2003 International Symposium on Advanced Intelligent Systems
September 25-28, 2003, Jeju, Korea

Reinforcement Learning Using a State Partition Method under Real Environment

Ken Saito(1), Shiro Masuda(1), Toru Yamaguchi(1)
(1) Graduate School of Engineering
Tokyo Metropolitan institute of Technology
6-6 Asahigaoka, Hino City, Tokyo 191-0065, Japan
email: smasuda@cc.tmit.ac.jp

Abstract **- This paper considers a reinforcement learning(RL) which deals with real environments. Most reinforcement learning studies have been made by simulations because real-environment learning requires large computational cost and much time. Furthermore, it is more difficult to acquire many rewards efficiently in real environments than in virtual ones. The most important requirement to make real-environment learning successful is the appropriate construction of the state space. In this paper, to begin with, I show the basic overview of the reinforcement learning under real environments. Next, I introduce a state-space construction method under real environments, which is State Partition Method. Finally I apply this method to a robot navigation problem and compare it with conventional methods.**

I Introduction

Reinforcement learning(RL)[1] is a kind of machine learning. Its goal is that an autonomous agent optimizes its behavior by progressively improving its performance based on given rewards from an unknown environment.

When we construct a state space in real environments, it is necessary to adjust the size of the state space. Coarse segmentation will cause so-called "perceptual aliasing problem" by which an agent cannot discriminate states important to accomplish a task. Fine segmentation to avoid the perceptual aliasing problem will produce too many states to manage with computational resources such as CPU time and memory. Such a problem of the trade-off gives us an assignment of constructing the appropriate state space. Therefore, in this research, to resolve this problem, we propose the learning system using State Partition Method based on history information of the states.

Many studies on state partition using the history information[3, 5] have been dealt with. However, if we only part the state based on the history information, the number of states explodes. Therefore, it is not applicable to large state spaces. So, most of the studies on the state partition method deal with, not real environments, but simulations in virtual environments. In this research, using ART neural network, we try to categorize the history information as some categories in order to keep the number of parted states as small as possible.

In Section II, we introduce the mobile robot Khepera used in this research. In Section III, we briefly explain the learning system of the real robot. In Section IV, we explain State Partition Method proposed in this research. In Section V, a result of an experiment is shown in order to confirm the availability of the proposed method. In Section VI, we summarize the result of this research and remark about future tasks.

II Outline of a mobile robot Khepera

The structure of the mobile robot Khepera used in this research is shown in Fig.1.

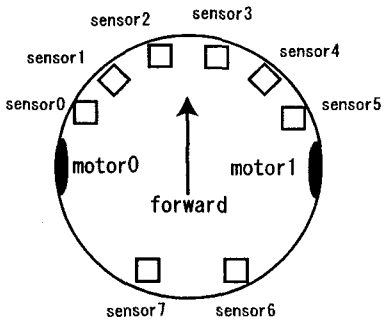

Figure 1. Structure of Khepera

Khepera has two wheels and eight proximity and light sensors around it. Using the proximity sensors, Khepera emits infra-red rays and perceives the response of obstacles. The measurement is shown by an integer from 0 to 1023, and the value changes due to the reflexivity(color, kind of surface...) of and the distance to the obstacle. And the value is much influ-

enced by the ambient light, too. The sensor can perceive obstacles at a distance from 5mm to 30mm. The light sensor perceives the light of the external world. The measurement is shown by an integer from 0 to 512, and the value changes due to the intensity of and the distance to the light source. In this research, we construct a state space using these two sensors.

