Constrained Reinforcement Learning from Intrinsic and Extrinsic Rewards

Chapter · January 2009

Source: InTech

CITATIONS
8

READS
241

2 authors:

Eiji Uchibe
Advanced Telecommunications Research Institute
105 PUBLICATIONS **1,215** CITATIONS

SEE PROFILE

Kenji Doya
Okinawa Institute of Science and Technology
370 PUBLICATIONS **13,639** CITATIONS

SEE PROFILE

11

Constrained Reinforcement Learning from Intrinsic and Extrinsic Rewards

Eiji Uchibe and Kenji Doya
Okinawa Institute of Science and Technology
Japan

1. Introduction

The main objective of the learning agent is usually determined by experimenters. In the case of reinforcement learning (Sutton & Barto, 1998), it is defined as maximization of a scalar reward function which should be designed for each task through a trial-and-error process. It is still important to implement learning algorithms that can efficiently improve the learning capabilities, but the principles for designing the appropriate reward functions become important more and more in the future. Reward functions are categorized into two types: extrinsic and intrinsic rewards. In many cases, extrinsic rewards are zero everywhere except for a few important points that correspond to the important events. Although designing such a sparse reward function is easier than designing a dense one, the sparse rewards prevent the learning agent to learn efficiently. On the contrary, the intrinsic reward is regarded as dense reward functions which give non-zero rewards most of the time because it is usually computed from the agent's internal information such as sensory inputs. Although the intrinsic reward is generally task-independent, it plays an important role for designing an open-ended system.

Recently, learning algorithms with intrinsic rewards have been studied by several researchers. Barto and his colleagues (Barto et al., 2004; Singh et al., 2005; Stout et al., 2005) proposed an algorithm for intrinsically motivated reinforcement learning based on the theory of options (Sutton, et al., 1999). Meeden *et al*. realized that a simulated robot tracked a moving decoy robot with the rewards based on the error of its own prediction (Meeden et al., 2004). Oudeyer and his collegues adopted progress of prediction learning as intrinsic rewards and showed that behavior evolution of the Sony's four-legged robot, AIBO, were realized by step-by-step learning (Oudeyer & Kaplan, 2004; Oudeyer et al., 2007). However, most previous studies did not discuss the negative effects of extrinsic rewards on intrinsically motivated learning suggested by (Deci and Flaste, 1996). It is still unclear how extrinsic rewards can help or hinder the learning process.

As the first step towards this problem, this chapter deals with the interaction between intrinsic and extrinsic rewards from a viewpoint of constrained optimization problems. The learning agent tries to maximize the long-term average intrinsic reward under the inequality constraints given by extrinsic rewards. We propose a new framework termed the *Constrained Policy Gradient Reinforcement Learning* (CPGRL) consisting of a Policy Gradient Reinforcement Learning (PGRL) algorithm (Baxter & Bartlett, 2001; Konda & Tsitsiklis, 2003;

Source: Theory and Novel Applications of Machine Learning, Book edited by: Meng Joo Er and Yi Zhou,
ISBN 978-3-902613-55-4, pp. 376, February 2009, I-Tech, Vienna, Austria

Morimura et al., 2005) and a gradient projection method (Rosen, 1960). Since The PGRL algorithms can estimate the gradients of the expected average rewards with respect to the policy parameters, they are nicely integrated with the gradient projection method. Although constrained Markov Decision Process (MDP) problems are previously studied based on linear programming techniques (Feinberg & Shwartz,1999; Dolgov & Durfee, 2005), their methods do not suit our case because the state transition probabilities are known, and because it cannot be easily extended to continuous state and action spaces. In order to evaluate the CPGRL we conduct two simulations: a simple MDP problem with three states and a control task of a robotic arm.

2. Constrained policy gradient reinforcement learning

2.1 Formulation

At each time step, an agent observes a state $\boldsymbol{x} \in \boldsymbol{X}$ and executes an action $\boldsymbol{u} \in \boldsymbol{U}$ with probability $\mu_\theta(\boldsymbol{x},\boldsymbol{u})$: $X \times U \to [0, 1]$ that represents a stochastic policy parameterized by an n-dimensional vector $\boldsymbol{\theta} \in R^n$. The agent calculates an intrinsic reward r_t^1 and extrinsic rewards r_t^2 (i=2, 3,..., m) at time t, which depend on the state and the action. Let $r_t^i = r^i(\boldsymbol{x}_t, \boldsymbol{u}_t)$ and $\boldsymbol{r}_t = [r_t^1\ r_t^2\ \ldots\ r_t^m]^T$ denote respectively the immediate reward at time t and the vectorized representation. The operation a^T means the transpose of vector/matrix a.

