as in the ASE decoder, the first intermediate variable of each state variable is partitioned.

For the NST, every intermediate variable of a state variable should be partitioned into the same number of discrete intervals. This is to prevent some entries in the look-up tables from being uncovered by generalization. For example, when x is quantized into 3 levels, then, at 3 quantization levels, the 3 intermediate state variables for x are partitioned into 4 intervals respectively as following:

The first level: $[-2.4, -0.96, 0.48, 1.92, 2.4]$
Second level: $[-2.4, -1.44, 0.0, 1.44, 2.4]$
Third level: $[-2.4, -1.92, -0.48, 0.96, 2.4]$

Of course, the partition at the first level is the same as the one in the ASE. In particular, the sizes of the 4th interval in the first level, [1.92, 2.4] and the first interval in the 3rd level, [-2.4, -1.92] are much smaller than other intervals. This is due to the consideration of the NST method.

Then, a system state is represented by a set of K intermediate state vectors, one from each quantization level. There are 448 sets of such vectors in the space *M*. Only the first among K element intermediate state vectors is known to the decoder.

In the *M-to-A* mapping, every combination of the intervals of each intermediate variable at the same level should construct a look-up table with 448 entries in a quantized state space, *A*. Then, for $K = 3$, there are 3 look-up tables. Because of the unique coding method of CMAC, a system state in *S* is mapped into 3 entries, one for each look-up table in the space *A*. Components of each intermediate state vector of the set of K intermediate state vectors play a role of key for an identifier of the entry of each look-up table.

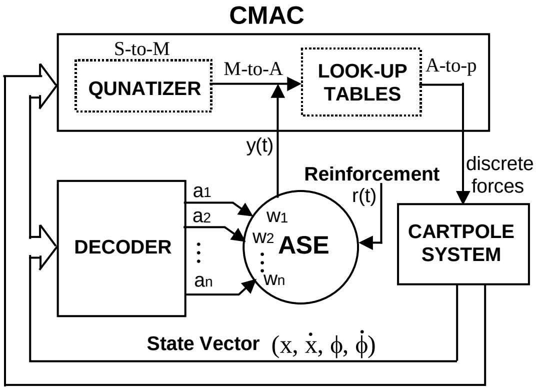

Fig. 2. An ASE/CMAC neural network where a CMAC is integrated into ASE controller. ASE provides the learning information for CMAC.

Establishing the contents of the look-up tables depends on the output of ASE and CMAC training method. However, careless training of CMAC will cause learning interference and result in bad contents in the look-up tables. Hence, the NST method is essential for training the CMAC integrated into the ASE controller.

The CMAC can be integrated into the ASE controller as shown in Fig. 2. The output of ASE, $y(t)$, is used as learning examples to train the CMAC. The output pathways of decoder are represented by the *pathway states*, and the pathway states are defined by the *pathway values* of each variable. The addresses of the pathway states in the look-up tables are fixed, since the pathway values of each variable are prescribed and fixed.

In fact, the pathway states are all mapped into different neighbourhood in the look-up tables of CMAC and they are all strangers to each other. The fixed addresses represent every different neighbourhood. Then, the NST method is for training the CMAC in such a way that training of a neighbourhood does not disturb other neighbourhood. In turn, NST method makes the learning by CMAC for a pathway state free from learning for other pathway states.

Learning the training examples from ASE is only with the pathway states that correspond to $1/K$ of the intermediate states in the space *M*. For other states, no direct learning is available due to the NST method. Then, for $(K-1)/K$ of the intermediate states, direct learning does not occur. However, these intermediate states belong to 2 neighbourhoods in the space *A*. So, the content of look-up tables for those intermediate states in space *A* is established by the generalization of learning due to the mechanism of the hash coding.

In the *A-to-p* mapping, the scalar values of K entries of the K look-up tables are summed up to produce discrete outputs of CMAC, where *p* denotes a scalar value in the action space.

If a system state is mapped into the same region in the space *A* as that is looked upon by the pathway state, the scalar value must be the same as $y(t)$, which was determined by the ASE through reinforcement learning and provided for training of CMAC. For the rest of the system states, the magnitudes of the output values are less than the magnitude of $y(t)$.

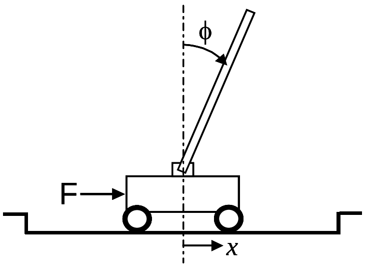

Fig. 3. A cartpole system.