III Learning system

As a reinforcement learning method to use the experience of getting rewards in later learning, we use On-line Profit Sharing(On-line PS)[6] in this research. However,the conventional PS and the On-line PS have a disadvantage that they can't implement roulette selection, as the learning proceeds in the environment which has negative reinforcement signals(penalties). This is because the weight of rules can become negative values. Therefore, in this research, as used in Q-learning[2], an agent selects an action by Boltzmann distribution of a $w(s_t, a)$ as follows.

$$prob(s_t, a_t) = \frac{exp^{\frac{w(s_t,a_t)}{\tau}}}{\Sigma_a exp^{\frac{w(s_t,a)}{\tau}}} \qquad (1)$$

where $w(s_t, a_t)$ is the weight of the state-action pair at a time step t and τ is a scaling constant. The larger it is, the more random the agent's action becomes. The nearer it is to 0, the more preferably the agent selects the action having the maximum weight.

IV State space construction in real environments

A *General State Space*

In this paper, the sensors used to construct a state space are the proximity sensors from No.0 to No.7 and the light sensors from No.2 to No.3 in Fig.1.

Concretely, the state space used in this research is shown as follows. (Table.1)

In this research, the average value of the sensors in the same category is dispersed. For example, in case of the value of the proximity sensor0 =420 and sensor1=890, the value is categorized as the interval [400,800], for $x(1)$=(420+890)/2=655. And, in case of $\mathbf{y}_t$=[0,2,0,0,1] at a time step t, we define the state $s_t = 0 \times 3^0 + 2 \times 3^1 + 0 \times 3^2 + 0 \times 3^3 + 1 \times 3^4 = 87$. In addition, the number of states in this state space is 162(= $3 \times 3 \times 3 \times 3 \times 2$).

B *State Partition Method using ART neural network*

In the proposed method, state transition records are stored in a state transition table. If a contradiction is found in the table, a state partition is implemented to be consistent with the table. Hereafter, first, we explain the state transition table and the mechanism of contradiction elimination.

Table 1. Construction of state space based on sensory inputs

x	Sensor No.	Interval	y	No.of $y(i)$
$x(1)$	Proximity 0,1	[0,400]	0	3
		[400,800]	$y(1)$=1	
		[800,1023]	2	
$x(2)$	Proximity 2,3	[0,400]	0	3
		[400,800]	$y(2)$=1	
		[800,1023]	2	
$x(3)$	Proximity 4,5	[0,400]	0	3
		[400,800]	$y(3)$=1	
		[800,1023]	2	
$x(4)$	Proximity 6,7	[0,400]	0	3
		[400,800]	$y(4)$=1	
		[800,1023]	2	
$x(5)$	Light2,3	[0,400]	$y(5)$=0	2
		[400,512]	1	

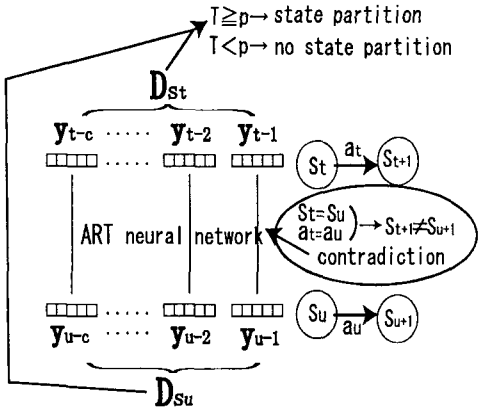

Figure 2. Outline of state partition

How to construct the state transition table

(1) When an input vector $\mathbf{x}_t$ is given, it is classified into a state space based on Table.1. This memory pattern is called a state s_t.
(2) Next, at the state s_t, the agent implements an action a_t and transits to the next state s_{t+1}. Then the state transition record "Action a_t at the state s_t results in the transition to the state s_{t+1}" is acquired. If s_{t+1} is not equal to s_t and the record is not defined in the table, the record (s_t, a_t, s_{t+1}) is stored.

Overcoming contradictions by the mechanism of State Partition

We assume that the state transition record (s_t, a_t,

s_{t+1}) is defined in the table. Furthermore, if another record "Action $a_u(=a_t)$ at the state $s_u(=s_t)$ results in the transition to the state $s_{u+1}(\neq s_{t+1})$" is obtained at time u, the record contradicts the table. This is partly because of the incompleteness of the state partition. In other words, in the predefined state space, it is guessed that confusing of the practically different states makes the contradiction. Therefore, to define state s_u and state s_t as different states, a new state partition is implemented as follows.