The objective for the agent is to find the policy parameter $\boldsymbol{\theta}$ that maximizes an average reward

$$g^1(\boldsymbol{\theta}) = \lim_{T\to\infty} \mathrm{E}_{\boldsymbol{\theta}} \left[\frac{1}{T} \sum_{t=1}^{T} r_t^1 \right] \tag{1}$$

under the constraints determined by the extrinsic rewards given by

$$g^i(\boldsymbol{\theta}) = \lim_{T\to\infty} \mathrm{E}_{\boldsymbol{\theta}} \left[\frac{1}{T} \sum_{t=1}^{T} r_t^i \right] \geq G^i, \quad i = 2, \ldots, m, \tag{2}$$

where G^i is a threshold for controlling a level of the constraint. It is noted that the inequality constraints on extrinsic rewards are also the functions of the average rewards.

Fig.1 illustrates the CPGRL system based on the actor-critic architecture (Sutton & Barto, 1998). It consists of one actor, multiple critics, and a gradient projection module that computes a projection onto a feasible region, which is the set of points satisfying all the inequality constraints. Based on the immediate reward r^i, each critic produces an estimate of the long-term average reward ρ^i and its gradient $\boldsymbol{\Delta}^i$ with respect to the policy parameters. Actor selects the action $\boldsymbol{u}$ according to the stochastic policy $\mu_\theta(\boldsymbol{x},\boldsymbol{u})$. The procedure of the CPGRL is listed below:

while $k < N_K$

1. Set $z_0 = \boldsymbol{0}$ and $\boldsymbol{\Delta}^i = \boldsymbol{0}$ for all i.
2. **while** $t < N_T$
   i. Observe $\boldsymbol{x}_t$ and execute $\boldsymbol{u}_t$.
   ii. Receive the rewards $\boldsymbol{r}_t$.
   iii. Estimate the average rewards and their gradients.
3. Store the estimated average rewards.
4. Update the policy parameter,

where N_K and N_T denote the number of episode and the maximum time step, respectively.

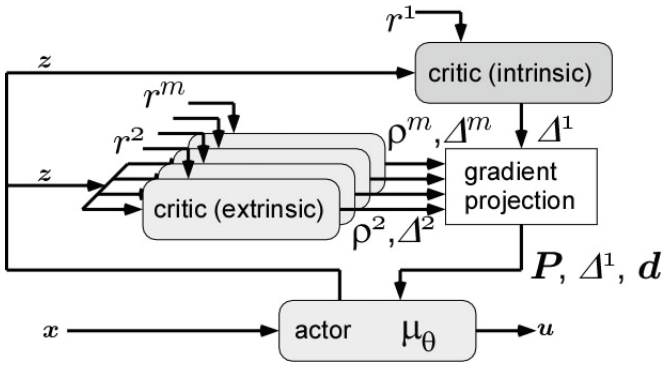

Fig. 1. Block diagram of the actor-critic architecture for learning behaviours from intrinsic and extrinsic rewards.

2.2 Gradient estimates by policy gradient reinforcement learning

The PGRL algorithms have recently been re-evaluated since they are well-behaved with function approximation. As opposed to the action value function based reinforcement learning such as Q-learning, the PGRL algorithms are naturally integrated with function approximators, and therefore they can deal with continuous actions. There exist several methods to compute the gradient of the average reward $\boldsymbol{\Delta}^i$. In the current implementation, we choose the GPOMDP algorithm (Baxter and Bartlett, 2001) and the actor-critic method (Konda and Tsitsiklis, 2003). At first, we briefly introduce the GPOMDP algorithm when the reward depends on the action as well as the state. According to the current state and action, the function $\boldsymbol{\psi}_t$ is defined by

$$\boldsymbol{\psi}_t(\boldsymbol{x}_t, \boldsymbol{u}_t) \triangleq \frac{1}{\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{u}_t)} \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{u}_t)}{\partial \boldsymbol{\theta}}.$$

