Natural Gradient Actor-Critic Algorithms using Random Rectangular Coarse Coding

Hajime Kimura[1]

[1]Department of Marine Engineering, Graduate School of Engineering, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, 819-0395 Japan
(http://sysplan.nams.kyushu-u.ac.jp/gen/index.html)

Abstract: Learning performance of natural gradient actor-critic algorithms is outstanding especially in high-dimensional spaces than conventional actor-critic algorithms. However, representation issues of stochastic policies or value functions are remaining because the actor-critic approaches need to design it carefully. The author has proposed *random rectangular coarse coding*, that is very simple and suited for approximating Q-values in high-dimensional state-action space. This paper shows a quantitative analysis of the random coarse coding comparing with regular-grid approaches, and presents a new approach that combines the natural gradient actor-critic with the random rectangular coarse coding.

Keywords: Reinforcement learning, actor-critic, Q-learning, continuous state-action spaces, Function approximation

1. INTRODUCTION

Reinforcement learning (RL) is a promising paradigm for robots to acquire control rules automatically in unknown environments. The most popular RL algorithm is tabular Q-learning [7] because of its simplicity and well-developed theory. However, RL in robotics is to deal with high-dimensional continuous state-action spaces, that immediately raises *the curse of dimensionality*, in which costs increase exponentially with the number of the state-action variables. It includes two issues in RL: One is a generalization problem for Q-function over high-dimensional continuous spaces, and the other is a sampling problem that is to select actions following a given distribution over high-dimensional continuous action spaces. To deal with these issues, a novel technique of Q-learning is proposed that combines *random rectangular coarse coding* with an action selection scheme using *Gibbs-sampling*[4]. The random rectangular coarse coding is very simple and quite suited both to approximate Q-functions in high-dimensional spaces and to execute Gibbs sampling. Gibbs sampling enables us to execute action selection following Boltsmann distribution over high-dimensional action space. The Q-learning algorithm can approximate optimum state-action value function independent of the policy being followed, however, *Boltzmann exploration strategy* does not always improve the agent's behavior efficiently.

In high-dimensimonal space, applying gradient methods to improve the agent's policy is one promising solution. *Natural-gradient actor-critic* algorithms are elegant policy gradient methods that are making use of first and second order partial derivatives, therefore its performance is outstanding especially in high-dimensional spaces than conventional actor-critic algorithms[2][5]. However, representation issues of stochastic policies or value functions are remaining because the actor-critic approaches need to design it carefully. This paper presents a new approach that combines the natural gradient actor-critic with the random rectangular coarse coding and Gibbs-sampling. It discharges us from burdensome designing policy functions for each task.

2. PROBLEM FORMULATION

Let $\mathcal{S}$ denote state space, $\mathcal{A}$ be action space, $\mathcal{R}$ be a set of real number. At each discrete time t, the agent observes state $s_t \in \mathcal{S}$, selects action $a_t \in \mathcal{A}$, and then receives an instantaneous reward $r_t \in \mathcal{R}$ resulting from state transition in the environment. In Markov decision processes (MDPs), the reward and the next state may be random, but their probability distributions are assumed to depend only on s_t and a_t. And the next state s_{t+1} is chosen according to the transition probability $T(s_t, a, s_{t+1})$, and the reward r_t is given randomly following the expectation $r(s_t, a)$. The agent does not know $T(s_t, a, s_{t+1})$ and $r(s_t, a)$ ahead of time. The objective of RL is to construct a policy that maximizes the agent's performance. Consider cumulative discounted reward as a performance measure for infinite horizon tasks:

$$V_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \, , \tag{1}$$

where the discount factor, $0 \leq \gamma \leq 1$ specifies the importance of future rewards, and V_t is the value at time t. In MDPs, the value can be defined as:

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \, r_t \middle| \, s_0 = s, \pi \right] \, , \tag{2}$$

where $E\{\cdot\}$ denotes the expectation. The learning objective in MDPs is to find optimum policies that maximize the value of each state s defined by Eq.(2). The state $\mathcal{S}$ and the action $\mathcal{A}$ are both defined as multi-dimensional continuous spaces in this paper.

3. RANDOM RECTANGULAR COARSE CODING FOR HIGH-DIMENSIONAL SPACES

Value function approximation can be significantly enhanced through the use of feature extraction, a process

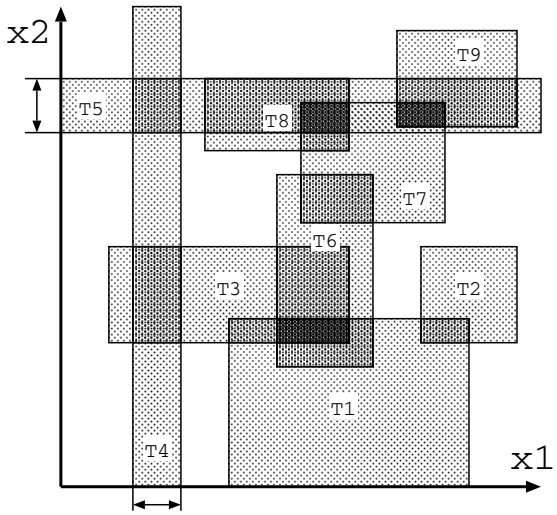

Fig. 1 An example of the random rectangular coarse coding using 9 features in two-dimensional input space. The coarse feature T4 is sensitive to the component x1, and it is defined by the arrow width in the sub-space x1. But the T4 ignores the component x2, then the rectangle covers the other all sub-space x2. The feature T5 is defined similarly.

that maps the input space x into some vector $F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_n(\boldsymbol{x}))$, that is refered to *feature vector* associated with $\boldsymbol{x}$. Feature vectors summarize what are considered to be important characteristics of the input space. One of the most important cases of the value function approximation is that in which the approximate function is a linear function of the parameter vector $W = (w_1, w_2, \cdots, w_n)$ that has the same number of the components as the feature vector $F(\boldsymbol{x})$. For example, the state value is approximated by

$$\hat{V}(s) = \frac{1}{||F(\boldsymbol{x})||} \sum_{i=1}^{n} w_i f_i(\boldsymbol{x}), \tag{3}$$

This value function is said to be *linear*. When the set of the feature vectors $\{F(x)|x \in \mathcal{S}\}$ is *linearly independent*, then it is known that the predictions of the TD-methods converge in expected value to the ideal predictions [6].