4. EXAMPLE: A CARTPOLE SYSTEM

The cartpole system is a well-known test bed for development of various neural networks, where a pole is hinged to a cart that moves on track to its right or left direction (Fig. 3). The pole rotates about the hinge. The control task is to keep the pole vertically balanced and the cart within the track boundaries.

For the cartpole system, the 4th order Runge-Kutta method with a time step of 0.02 seconds was used to solve the friction dynamics equations with the cartpole parameters in (Barto, et al., 1983). With the solution, the outcome of the ASE controller is evaluated only in terms of the failure or success of the system, where failure is when the pole falls beyond ± 12.0 degree or the cart hits the track boundary at ± 2.4 m. The equations of motion of the cartpole system are not known to the controller, but the state vector describing the system's state at each time step is known. That is, the cartpole system is considered as a black box by the reinforcement learning system. The noise in producing the ASE output was neglected in the learning process.

The CMAC was trained by NST method. Then, the ASE output is distributed while it is stored in the look-up tables. For the output of ASE/CMAC where ASE is with $F = \pm 12.0$ (N), and CMAC with $K = 3$, there are 4 discrete forces such as ± 12.0 (N), ± 8.0 (N), ± 4.0 (N), and 0 (N).

Table 1 shows the numbers of trials until ASE and ASE/CMAC with $K = 3$ succeeded in balancing the pole with 3 consecutive runs for 30 trial cases. For example, for the case No. 1, ASE succeeds in balancing the pole at the 113th trial by applying $F = \pm 10.0$ (N) to the cart at every time step. For Two different systems are tested for the ASE/CMAC , one with $F = \pm 10.0$ (N) and the other with $F = \pm 12.0$ (N). The trial values of x (m) and ϕ (rad) were set at the midpoint of each interval in the partitions of the variables for the decoder. Particularly, all $\dot{x}$ and $\dot{\phi}$ values were set at 1.0 (m/sec) and 0.2 (rad/sec),

Table 1. The Numbers of trials until ASE controller and ASE/CMAC neural networks succeed in balancing the pole with 3 consecutive runs for given trial states.

No	Trial States	ASE	ASE/CMAC, K=3	
			F=10.0	F=12.0
1	(-2.0, 1.0, 0.01, 0.2)	113	1562	18
2	(-2.0, 1.0, 0.08, 0.2)	138	125	66
3	(-2.0, 1.0, 0.15, 0.2)	157	76	22
4	(-2.0, 1.0, -0.04, 0.2)	350	68	266
5	(-2.0, 1.0, -0.11, 0.2)	254	1205	60
6	(-2.0, 1.0, -0.16, 0.2)	332	14	F
7	(-1.2, 1.0, 0.01, 0.2)	299	32	18
8	(-1.2, 1.0, 0.08, 0.2)	355	12	23
9	(-1.2, 1.0, 0.15, 0.2)	419	F	179
10	(-1.2, 1.0, -0.04, 0.2)	532	60	204
11	(-1.2, 1.0, -0.11, 0.2)	352	189	189
12	(-1.2, 1.0, -0.16, 0.2)	66	170	27
13	(0.0, 1.0, 0.01, 0.2)	492	136	262
14	(0.0, 1.0, 0.08, 0.2)	230	562	364
15	(0.0, 1.0, 0.15, 0.2)	187	201	52
16	(0.0, 1.0, -0.04, 0.2)	253	18	F
17	(0.0, 1.0, -0.11, 0.2)	314	52	F
18	(0.0, 1.0, -0.16, 0.2)	454	78	24
19	(1.2, 1.0, 0.01, 0.2)	267	F	13
20	(1.2, 1.0, 0.08, 0.2)	456	F	786
21	(1.2, 1.0, 0.15, 0.2)	525	495	141
22	(1.2, 1.0, -0.04, 0.2)	66	52	F
23	(1.2, 1.0, -0.11, 0.2)	997	177	F
24	(1.2, 1.0, -0.16, 0.2)	1549	18	F
25	(2.0, 1.0, 0.01, 0.2)	F	F	F
26	(2.0, 1.0, 0.08, 0.2)	F	F	F
27	(2.0, 1.0, 0.15, 0.2)	F	F	F
28	(2.0, 1.0, -0.04, 0.2)	F	F	F
29	(2.0, 1.0, -0.11, 0.2)	F	F	F
30	(2.0, 1.0, -0.16, 0.2)	F	F	F

respectively, following the work by Geva and Sitte (1993).