The learning agent interacts with the environment, producing a state, action, reward sequence. After receiving experiences ($\boldsymbol{x}_t$, $\boldsymbol{u}_t$, $\boldsymbol{x}_{t+1}$, $\boldsymbol{u}_{t+1}$, $\boldsymbol{r}_{t+1}$), the GPOMDP updates an eligibility traces $\boldsymbol{z}_t \in \mathrm{R}^n$

$$\boldsymbol{z}_{t+1} = \beta \boldsymbol{z}_t + \boldsymbol{\psi}_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$$

where $\beta \in [0, 1)$ is a discount rate that controls the variance of the gradient estimate. Since $\boldsymbol{z}_t$ is independent of the reward functions, $\boldsymbol{z}_t$ can be used for estimating gradients of different average rewards. Then, all the gradients are updated in the same manner. That is, the gradient of the long-term average reward is approximated by

$$\boldsymbol{\Delta}^i_{t+1} = \boldsymbol{\Delta}^i_t + \frac{1}{t+1}\left[r^i_{t+1}\left(\boldsymbol{z}_{t+1} + \boldsymbol{\psi}_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_{t+1})\right) - \boldsymbol{\Delta}^i_t\right] \tag{3}$$

for all $i = 1, \ldots, m$. The estimate of the average reward r^i is updated by

$$\rho^i_{t+1} = \rho^i_t + \alpha_r(r^i_{t+1} - \rho^i_t), \tag{4}$$

where α_r is a positive step-size meta-parameter. It is noted that $\rho_{i,t+1}$ gives an estimate of $g_i(\theta)$ and plays an important role for finding active constraints. Although the GPOMDP can estimate the gradient with less number of parameters, it has a large variance as $\beta \to 1$.
We also use a simplified method based on the actor critic method (Konda & Tsitsiklis, 2003) that exploits a value function. The gradient of the long-term average reward is calculated by

$$\boldsymbol{\Delta}_{t+1}^{i} = \boldsymbol{\Delta}_{t}^{i} + \frac{1}{t+1}\left[Q^{i}(\boldsymbol{x}_t, \boldsymbol{u}_t)\boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{u}_t) - \boldsymbol{\Delta}_{t}^{i}\right], \tag{5a}$$

$$Q^{i}(\boldsymbol{x}, \boldsymbol{u}) = (\boldsymbol{w}^{i})^{\top}\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u}). \tag{5b}$$

where $Q^i(\boldsymbol{x}, \boldsymbol{u})$ and $\boldsymbol{w}^i$ denote an approximated state-action value function and a parameter vector, respectively. In order to train $\boldsymbol{w}^i$, the standard temporal difference method is carried out

$$\boldsymbol{w}_{t+1}^{i} = \boldsymbol{w}_{t}^{i} + \alpha_r \delta_t^i \boldsymbol{z}_{t+1},$$

where the temporal difference δ_t^i is defined by

$$\delta_t^i = r_{t+1}^i - \rho_{t+1}^i + (\boldsymbol{w}_t^i)^{\top}\left[\boldsymbol{\psi}_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_{t+1}) - \boldsymbol{\psi}_t(\boldsymbol{x}_t, \boldsymbol{u}_t)\right].$$

Although Konda's actor-critic requires an additional learning mechanism to approximate the state-action value function, it can utilize the Markov property.

2.3 Gradient projection

As described in section 2.2, the average rewards and their gradients are obtained at the end of each episode. Next, we apply a gradient projection method to solve the maximization problem with inequality constraints. In order to derive a modified learning rule, a set of indices of the active inequality constraints is defined by

$$\mathcal{A} = \left\{ i \mid \rho^i - G^i \le 0,\ i = 2, \ldots, m \right\}$$

and let $a = |A|$ denote the number of active constraints in which A is called an active set. If no constraints are active (the case $a = 0$), the solution lies at the interior of the feasible region. A standard learning rule can be applied in the case of $a = 0$, the case $a \neq 0$ is considered hereafter. With the outputs from the multiple critics, we define

$$\boldsymbol{g}_{\mathcal{A}} \triangleq \left[\rho^{i_1} - G^{i_1} \quad \cdots \quad \rho^{i_a} - G^{i_a}\right]^{\top},$$
$$\boldsymbol{N}_{\mathcal{A}} \triangleq \left[\boldsymbol{\Delta}^{i_1} \quad \cdots \quad \boldsymbol{\Delta}^{i_a}\right],$$