To generate feature vectors that are linearly independent in high-dimensional spaces, random rectangular coarse coding is introduced. To explain the concept of the random rectangular coding, consider a randomized rectangular feature $f(x)$ that is an element of the feature vector. When an input vector x is inside of the rectangle, then the corresponding feature has the value 1, otherwise 0. In our approach, the feature rectangles are overlapped in many places. This kind of approach is known as *coarse coding* [6]. In higher dimensional space, a feature $f(x)$ selects a few arbitrary sensitive components of the input vector x, and composes a hyper rectangle in the sensitive subspace. That is, a feature $f(x)$ ignores non-sensitive components of the input vector x. For example in Fig.1, the input vector $\boldsymbol{x}$ is two-dimensional (x_1, x_2), and the rectangular T1, T2, $\cdots$ T9 are associated with components of the feature vector respectively. Notice that T1, T2, T3, T6, T7, T8 and T9 are sensitive to the components both x_1 and x_2, however, T4 is sensitive only to x_1 and T5 is sensitive only to x_2.

For practical use, the feature $f_i(\boldsymbol{x})$ is defined by a rectangular area in a subspace of the input vector. The subspace is composed of arbitrary plural components of the input vector, named *sensitive elements*. When the location of the input vector $\boldsymbol{x}$ is the inside of the rectangle, the value of the feature is given by $f_i(\boldsymbol{x}) = 1$, otherwise $f_i(\boldsymbol{x}) = 0$. The selecting the sensitive elements or the size of the rectangle are given randomly, but it has sound statistics, i.e., the expected activating probability is 10 % when the input vector $\boldsymbol{x}$ is thrown by uniform distribution. Aggregating a number of such features, it is possible to give a set of feature vectors that is linearly independent so that the state-action values can be adequately approximated over essential states (or actions) in real-world tasks, without exponential growth of the memory requirements.

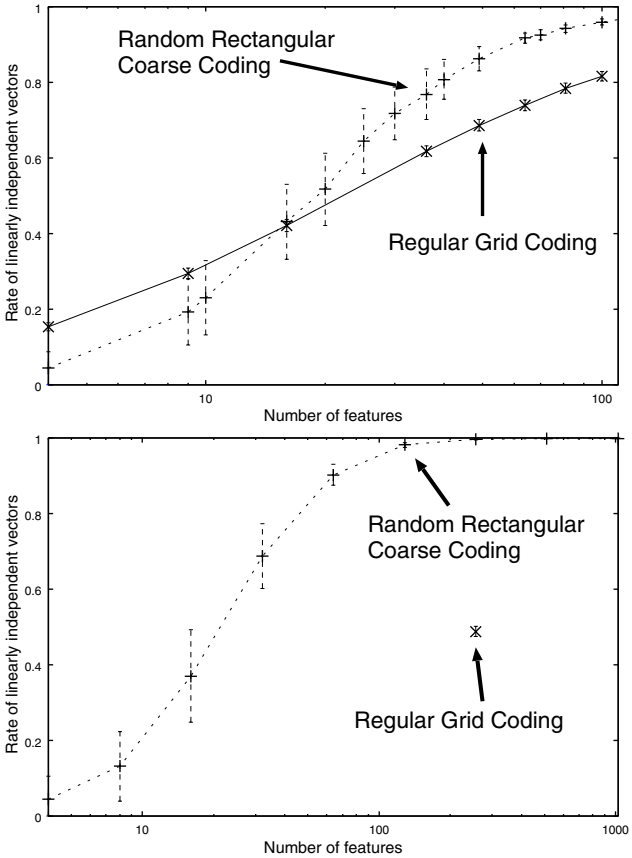

Fig. 2 Linearly independence rates of the feature vectors varying the number of the features (i.e., hyper rectangles) in multi-dimensional input spaces. The upper is 2-dimensional, the lower is 8-dimensional space.

The random rectangular coarse coding seems ad hoc, but its performance is nothing to regular grid approaches' in the condition of the same number of features. Figure 2 shows linearly independence rates of the feature vectors varying the number of the features in 2 or 8 dimensional

1. Observe state s_t, choose action a_t with probability $\pi(a_t|s_t, \boldsymbol{\theta})$ that is a behavior policy, and perform it. Observe immediate reward r_t and next state s_{t+1}.
2. Calculate the eligibility $\frac{\partial \ln \pi(a_t|s_t,\boldsymbol{\theta})}{\partial \theta}$ with respect to policy parameters $\boldsymbol{\theta}$. Using these eligibilities and corresponding weight parameters, find the estimated advantage function:
$$\hat{A}^{\pi}(s_t, a_t) = \left(\nabla_\theta \ln \pi(a_t|s_t, \boldsymbol{\theta})\right)^T \boldsymbol{w}$$
3. Calculate the TD_error and update the estimated state value in the critic according to
$$\begin{aligned} \delta_t &= r_t + \gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t) \,, \\ \hat{V}^{\pi}(s_t) &\leftarrow \hat{V}^{\pi}(s_t) + \alpha\delta_t \,, \end{aligned}$$
where α is a learning rate, δ_t is temporal difference (TD).
4. Update weight parameters $\boldsymbol{w}$ for the advantage function using δ_t and the eligibility:
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \left(\delta_t - \hat{A}^{\pi}(s_t, a_t)\right) \nabla_\theta \ln \pi(a_t|s_t, \boldsymbol{\theta})$$
5. Update policy parameters $\boldsymbol{\theta}$ using the advantage function parameter $\boldsymbol{w}$:
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_p \boldsymbol{w} \,,$$
6. Let $t \leftarrow t + 1$, and go to step 1.