The cases No. 25 through No. 30 are tough trial states where the cart starts near the right boundary of the track with initial velocity of 1.0 (m/sec). In these cases, every controller failed when the cart hits the right boundary of the track just after a few time steps. In Table 1, F implies that the controller could not learn to balance the pole until the 2001st trial.

Observing Table 1, one can see that the ASE controller runs far more number of trials for most of the trial states, although there are some cases for which ASE/CMAC did try more runs. The ASE/CMAC learns faster than the ASE controller for most cases. Moreover, the ASE/CMAC with increased force, $F = \pm 12.0$ (N), succeeded in learning even faster for some cases. It was observed that, for some cases, ASE/CMAC spent more trials when the forces are less than 10 (N). In view of realizing the cartpole system, the ASE controller learns too slowly. The number of trials, for instance, more than 20, is just too much to be realistic.

To see the learning performance, initially the ASE/CMAC with $K = 3$ and $F = \pm 12.0$ (N) was trained for the neutral state (0.0, 0.0, 0.0, 0.0). This is a simple state to realize for experiment and the controller tried 15 times to learn the state. With the learning of the neutral state, the controller was able to balance the pole from the state (-1.5, 0.0, 0.0, 0.0) right away without failure. Then, the controller failed 1 trial before it could learn the state (1.5, 0.0, 0.0, 0.0). It took 9 trials to learn the state (2.0, 0.0, 0.0, 0.0) and 1 trial for the state (-2.0, 0.0, 0.0, 0.0). With those learning experience, the controller could even learn, without failure, the trial case No. 6 for which the controller failed previously. This means that the ASE/CMAC neural network is able to learn fast as well as to generalize learning effectively.

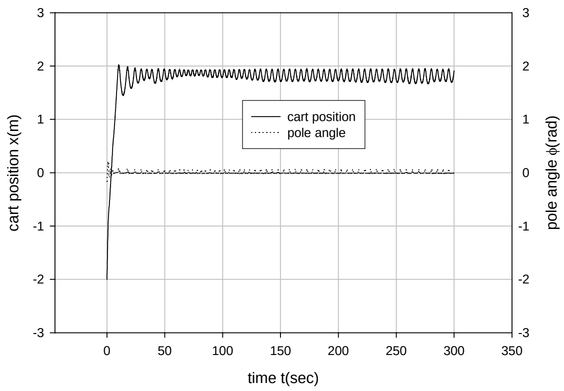

Fig. 4. Cart position and pole angle versus time for the ASE/CMAC with $K = 3$ and $F = \pm 12.0$ (N).

Figure 4 shows the graph of the cart position and pole angle versus time when the ASE/CMAC with $K = 3$ and $F = \pm 12.0$ (N) neural network was tested for the trial case No. 6. Although there are some residual oscillations of the cart and the pole, it is obvious that the controller succeeds in balancing the pole while moving the cart back and forth a little bit near the position $x = 1.8$ (m). Here, it is noted that the system behaviour described by Fig. 4 is not typical for the trial state. The system behaviour by ASE controller is very dependent on other factors, such as the partition of state variables. With training through some other trial states, the ASE/CMAC neural network with $K = 3$ and $F = \pm 12.0$ (N) could balance the pole from most of the cart positions. Thus, the network is a realizable controller.

The ASE/CMAC with $K = 4$ and $F = \pm 12.0$ (N) was also tested. However, learning speed was not as good as that of the ASE/CMAC with $K = 3$ and $F = \pm 12.0$ (N). This suggests that more study is required on the parameters that affect the integration of CMAC into the ASE-reinforcement learning and the training of the network, reminding that bigger K has an influence on the broader range of generalization.

To measure the control quality, the force update interval was increased to 0.03 seconds, in which case no trial states could be learned. As explained by Geva and Sitte (1993), the longer interval is too severe for the ASE controller and even for the ASE/CMAC neural network.

5. CONCLUSION

For a function approximation in the ASE-reinforcement learning, CMAC was integrated into the ASE controller. The CMAC uses the ASE output as learning examples and the output is generalized by Neighbourhood Sequential Training method and stored in the look-up tables. Analysis of the simulation data based on the number of trial times for various trial states leads to conclude that ASE/CMAC neural network learns faster than the ASE and it can be a realistic controller.