As the clue to distinguish between s_t and s_u, the history information of the states is used in this research. The history information of the state s_t and s_u is defined as

$$\mathbf{D}_{s_t} = [\mathbf{y}(t-1), \mathbf{y}(t-2), ..., \mathbf{y}(t-c)]^T \quad (2)$$
$$\mathbf{D}_{s_u} = [\mathbf{y}(u-1), \mathbf{y}(u-2), ..., \mathbf{y}(u-c)]^T$$

respectively. In other words, $\mathbf{D}_{s_t}$ is defined as the interval values of sensory inputs measured at the last c transition states. Then some methods try to part the state whose history information is different from the others, but such a partition results in the explosion of the number of the states. Therefore, in this research, controlling of the state partition is implemented as follows, using the number of transitions to the state and ART.

(1) Control of the State Partition which bears no relevance to rewards

In the exploitation-oriented methods, the reasonable rules which contribute to acquiring rewards are intensively selected and the unreasonable rules are not selected so often. As a result, the number of transiting to the states which are not important to get rewards becomes fewer. Therefore, in this research, we control the state partition, using the number of transitions to the state to prevent profitless state partitions. Concretely, in implementing a state partition, the state partition is cancelled if the number of transitions to the state isn't over N times.

(2) Control of the State Partition by clustering history information using ART

We assume that the condition (1) is satisfied. Then, in this research, ART neural network[4] is used to control the profitless state partitions further, which classifies the history information of states.

Concretely, in case there is a contradiction between state s_t and s_u, classifying $\mathbf{D}_{s_t}$ and $\mathbf{D}_{s_u}$ using ART is implemented. If the Euclidean distance between $\mathbf{D}_{s_t}$ and $\mathbf{D}_{s_u}$ is over or equal to p, in other words, if

$$T_i = \sqrt{(\mathbf{D}_{s_t} - \mathbf{D}_{s_u})^2} \geq p \quad (3)$$

is satisfied, the state partition is implemented. By the state partition, the new memory pattern j' is added into the output layer and the weight of memory pattern $\mathbf{w}_{j'}$ is defined as $\mathbf{w}_{j'}=\mathbf{D}_{s_u}$. However, if not, the partition is not implemented, and the agent defines $s_u=s_t$. Therefore, a smaller vigilance parameter results in fine classification and a larger one results in coarse classification. This is how the states which have similar history information are not parted, so the profitless partitions are controlled.

In addition, after parting the state s_t, at time v, if the agent transits to s_t again, the agent calculates the selection intensity T_i of s_v and the parted states. If the minimum value is over or equal to p, the state s_v is classified into the parted state. Otherwise, the agent keeps learning with $s_v = s_t$.

V Experiment under a real environment using Khepera

A *Experimental environment and method*

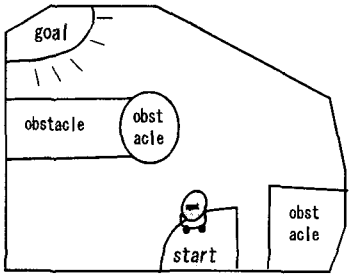

Figure 3. a real environment used for the experiment

In this section, we demonstrate the performance of the State Partition Method and On-line PS described in Section IV and III respectively, using the mobile robot Khepera. The experimental environment is shown in Fig.3 and the agent aims at reaching the GOAL of the light source by implementing an action repeatedly. Each action is outputted every 0.1 sec. The action set **A** is defined by

A=(forward, backward, turn right, turn left).

And the reinforcement signals are defined as follows.

$$R(s_t, a_t) = \begin{cases} 200 & \textit{if the robot reaches the goal area} \\ -5 & \textit{access or crush to the obstacle} \\ -4 & \textit{keep turning at the same place} \\ 0 & \textit{otherwise} \end{cases}$$

Under these conditions, Random Search, On-line PS(without State Partition), and On-line PS(with State Partition) are compared.