Fig. 3 A natural gradient Actor-Critic algorithm

input space. The rate is higher, it grows the higher probability of distinction between two inputs nearby, where one input is given by uniform distribution provided in $[0, 1]$ for each axis, the other input is given by the normal distribution of which the center is the former input and the deviation is 0.1. Reinforcement learning in robotics often encounters such a difficult situation that the agent must distinguish near two points in the input space. The reason of the outperformance of the random coarse coding is that the regular grid coding generates unnecessarily restricted feature vectors so that only one of the component is 1 and the others are 0.

4. LEARNING ALGORITHM

4.1 A Natural-Gradient Actor-Critic

Natural-Gradient Actor-Critic Algorithms (Fig.3) are elegant policy gradient methods that are making use of first and second order partial derivatives, therefore its performance is outstanding especially in high-dimensional spaces. The advantage function is given by substructing the state value from the state-action value. Note that the basis function for approximating the advantage function is given by the eligibility of the policy parameters. In this paper, the action-selection probability $\pi(a_i|s, \boldsymbol{\theta})$ is given by Boltzmann distribution that contains associated $\Theta(s, a_i, \boldsymbol{\theta})$ so that

$$\pi(a_i|s, \boldsymbol{\theta}) = \frac{\exp\left(\Theta(s, a_i, \boldsymbol{\theta})\right)}{\sum_{j=1}^{D^N} \exp\left(\Theta(s, a_j, \boldsymbol{\theta})\right)} \,, \tag{4}$$

where N denotes the number of regular quantization for each axis of the action space, and D is the number of the dimension of the action space. The function $\Theta(s, a, \boldsymbol{\theta})$ is given by

$$\Theta(s, a, \boldsymbol{\theta}) = \frac{1}{||F(\boldsymbol{s}, \boldsymbol{a})||} \sum_{k=1}^{n} f_k(\boldsymbol{s}, \boldsymbol{a})\, \theta_k \,, \tag{5}$$

where $F(\boldsymbol{s}, \boldsymbol{a})$ is a feature vector $F(\boldsymbol{s}, \boldsymbol{a}) = (f_1(\boldsymbol{s}, \boldsymbol{a}), f_2(\boldsymbol{s}, \boldsymbol{a}), \cdots, f_n(\boldsymbol{s}, \boldsymbol{a}))$, that is the rectangular coarse coding defined over the state-action space (s, a). A policy parameter θ_k is associated with the feature $f_k(\boldsymbol{s}, \boldsymbol{a})$. From partially differentiating Eq.(4),

$$\frac{\partial \ln \pi(a_i|s_t, \boldsymbol{\theta})}{\partial \Theta(s_t, a_k, \boldsymbol{\theta})} = \begin{cases} 1 - \frac{\exp\left(\Theta(s_t, a_i, \boldsymbol{\theta})\right)}{\sum_{j=1}^{D^N} \exp\left(\Theta(s_t, a_j, \boldsymbol{\theta})\right)} & , k = i \\ -\frac{1}{\sum_{j=1}^{D^N} \exp\left(\Theta(s_t, a_j, \boldsymbol{\theta})\right)} & \text{otherwise.} \end{cases}$$

Since the denominator in Eq.(4) grows too large in high-dimensional action space, the eligibility in Fig.3 can be approximated as :

$$\frac{\partial \ln \pi(a_i|s_t, \boldsymbol{\theta})}{\partial \theta_k} \simeq \frac{f_k(\boldsymbol{s_t}, \boldsymbol{a_i})}{||F(\boldsymbol{s_t}, \boldsymbol{a_i})||}$$

As a result, the algorithm becomes very simple form.

The feature vector over state-action space $F(\boldsymbol{s}, \boldsymbol{a})$ is composed of state features and action features as below. State features $f_j(s)$ where $j \in \{1, 2, \ldots T_s\}$ that are defined in state space and action-features $g_k(a)$ where $k \in \{1, 2, \ldots T_a\}$ that are defined in action space are introduced. It is desirable that L_1 norm of the feature vectors are constant in linear function approximation. Therefore, state-feature vectors $F(s)$ that has $2T_s$ components $F(s) = (F_1(s), F_2(s), \cdots F_{2T_s}(s))$ and action-feature vectors $G(a)$ that has $2T_a$ components $G(a) = (G_1(a), G_2(a), \cdots G_{2T_a}(a))$ are given by the state-features $f_j(s)$ and action-features $g_k(a)$ respectively according to

$$F_j(s) = \begin{cases} f_j(s) & \text{, where } j \le T_s \\ 1 - f_{(j-T_s)}(s) & \text{otherwise.} \end{cases} \tag{6}$$

$$G_k(a) = \begin{cases} g_k(a) & \text{, where } k \le T_a \\ 1 - g_{(k-T_a)}(a) & \text{otherwise.} \end{cases} \tag{7}$$

In this paper, the state-action feature $F(\boldsymbol{s}, \boldsymbol{a}) = (f_1(\boldsymbol{s}, \boldsymbol{a}), \cdots, f_n(\boldsymbol{s}, \boldsymbol{a}))$ is given by the product of the state feature $F_j(s)$ and the action feature $G_k(a)$. Then, the function $\Theta(s, a, \boldsymbol{\theta})$ in Eq.5 is given by

$$\Theta(s, a, \boldsymbol{\theta}) = \frac{1}{T_s\, T_a} \sum_{j=1}^{2T_s} \sum_{k=1}^{2T_a} F_j(s)\, G_k(a)\, \theta_{jk} \quad , \tag{8}$$

where θ_{jk} are policy parameters of which the number is $2T_s \times 2T_a$, and $\frac{1}{T_s\, T_a}$ is a normalization factor so that the L_1 norm of the feature vectors composed of $F_j(s)$ and $G_k(a)$ keep 1 constant. The state value $\hat{V}^{\pi}(s)$ in Fig.3 is represented using only the state feature $F_j(s)$ as Eq.3.