where i_a is an index to count the element in A. Fig.2 illustrates the basic idea of gradient projection based on the nonlinear programming. The gray are represents the feasible region and therefore the policy parameter vector at the k-th episode must approach the feasible region while move into the direction $\boldsymbol{\Delta}_1$. Suppose that the policy parameter is modified without considering a restoration move $\boldsymbol{d}$. When the k-th episode ends, the update rule is given by

$$\boldsymbol{\theta}_k' = \boldsymbol{\theta}_k + \alpha_1 \boldsymbol{s},$$

where $\boldsymbol{s}$ is the steepest ascent direction.

By using the estimated gradients, the set of active constraints can be approximated by the following linear equation:

$$\boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{\theta} + \boldsymbol{b} \approx \mathbf{0},$$

where $\boldsymbol{b}$ is an appropriate vector. Since the gradient projection method (Rosen, 1960) assumes that $\boldsymbol{\theta}$ lies in the subspace tangent to the active constraints, both $\boldsymbol{\theta}'_k$ and $\boldsymbol{\theta}_k$ should satisfy the above equations. Then, we obtain an equality constraints $\boldsymbol{N}^{\mathrm{T}}\boldsymbol{s} = \mathbf{0}$. Since the steepest ascent direction $\boldsymbol{s}$ satisfying the above constraints is required, we can pose this problem as

$$\max\ \boldsymbol{s}^{\top}\boldsymbol{\Delta}^{1} \quad \text{s.t.} \quad \boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{s} = \mathbf{0} \ \text{ and } \ \boldsymbol{s}^{\top}\boldsymbol{s} = 1.$$

The second constraint is required to normalize $\boldsymbol{s}$. In order to solve this problem with equality constraints, the Lagrange multiplier method is applied. Now, the Lagrangian function is defined as

$$\mathcal{L}(\boldsymbol{s}, \boldsymbol{\lambda}, \kappa) = \boldsymbol{s}^{\top}\boldsymbol{\Delta}^{1} - \boldsymbol{s}^{\top}\boldsymbol{N}_{\mathcal{A}}\boldsymbol{\lambda} - \kappa(\boldsymbol{s}^{\top}\boldsymbol{s} - 1),$$

where $\boldsymbol{\lambda}$ and κ are Lagrange multipliers. The condition for L to be stationary is given by:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}} = \boldsymbol{\Delta}^{1} - \boldsymbol{N}_{\mathcal{A}}\boldsymbol{\lambda} - 2\kappa\boldsymbol{s} = \mathbf{0}.$$

By pre-multiplying $\boldsymbol{N}_A^{\mathrm{T}}$ into the above equaiton, we obtain

$$\boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{\Delta}^{1} - \boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{N}_{\mathcal{A}}\boldsymbol{\lambda} = \mathbf{0}. \tag{6}$$

It should be noted that $\boldsymbol{N}_A^{\mathrm{T}}\boldsymbol{s} = \mathbf{0}$. If $\boldsymbol{N}_A^{\mathrm{T}}\boldsymbol{N}_A$ is invertible, the Lagrange multipliers $\boldsymbol{\lambda}$ can be represented by

$$\boldsymbol{\lambda} = \left(\boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{N}_{\mathcal{A}}\right)^{-1}\boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{\Delta}^{1}. \tag{7}$$

From Equations (6) and (7), $\boldsymbol{s}$ is derived as

$$\boldsymbol{s} = \frac{1}{2\kappa}\left[\boldsymbol{I} - \boldsymbol{N}_{\mathcal{A}}\left(\boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{N}_{\mathcal{A}}\right)^{-1}\boldsymbol{N}_{\mathcal{A}}^{\top}\right]\boldsymbol{\Delta}^{1},$$

where the scalar Lagrange multiplier κ remains unknown. However, this parameter is not important because we are interested in the modified direction of the gradient $\boldsymbol{\Delta}^{1}$. As a result, when k-th episode ends, the policy parameters are update as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_1\boldsymbol{P}\boldsymbol{\Delta}^{1} - \alpha_e\boldsymbol{d} \tag{8}$$