The number of experiments is 20, and the average value of total rewards at 5000 steps is used for comparison.

In addition, the other parameters used in this experiment are shown in Table.2

Table 2. parameters for the experiment of navigating mobile robot

parameters	description
γ=10.0	the scale constant in Eq.(1)
p=3	the vigilance parameter in Eq.(3)
$w_0(s_i, a_i) = 50$	the initial value in Eq.(1)
$N = 40$	the minimum No. of transitions to the states to implement state partition
$n = 3$	the No. of the previous states for history information in Eq.(2)

B *Result of the experiment*

Table 3. The result of real-environment experiment

	Total Rewards
(1)Random Search	140
(2)On-line PS(without Partition)	2930
(3)On-line PS(with Partition)	3320

The experimental result is shown in Table.3.

With Random Search, the agent can seldom reach the goal within 5000 steps. This is because the environment has the large state space where it takes at least about 60 steps to reach the goal.

Using On-line PS(Without Partition), the agent learns the policy quickly and often reaches the goal. However, because its state space is rather coarse, the agent sometimes falls into perceptual aliasing problem. In the worst case, the agent runs on the same area far from the goal repeatedly.

On-line PS(With Partition) resolves the perceptual aliasing problem well, but doesn't produce too many states because of the control of state partition using ART and the number of transitions to the state. In addition, the number of parted states is shown in Table.4.

Table 4. Time courses of the number of parted states

No. of steps	No. of parted states
1000	1.8
2000	10.7
3000	14.1
4000	15.2
5000	16.0

As shown in Table.4, the state partition occurs from 1000 to 2000 steps most frequently. The state partition seldom occurs over 3000 steps. The result of preliminary experiments shows that the number of the parted states converges to 18.3 on average.

Of course the number of the parted states are dependent on the vigilance parameter p in Eq.(3). The larger p results in the decrease of the number of the states, and the smaller p results in the increase of the number of the states. And the appropriate state partition greatly depends on the tuning of this parameter p. Currently, using the preliminary experiments, we adopt the desirable value. However, according to the given environments, the framework which enables the agent to tune the parameter p is our future task.

VI Conclusion

In this research, we proposed On-line Profit Sharing with State Partition Method which aims at learning under real environments and explained the outline of the reinforcement-learning system. And we showed the availability of this method, using the experiment in Section.V

For the future, we want to deal with the following studies.

(1) Real-robot learning applicable to multi-agent systems.

(2) Robust reinforcement-learning system which can deal with the drastic change of the environment and disturbances.

(3) Response to the continuous action space

By proceeding these studies, we would like to construct the reinforcement learning system which can deal with more complicated tasks.

References

[1] Sutton, R. S. and Rarto, A., Reinforcement Learning: An Introduction, A Bradford Book, (Cambridge,MA: The MIT Press., 1988)

[2] C.J.C.H. Watkins and P. Dayan, Q-learning, J.Machine Learning, vol.8, pp.279~292(1992)

[3] McCallum, R. A., *Instance-Based Utile Distinctions for Reinforcement Learning with Hidden States*,Proc. of the 12th International Conference on Machine Learning, pp.387–395 (1995)

[4] H. Handa, A. Ninomiya, T. Horiuchi, T. Konishi and M. Baba: "An Incremental State-Segmantation Method for Reinforcement Learning Using ART Neural Network", Proceedings of the 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation, pp.2732-2737, 2000.

[5] Ken Saito, Shiro Masuda. On a Reinforcement Learning Based on Profit Sharing under Multi-Rewards Environment, Proceedings of Electronics, Information and Systems Conference Society, I.E.E of Japan. Electronics, Information and Systems, OS6-5, 2002

[6] Tohgoroh Matsui, Nobuhiro Inuzuka, Hirohisa Seki. On-line Implementation of Profit Sharing Based On Eligibility Traces. Proceedings of FIT, pp.269~270 (2002)