4.2 Action Selection Scheme in High-Dimensional Action Space

As shown in Eq.(4), the naive action-selection scheme needs to calculate all discrete action values, that gives rise to exponentially increase with the number of the action dimension. However, an action-selection method for high-dimensional action space using *Gibbs sampling* is a powerful technique for generating samples from multi-dimensional complex distribution [4].

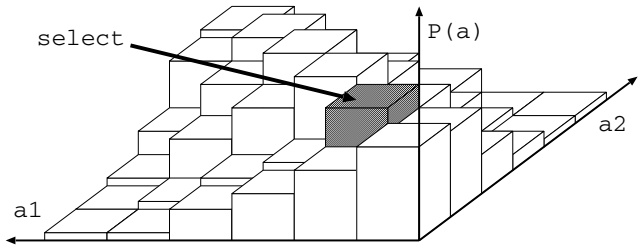

Fig. 4 An action-selection with a regular grid approach in two-dimensional action space. The probability function of the action selection $P(a)$ is represented by quantizing the action space $a1 - a2$ into a finite number of cells.

In this paper, regular grid partitioning is used that divide D equivalent parts into each dimension of the action-space. That is, when the action-space is N-dimensional, the partitioning yields D^N cells. Each cell is considered to be a discrete action a_i where $i = \{1, \cdots, D^N\}$. For each decision epoch, an action a_i is selected following an associated probability function $P(a_i)$. Fig.4 shows the regular grid partitions and its distribution function in two-dimensional action-space.

In the Gibbs sampler, the components of the multidimensional random variable are resampled in turn, conditional on the others, using some fixed ordering. Fig.5 illustrates the Gibbs sampling for a case with two variables $(a1, a2)$. The distribution for action selection probability is equal to Fig.4. In the initial step of Gibbs sampling, an arbitrary sample $(a1, a2)$ is selected in advance. In each iteration, the top right (1) in Fig.5 represents selecting action $a1$ following the conditional probability given $a2$, and the bottom left (2) in Fig.5 represents selecting action $a2$ following the conditional probability given $a1$ at the top right (1) in Fig.5, then the iteration is completed. The bottom right (3) in Fig.5 represents the next iteration, selecting action $a1$ again, following the conditional probability given $a2$ at the bottom left (2) in Fig.5. After a sufficient number of iterations, we can take $a1$ and $a2$ as the sample from the distribution. The resulting sample can be considered to be independent of the sample in the initial step. It is a difficult problem to predict the lower bounds of the number of the iteration, therefore we should choose it experimentally. Gibbs sampling is particularly well-adapted when the joint distribution is not known explicitly, but the conditional distribution of each variable is known. In high-dimensions, getting joint distribution costs too expensive computational resources. For example, if a regular grid partition divides each dimension of the 8-dimensional action space into 10 parts, then finding the joint distribution for the flat action selection costs memory capacity and computation over 10^8 variables, whereas the Gibbs sampling, that uses conditional distribution, costs $10 + 8$ variables and the number of iteration can be reduced to 10^2 or 10^3. That is, Gibbs sampling cuts down the computational cost over $1/10000$.

Throughout this paper, probability of the action selection is given by Eq.(4), but we use particular notation for Gibbs sampling. Suppose that a regular grid partition divides D parts into each dimension of the N-dimensional action space, same as the case of the flat action selection. We give a N-dimensional action a as $a = (a^1, a^2, \ldots a^N)$, where each component a^n $(n \in \{1, 2, \cdots N\})$ is $a^n \in a^n_d$ $(d \in \{1, 2, \cdots D\})$. A value of $\Theta(s, a, \boldsymbol{\theta})$ in Eq.(4) is given by $\Theta(s, a, \boldsymbol{\theta}) = \Theta(a^1, a^2, \ldots a^N | s)$. This value is equal to the flat action selection method's. Sampled components of the action vector at the iteration t in the Gibbs sampling is given by $a^1(t), a^2(t), \ldots a^N(t)$. Then, the Gibbs sampler updates components of the action vector at the iteration $t + 1$ following the conditional distribution as

$$\begin{aligned} a^1(t+1) &\sim P(a^1 | a^2(t), a^3(t), \ldots a^N(t)) , \\ a^2(t+1) &\sim P(a^2 | a^1(t), a^3(t), \ldots a^N(t)) , \\ &\vdots \\ a^N(t+1) &\sim P(a^N | a^1(t), a^2(t), \ldots a^{N-1}(t)) , \end{aligned}$$

where the conditional probability $P(a^n_i | a^1, a^2, \ldots a^N)$ is given by the following Boltzmann distribution:

$$P(a^n_i | a^1, a^2, \ldots a^N) = \frac{\exp\left(\Theta(a^1, a^2, \ldots a^n_i, \ldots a^N | s)\right)}{\sum_{d=1}^{D} \exp\left(\Theta(a^1_d, a^2_d, \ldots a^n_d, \ldots a^N_d | s)\right)} . \quad (9)$$

It is simply repetition of similar processes to the flat action selection's shown in Eq.(4), however, getting off with one dimensional action selection. The obvious advantage is that the Gibbs sampler can get equivalent[1] samples following Eq.(4) without checking out all $\Theta(s, a_i, \boldsymbol{\theta})$ over the high-dimensional action-space. Note that since the calculation of the conditional probability in each iteration is along an arbitrary axis of the action space, the computational cost can be reduced in the proposed rectangular coarse coding.