where $\alpha_1, \alpha_e \in [0, 1)$ are learning rates, $\boldsymbol{P}$ is a matrix that projects $\boldsymbol{\Delta}^{1}$ into the subspace tangent to the active constraints, and is a restoration move for the violating constraints. The projection matrix $\boldsymbol{P}$ and restoration move $\boldsymbol{d}$ are given by

$$\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{N}_{\mathcal{A}}\left(\boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{N}_{\mathcal{A}}\right)^{-1}\boldsymbol{N}_{\mathcal{A}}^{\top}, \tag{9}$$

$$\boldsymbol{d} = \boldsymbol{N}_{\mathcal{A}}\left(\boldsymbol{N}_{\mathcal{A}}^{\top}\boldsymbol{N}_{\mathcal{A}}\right)^{-1}\boldsymbol{g}_{\mathcal{A}}. \tag{10}$$

It should be noted that $\boldsymbol{Pd} = \mathbf{0}$. If the active set A is empty, $\boldsymbol{P}$ and $\boldsymbol{d}$ are set to the identity matrix and zero vector, respectively.

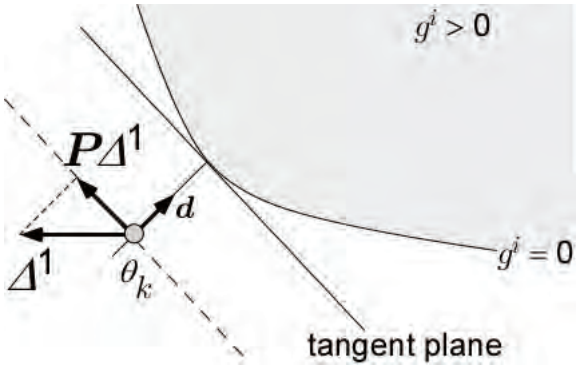

Fig. 2. Graphical interpretation of gradient projection. The gray area represents the feasible region in the policy parameter space. $\boldsymbol{\theta}_k$, $\boldsymbol{\Delta}^1$, and $\boldsymbol{d}$ are the current policy parameter, the policy gradient of the average reward r^1, and the restoration vector, respectively. P is a projection matrix which maps the vector into the subspace tangent to the feasible region.

Here, we should note two points when $\boldsymbol{P}$ and $\boldsymbol{d}$ are computed in the program. At first, it must be noted that the matrix $\boldsymbol{N}_A^{\mathrm{T}}\boldsymbol{N}_A$ in (9) and (10) is not invertible if the set of active constraint gradients $\{\boldsymbol{\Delta}^{i_j} \mid \mathrm{j} = 1, \ldots, a\}$ is linearly dependent. In practice, rank deficiency of $\boldsymbol{N}_A^{\mathrm{T}}\boldsymbol{N}_A$ is sometimes observed due to the accuracy of numerical computation and/or biased samples. The pseudo-inverse of $\boldsymbol{N}_A^{\mathrm{T}}\boldsymbol{N}_A$ should be used if it is not full-rank. In addition we must consider the situation where $\boldsymbol{P\Delta^1}=\mathbf{0}$ because it may be possible to modify the parameters. This situation can be detected by using Lagrange multipliers (7). If $\boldsymbol{\lambda}$ has no negative components, we have a solution and terminate. Otherwise, the constraint with maximum Lagrange multiplier is calculated by

$$r = \arg\max_{i \in \mathcal{A}} \lambda_i,$$

and then it is removed from the active set as $A \leftarrow A \backslash \{\mathrm{r}\}$. After deleting one constraint from the active set, $\boldsymbol{P}$ and $\boldsymbol{d}$ are evaluated again by using (9) and (10).

2.4 Backups of the long-term average rewards

Since the CPGRL uses ρ^i (i = 2,..., m) to determine the set of active constraints A, these estimates directly specify the feasible region in the policy parameter space.