A key for integrating CMAC into the ASE controller in the ASE/CMAC neural network is to specify the pathway states for ASE and to use NST method in training the CMAC. In addition, the quantization of continuous state space and the partition of state variables are performed by the consideration of the NST method.

REFERENCES

Albus, J. S. (1975), "Data Storage in the Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control, Transactions ASME, Series G*, Vol. 97, No. 3, September 1975, pp. 228-233.

Barto, A. G., R. S. Sutton, and C. W. Anderson (1983), "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 5, September/October 1983, pp. 834-846.

Fujisawa, S., R. Kurozumi, T. Yamamoto, and Y. Suita (2002), "Path Planning for Mobile Robots using an Improved Reinforcement Learning Scheme," *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*, 27-30 October 2002, pp. 67-74.

Geva, S. and J. Sitte (1993), "A Cartpole Experiment Benchmark for Trainable Controllers," *IEEE Control Systems*, October 1993, pp. 40-51.

Gullapalli, V., J. A. Franklin, and H. Benbrahim (1994), "Acquiring Robot Skills via Reinforcement Learning," *IEEE Control Systems*, February 1994, pp. 13-24.

Han, M. and B. Zhang (1994), "Control of Robotic Manipulators using a CMAC-Based Reinforcement Learning System," *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994*, Volume 3, 2-16, September 1994, pp. 2117-2122.

Hu, Y. and R. Fellman (1994), "A Hardware Efficient Implementation of a Boxes Reinforcement Learning System," *Proceedings of IEEE International Conference on Neural Networks*, Vol. 4, 27 June – 2 July 1994, pp. 2297-2302.

Kaelbling, L. P., M. L. Littman, and A. W. Moore (1996), "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, Vol. 4, 1996, pp. 237-285.

Kondo, T. and K. Ito (2002), "A Reinforcement Learning using Adaptive State Space Construction Strategy for Real Autonomous Mobile Robots," *Proceedings of the 41st SICE Annual Conference*, Vol. 5, 5-7 August, 2002, pp. 3139-3144.

Kretchmar, R. M. and C. W. Anderson (1997), "Comparison of CMACs and Radial Basis Functions for Local Function Approximators in Reinforcement Learning," *Proceedings of International Conference on Neural Networks*, Vol. 2, 9-12 June 1997, pp. 834-837.1997.

Lin, C. and H. Kim (1991), "CMAC-Based Adaptive Critic Self-Learning Control," *IEEE Transactions on Neural Networks*, Vol. 2, No. 5, September 1991, pp. 530-533.

Michie, D. and R. A. Chambers (1968), "'BOXES' as a Model of Pattern-Formation," *Towards a Theoretical Biology*, pp. 206-215.

Ricordeau, M. (2003), "Q-Concept-Learning: Generalization with Concept Lattice Representation in Reinforcement Learning," *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, 3-5 November 2003, pp. 316-323.

Rosenstein, M. T. and A. G. Barto (2004), "Reinforcement Learning with Supervision by a Stable Controller," *Proceedings of the 2004 American Control Conference*, Boston, Massachusetts, June 30 – July 2, 2004, pp. 4517-4522.

Sayil, S. and K. Y. Lee (2002), "A Hybrid Maximum Error Algorithm with Neighborhood Training for CMAC", *Proc. 2002 International Joint Conference on Neural Networks, 2002 World Congress on Computational Intelligence*, Honolulu, Hawaii, May 12-17, pp. 165-170.

Sutton, R. S. (1996), "Generalization in Reinforcement Learning: Successful Examples using Sparse Coarse Coding," *Advances in Neural information Processing Systems* 8, 1996, pp. 1038-1044.

Sutton, R. S. and A. G. Barto (1998), *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, Massachusetts.

Thompson, D. E. and S. Kwon (1995), "Neighborhood Sequential and Random Training Techniques for CMAC," *IEEE Transactions on Neural Networks*, Vol. 6, No. 1, January 1995, pp. 196-202.

Wei, Y. and M. Zhao (2003), "Effective Strategies for Complex Skill Real-time Learning using Reinforcement Learning," *Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, October 2003, pp. 388-392.

Xu, X., D. Hu, and H. He (2002), "Accelerated Reinforcement Learning Control using Modified CMAC Neural Networks," *Proceedings of the 9th International Conference on Neural Information Processing*, Vol. 5, pp. 2575-2578.

Zhao, L. and Z. Liu (1996), "A Genetic Algorithm for Reinforcement Learning," *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 2, 3-6 June 1996, pp 1056-1060.