5. EXPERIMENTS

This section evaluates the proposed method empirically comparing with Q-learning.

5.1 Crawling Robots

Fig.6 shows a several-legged crawling robot in which both the state and the action are high-dimensional. In the robot, each leg has two joints, and it is arranged in parallel. Therefore the dimension of state-action spaces can be easily varied according to adding or removing the

[1] It is supposed that the number of the iteration is enough. Since predicting the lower bounds of this number is a difficult problem, the samples would be likely to approximate Eq.(4).

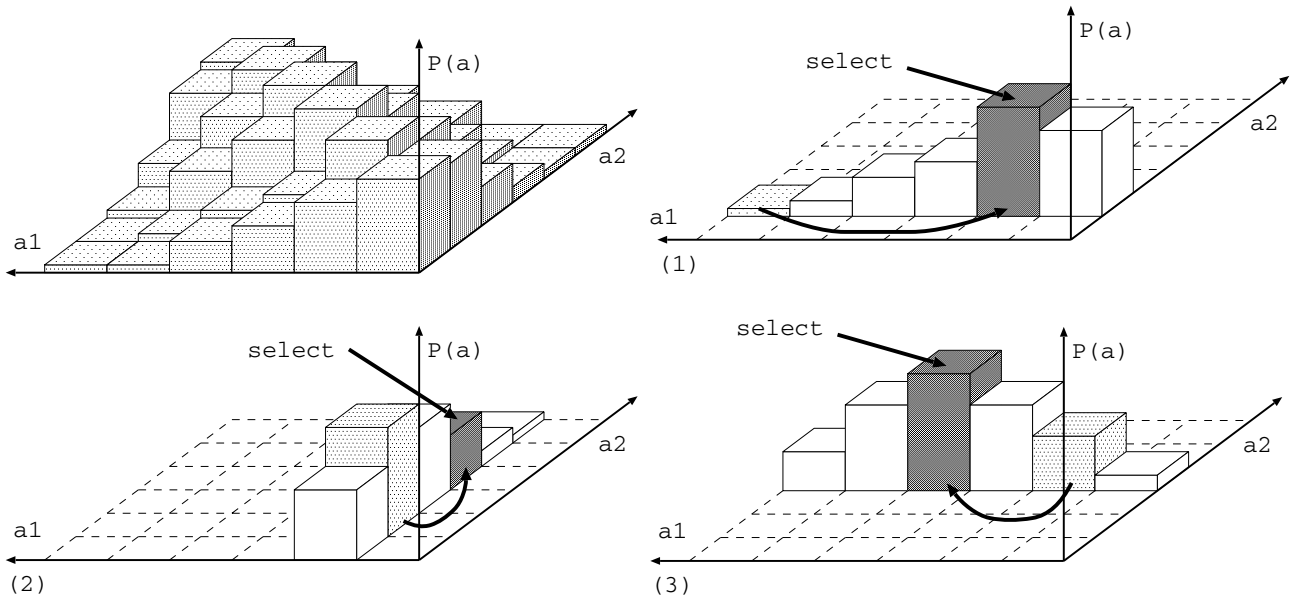

Fig. 5 An example of the Gibbs sampling scheme with the regular grid approach in two-dimensional action space. The top left shows a joint distribution of the action selection probability. The top right (1) represents selecting action $a1$ following the conditional probability given $a2$. The bottom left (2) represents selecting action $a2$ following the conditional probability given $a1$ at the top right (1). The bottom right (3) represents selecting action $a1$ again, following the conditional probability given $a2$ at the bottom left (2).

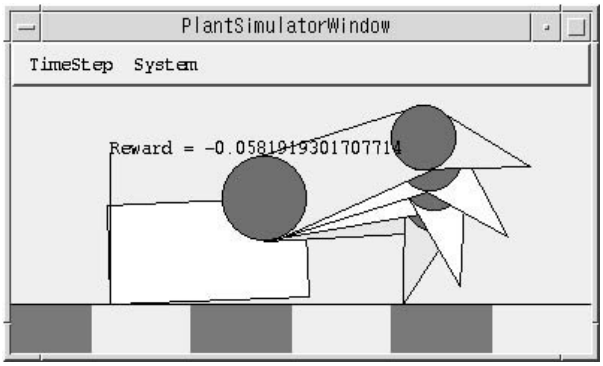

Fig. 6 A simulator of a several-legged crawling robot.

Table 1 Relation between the number of legs and dimensions of state or action space

Legs	State dimension	Action dimension
1	3	2
2	6	4
3	9	6
4	12	8

legs. Note that although the state and action space grows largely, the complexity of the problem is not so changed because the number of solutions also increases. The main issue of this simulation is to find better policies in high-dimensional action space.

In this simulation, the objective of learning is to find control rules to move forward, but the agent does not know the dynamics ahead of time. The controller improves its behavior through a process of trial and error. At each time step, the agent observes current state, and select action, and immediate reward is given as a result of the action and state transition, and the time step proceeds to the next step. The reward signal reflects the speed of the body at each step. These robots have bounded continuous and discrete state variables. Continuous state variables are angular-position of the joints, and discrete state variables represent touch sensors for each leg. The learning agent observes these state variables. The action that the agent selects is an objective angular-position of the joint-motors. That is, the number of dimension of the action space is the same as the continuous state's. When the agent selects action, the robot moves the motors towards the commanded positions. When the joint-angles move to the commanded position, or changing the variables of touch sensors in the way of moving, then the movement stops and reward is given as the result of the transition, and the time step proceeds to the next step. When the case of sensor variable changing in the way of moving joint-motors, the angular-position would not correspond to the selected objective position. The body moves forward or backward when some legs are touching the ground and moving it. The learning controller should find good assignment of crawling control rules for each leg. For simplicity, it is assumed that there is no noise in the state observation. Note that there are many control rules that can move the body ahead.