3. Computer simulation in a simple MDP task

3.1 MDP setting

In order to evaluate the performance of the CPGRL from a viewpoint of constrained optimization problems, we apply the CPGRL to a simple three-state Markov Decision Problem shown in Fig.3. The sets of states and actions are $\{s_1, s_2, s_3\}$ and $\{a_1, a_2, a_3\}$, respectively. Let s and a denote the original state and action in this MDP problem while the variables x and u represent the state and action at each time step. Therefore, $x_t \in \{s_1, s_2, s_3\}$

and $u_t \in \{a_1, a_2, a_3\}$. Each action achieves the intended effect with probability 0.8, but it makes a random transition otherwise. For example, from the state s_1 the action a_1 moves the agent to s_2, s_3, s_1 with probabilities 0.8, 0.1, 0.1, respectively.
One intrinsic reward r^1 and three extrinsic rewards r^2, r^3, r^4 are prepared in this problem;

$$r^1 = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & 2 \\ 1 & 0 & -1 \end{bmatrix}, \; r^2 = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \; r^3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix}, \; r^4 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix},$$

where $r^i = (r_{jk})$ is a reward value of r^i when the action a_k is selected at the state s_j. For instance, the reward vector is $\boldsymbol{r}$ = [2 1 -1 0]$^{\mathrm{T}}$ when the agent selects the action a_3 at the state s_2. Under these settings, the optimal policy is to select a_1 in each state, and the corresponding long-term average reward vector is [1 0 0 0]$^{\mathrm{T}}$. It should be noted that the extrinsic rewards are competitive with each other. As the stochastic policy, we use a lookup table with softmax distribution

$$\mu_{\boldsymbol{\theta}}(x_t, u_t) = \frac{\exp(\theta_{x_t,u_t})}{\sum_{u'} \exp(\theta_{x_t,u'})},$$

yielding a total of nine policy parameters. That is, the policyparameters are assigned such that $\theta_1 = \theta_{s_1,a_1}$, $\theta_2 = \theta_{s_1,a_2}$, and so on.

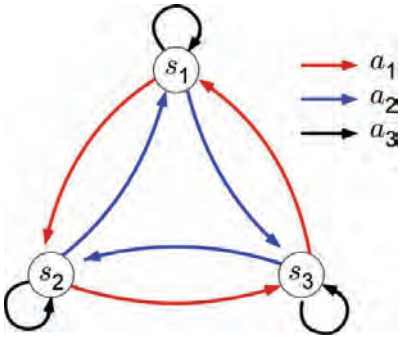

Fig. 3. Simple MDP with three states, three actions, and four reward functions.

The GPOMDP algorithm is adopted for estimating the policy gradient. Each policy parameter is randomly initialized with a uniform distribution over the interval [0, 1]. Thresholds used in inequality constraints are set as $G^2 = G^3 = G^4 = 0$. Other meta-parameters are set as follows: $\alpha_\rho = 0.02$, $\beta = 0.99$, $\alpha_1 = \alpha_e = 0.02$. These values are determined by trial and error. In order to compare the performance, we consider two different approaches named the CONstraints-Based (CONB) and the SUM method. The CONB switches the policy gradient of each reward according to the following condition:

$$\boldsymbol{\Delta} = \begin{cases} \boldsymbol{\Delta}^1 & \text{if } g^i - G^i > 0 \text{ for } i = 2, \ldots, m \\ \boldsymbol{\Delta}^j & \text{otherwise}, \quad j = \arg\min_{i \in \mathcal{A}} (\rho^i - G^i). \end{cases}$$

to maximize the average reward of r^1 and to satisfy the constraints. The SUM learns to maximize the average reward of the summation of all rewards

$$r^{\mathrm{sum}} = r^1 + r^2 + r^3 + r^4$$

based on the standard policy gradient method. However, it is expected that the learned parameters does not satisfy the constraints since the SUM does not consider the constraints at all. The agent starts at the state $x_0 = s_1$. The number of episodes and steps are $N_T = 100$ and $N_K = 10000$, respectively. We perform 20 simulation runs.

3.2 Experimental results

Fig.4 shows the means and the standard deviations of 20 simulation runs obtained by the CPGRL, CONB, and SUM, respectively. The CPGRL found the parameters that satisfy the inequality constraints at the very early stage of learning, and the standard deviations of the average rewards of constraints were very small after 1x10³-th episode. The CONB also obtained the policy parameters satisfying constraints, but it took a longer time than the CPGRL. The long-term average reward of r^1 was gradually increased by the CPGRL. Interestingly, the CONB failed to maximize the average reward of r^1. In addition, we found that the standard deviation estimated by the CONB was larger than that of the CPGRL. The performance of the SUM was different from those of the CPGRL and the CONB because the constraint on r^3 was violated at all. Although the SUM obtained the best average reward on r^1, it failed to find the policy parameters satisfying the constraint on r^3.

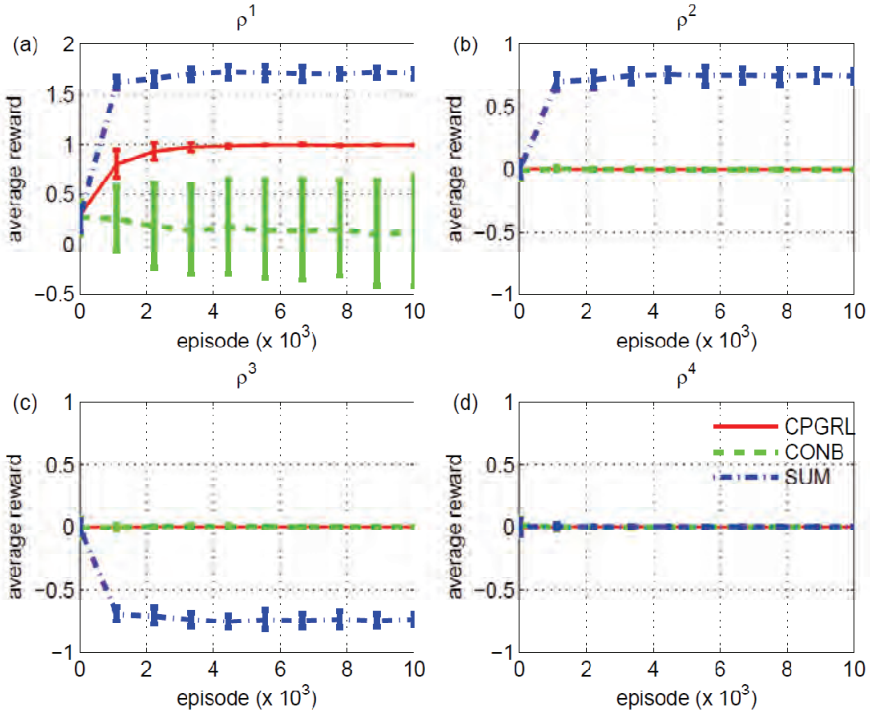

Fig. 4. Transition of the estimated average rewards. (a) ρ^1, (b) ρ^2, (c) ρ^3, and (d) ρ^4, respectively. These figures show the means and standard deviations of 20 independent runs.

Then, we checked the stochastic policy during the learning process. Fig.5 shows the evolution of the policy parameters of the CPGRL, CONB, and SUM, respectively. In this figure, all policies lie in the lower-left triangle, and gray-colour represents the average reward of r^1. Although all methods could obtain appropriate action at the state s_1 ($\Pr(a_1 \mid s_1) = 1$ is optimal), the CONB and SUM obtained inappropriate actions at the states s_2 and s_3. For example, the SUM leaned to select a_3 at s_2 because the large positive reward (2 + 1 - 1 + 0 = 2) was received in this case. Obviously, this violated the constraints on r^3. The CONB failed to obtain the appropriate action at s_3. It should be noted that both of a_1 and a_2 did not generate negative rewards in this state. However, the CONB failed to improve the average reward of r^1 because the gradient of r^1 was rarely selected. Since the estimated average reward by (4) is not deterministic, some constraints were violated suddenly. On the contrary, the CPGRL successfully obtained the optimal policy that satisfied all constraints in this simulation.

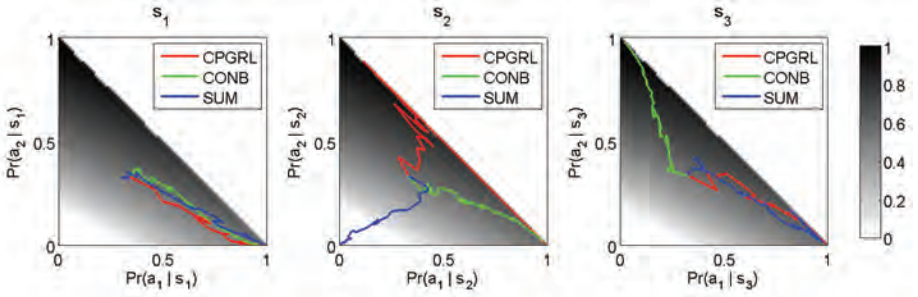

Fig. 5. Evolution of probabilities for action selection calculated from the policy parameters. Each lower triangle shows a feasible region of the values of the probabilities.