Table 1 shows the relation between the number of legs and dimensions of state or action spaces. When the robot has only one leg, then the state space is 3-dimensional where one is the touch sensor of the leg top, the others are angular positions of the joints, and the action is 2-dimensional where the destination of the angular po-

Table 2 Relation between the number of legs and the number of discrete states or actions by regular partition

Legs	Number of states	Number of actions
1	2×100^2	100^2
2	$2^2 \times 100^4$	100^4
3	$2^3 \times 100^6$	100^6
4	$2^4 \times 10^8$	10^8

sitions of the 2-joints. A regular grid partition divides each dimension of the state or action space into 10 or 100 parts. Table 2 shows the number of discrete states or actions by regular grid partitioning. The edges of the random hyper-rectangles for the features are generated along the dividing lines of the regular grid. In this experiment, all the agents use 200 randomized rectangular state features where its sensitive elements and the size of the rectangle are randomly given so that the expected activating probability is 10 % under the uniform distribution. The features for the action space are also generated 200 randomized hyper-rectangles in the same way as the state's. The number of iteration in the Gibbs sampling is $(\text{numOfActionDimension}) \times 5$, the discount factor $\gamma = 0.9$, and the learning rate for the critic $\alpha = 0.2$, and the learning parameter for the actor $\alpha_p = 0.001$, 0.0005 or 0.00025. Before learning, all the state values $\hat{V}(s)$ are initialized to 20.

The proposed method is compared with a Q-learning approach using the same rectangular coarse coding and Gibbs sampling [4][5]. In the Q-learning, the action-selection probability $P(a_i)$ is given by Boltzmann distribution that contains associated $Q(s, a_i)$ so that

$$P(a_i) = \frac{\exp\left(\frac{Q(s,a_i) - offset(s)}{T \times width(s)}\right)}{\sum_{j=1}^{D^N} \exp\left(\frac{Q(s,a_j) - offset(s)}{T \times width(s)}\right)} . \qquad (10)$$

The discount factor $\gamma = 0.9$, the temperature T keeps 0.4 constant, and the learning rate $\alpha = 0.2$, $offset(s)$ and $width(s)$ in Eq.(10) are given by the average and deviation of Q-values in state s over 20 actions that are sampled from uniform distribution. The number of iteration in the Gibbs sampling is the same as the proposed method's. The features of the state and action are also the same.

Fig.7, 8, 9 and 10 compare the performance of the natural gradient actor-critic (NGAC) against Q-learning averaging over 10 runs. When the state and action spaces are small as Fig.7, the Q-learning outperforms than the proposed method. However, even though the learning parameters are all the same, the performance of the proposed NGAC is improved in high-dimensional state-action spaces. The learning parameter for the actor should be small for the learning stability in all cases. The deviaton of the performance of the NGAC is larger than the Q-learning's. The performance of the Q-learning is not so different between small and large state-action spaces. In both algorithms, the learned behavior seems

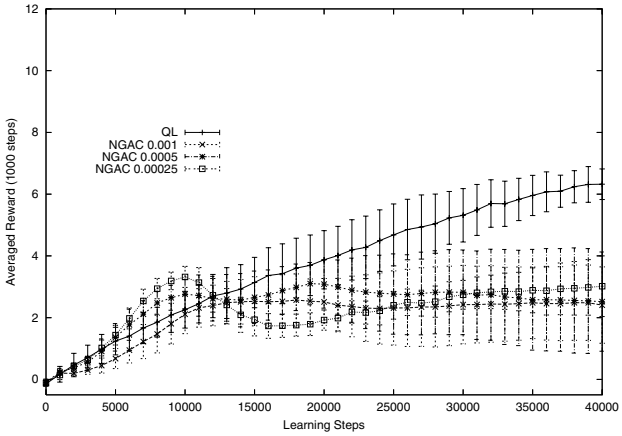

Fig. 7 Learning results averaged over 10 runs in the crawling robot where state is 3-dimensional and action is 2-dimensional (one leg). The vertical axis gives the averaged reward over 1000 decision steps. The errorbars are standard deviation.

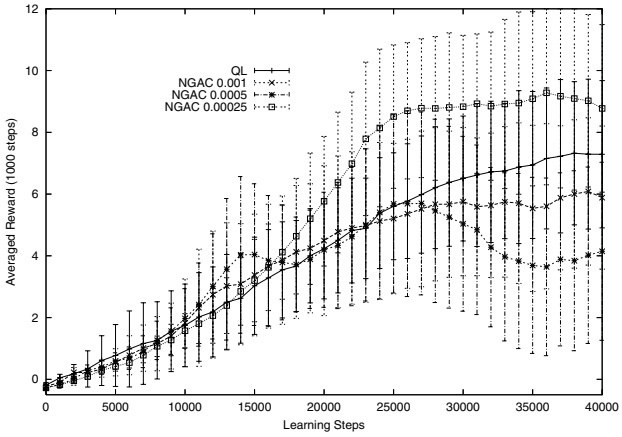

Fig. 8 Learning results averaged over 10 runs in the crawling robot where state is 6-dimensional and action is 4-dimensional (two legs). The vertical axis gives the averaged reward over 1000 decision steps. The errorbars are standard deviation.

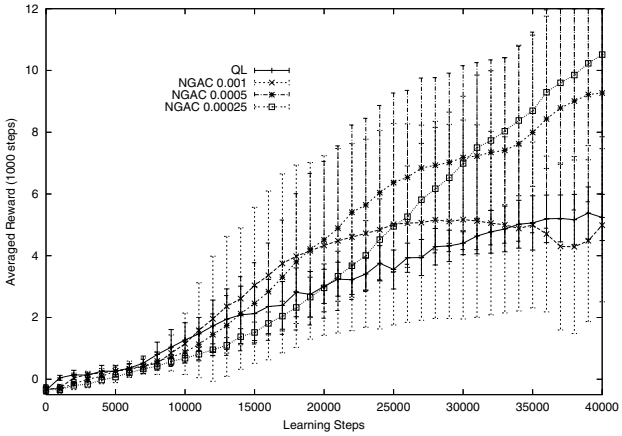

Fig. 9 Learning results averaged over 10 runs in the crawling robot where state is 9-dimensional and action is 6-dimensional (3 legs). The vertical axis gives the averaged reward over 1000 decision steps. The errorbars are standard deviation.

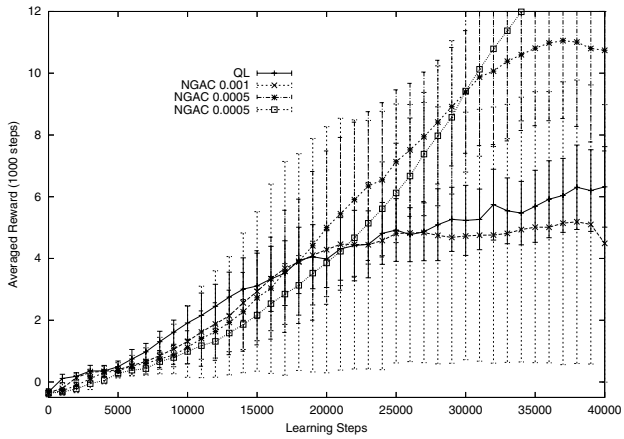

Fig. 10 Learning results averaged over 10 runs in the crawling robot where state is 12-dimensional and action is 8-dimensional (4 legs). The vertical axis gives the averaged reward over 1000 decision steps. The errorbars are standard deviation.

similar. In the case of one-leg, the robot crawls intermittently only when the leg scratches the ground . In the case that the number of legs is 2, 3 or 4, the robots move two-legs alternately so that the body always moves to the front without stopping.

5.2 Reaching Tasks using Multi-Link Redundant-Arms

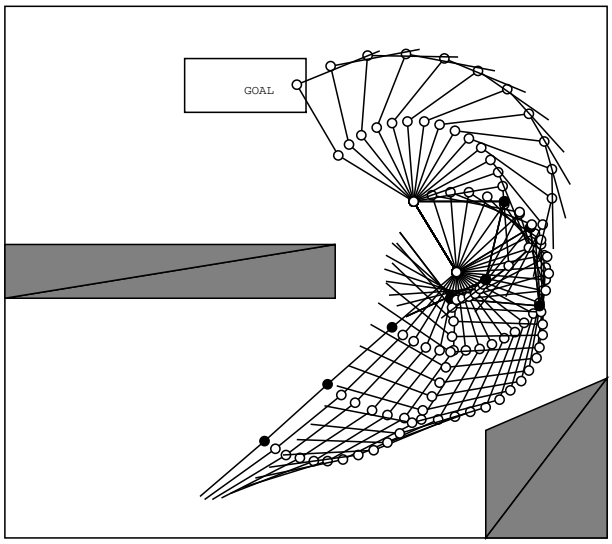

Fig. 11 A learned example behavior of a reaching task using redundant-arm that have four-joints. The conditions that the joints are shown by black circles are the positions at which the learner is making decisions.

The second example is multi-joint arm reaching tasks [4][5]. Fig.11 shows a sample behavior of the task with a redundant arm that is composed of four links and four joints in series where the state space and action space are both 4-dimensional. Fig.12 shows a sample behavior in 8-joints where the state space and action space are both 8-dimensional. In both tasks, the best solutions are composed of only three decision steps: 1) curling the arm, 2) rotating, 3) extending the arm, and thereafter reaching the goal. Also many best solutions exist in these tasks. In the 4-link-arm, the agents divide each dimension of the state and action spaces into 100 parts by regular grids, and in the 8-link-arm the agents divide each dimension into 10 parts. All the agents use 200 randomized rectangular state features and 200 randomized rectangular action features same as the crawling robots'.

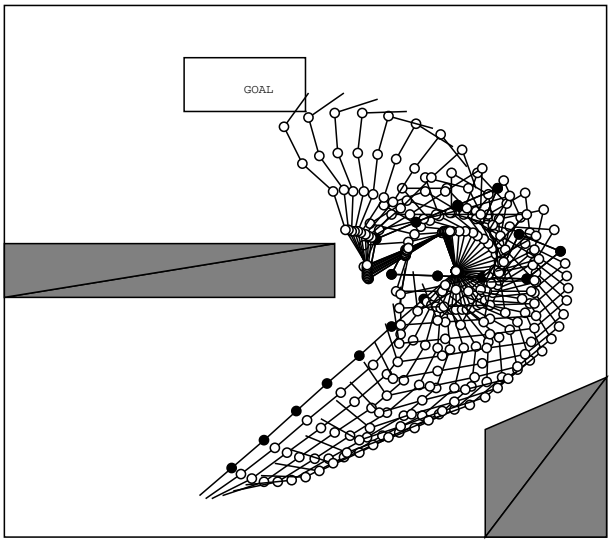

Fig. 12 A learned example behavior of a reaching task using redundant-arm that have eight-joints. The conditions that the joints are shown by black circles are the positions at which the learner is making decisions.

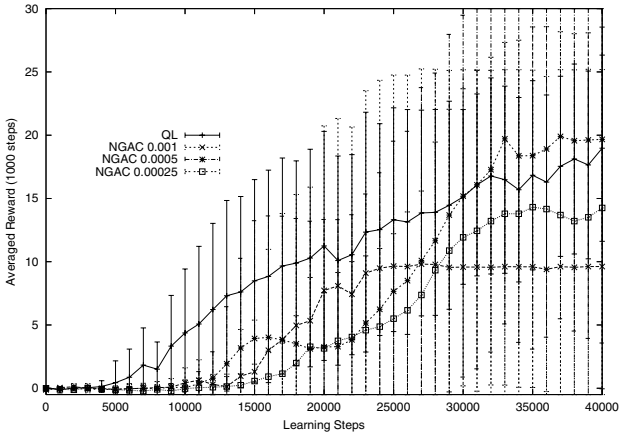

Fig. 13 Learning results averaged over 10 trials in the 4-joints redundant-arm reaching task. The vertical axis gives the averaged reward over making 1000 decision steps.

Fig.13 and 14 compare the performance of the proposed NGAC against Q-learning averaging over 10 runs. When the state and action spaces are small as Fig.13, NGAC is comparable to Q-learning. However, even though the learning parameters are the same, the NGAC is fairly good in high-dimensional state-action space as Fig.14. The deviaton of the performance of the NGAC is larger than the Q-learning's in all the cases.

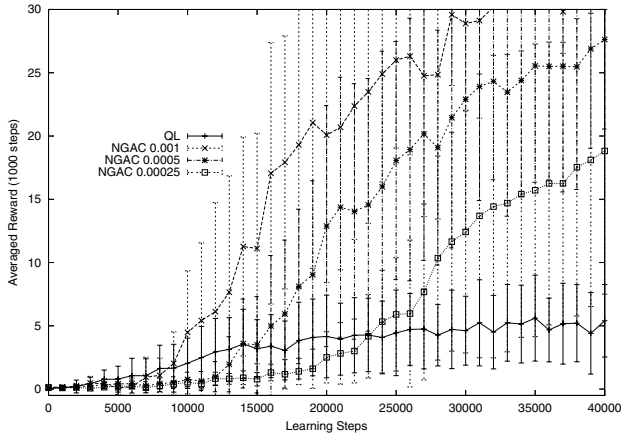

Fig. 14 Learning results averaged over 10 trials in the 8-joints redundant-arm reaching task. The vertical axis gives the averaged reward over making 1000 decision steps.

6. DISCUSSION

6.1 Relation between increase of action dimension and the learning performance

When the dimension of the state-action space is getting large, the performance of the proposed NGAC shows a marked rising tendency whereas the performance of the Q-learning is not so different or becoming worse. This result supports the efficiency of the natural-gradient method in high-dimensional space.

6.2 Setting of learning parameter for policy update

In actor-critic algorithms, since the policy updating must be done sufficiently slowly than the estimating the state-action value, we need to select adequate value of the learning parameter for the actor α_p carefully. For this reason, α_p is a critical parameter under the current circumstances. The appropriate range of the α_p would depend on a maxmum absolute value of the advantage function and representation of the policy function. Finding that range is a future work.

6.3 Finding an appropriate number of the iteration in Gibbs sampling

In the experiments, the numbers of the iteration are given ad hoc. In general, it is difficult to predict the lower bounds of the number of the iteration in Gibbs sampling. It is noteworthy that the Gibbs sampling affects only the time for action selection, it does not affect the learning steps. Therefore, the number of the iteration should be given as long as possible. To get good samples with less time, overrelaxation techniques or simulated annealing [3] are possible.

6.4 Applying eligibility traces to the NGAC

Eligibility traces enhance the robustness of the learning algorithms against not only the non-Markovian effects but incomplete state-action value approximation [5]. Combining it with the proposed approach is very easy.

7. CONCLUSION

This paper proposed a novel actor-critic approach combining random rectangular coarse coding with Gibbs-sampling action selection to cope with reinforcement learning in high-dimensional domains. The proposed coding method is very simple and quite suited both to represent action-selection probability function in high-dimensional spaces, and to execute Gibbs sampling for action selection. The rectangular features are given randomly, however, it works well in several robot problems. The scheme using Gibbs-sampling enables to handle high-dimensional action space in RL problems without burdensome tuning of several learning parameters for varying dimensions. The natural-gradient based actor-critic method is quite efficient in high-dimensional spaces.

REFERENCES

[1] Bertsekas, D.P. & Tsitsiklis, J.N.: Neuro-Dynamic Programming, Athena Scientific (1996).

[2] Bhatnagar, S., Sutton, R., Ghavamzadeh, M. & Lee, M.: Incremental Natural Actor-Critic Algorithms, the 21st Annual Conference on Neural Information Processing Systems (NIPS), 2007.

[3] Jordan, M. I.: Learning in Graphical Models, The MIT Press, (1999).

[4] Kimura, H.: Reinforcement Learning in Multi-Dimensional State-Action Space Using Random Rectangular Coarse Coding and Gibbs Sampling, Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS07), pp.88–95, 2007.

[5] Kimura, H.: Natural Gradient Actor-Critic using Eligibility Traces, Conference Proc. of the 19th SICE Symposium on Decentralized Autonomous Systems, pp.67–72 (2007 in Japanese).

[6] Sutton, R.S. & Barto, A.: Reinforcement learning: An introduction, A Bradford Book, The MIT Press (1998).

[7] Watkins, C.J.C.H. & Dayan, P.: Technical Note: *Q*-Learning, Machine Learning, Vol.8, pp.279–292 (1992).