4. Control task of a robotic arm

4.1 Simulation setting

Then we conduct a control task of a robotic arm to investigate how the thresholds G^i used in the inequality constraints affect the learning processes in the CPGRL framework. Fig.6 (a) shows a simulated environment. There exists a typical two-link arm that can interact with four objects (circle, star, square, and triangle). These four objects are fixed in the environment. The intrinsic reward r^1 is computed from the distance between the position of the end-effector of the arm and the nearest objects. Fig.6 (b) shows a distribution of r^1. This dense reward function enables the robotic arm to learn touching behaviours actively. Then, two extrinsic rewards r^2 and r^3 are introduced in this task. The first extrinsic reward gives upper and lower bounds on the joint angles $\phi_1$and ϕ_2 while the second extrinsic reward depends on whether the touched object is appetitive or aversive:

$$r^2 = \begin{cases} 0 & \frac{\pi}{10} \leq \phi_1 \leq \frac{\pi}{2},\ \frac{\pi}{4} \leq \phi_2 \leq \frac{3\pi}{4}, \\ -1 & \text{otherwise}, \end{cases}$$

$$r^3 = \begin{cases} 1 & \text{if the object is appetitive}, \\ -1 & \text{if the object is aversive}, \\ 0 & \text{otherwise}. \end{cases}$$

It should be noted that zero reward is given when the robotic arm touches the circular and triangular objects.

The continuous state is $\boldsymbol{x} = [\phi_1, \phi_2]^{\mathrm{T}}$ while the continuous action consists of desired joint velocities, $\boldsymbol{u} = [\Delta\phi_1, \Delta\phi_2]^{\mathrm{T}}$. To represent the stochastic policy, we use a normalized Gaussian network,

$$\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{u}_t) = \eta_1 \exp\left[-\eta_2 \left\|\boldsymbol{u}_t - \boldsymbol{\theta}^{\top}\boldsymbol{n}(\boldsymbol{x}_t)\right\|\right],$$

where η_1, η_2 and $\boldsymbol{n}(\boldsymbol{x})$ denote the constant values and the vector of the basis function. The number of basis functions is 40, determined by trial and error. In this experiment, Konda's actor-critic method is used to compute the policy gradient.

This simulation does not consider dynamics of the arm. The initial joint angles are initialized randomly. The same meta-parameters such as learning rates are used in section 3. When the arm touches one of the objects or N_T = 1000 time steps are expired, the episode terminates. One episode lasts for N_K = 10000 episodes and we perform 20 simulation runs.

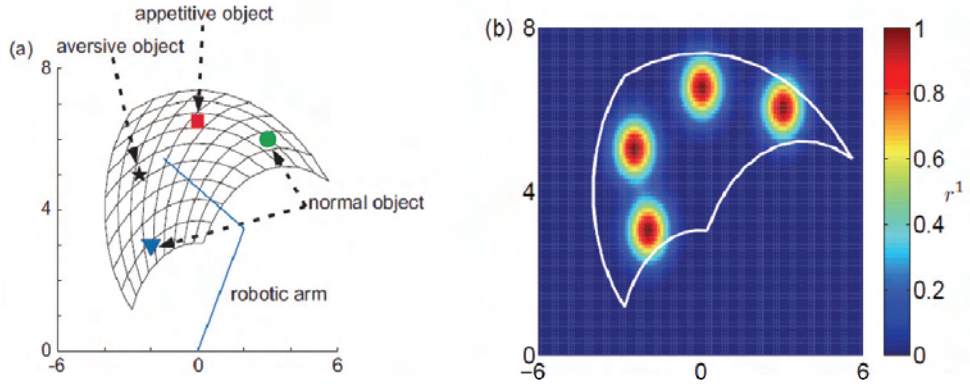

Fig. 6. Control task of a robotic arm. (a) Simulated environment where a mesh represents a reachable region of the end-effector of the arm. (b) Distribution of the intrinsic reward r^1.

4.2 Experimental results

Fig.7 (a) shows the number of touches on the objects in each 100 episodes and a typical learned behaviour at the end of episodes when the robotic arm is motivated only by the intrinsic reward. Since r^1 was a dense reward function, it was not hard to obtain touching behaviours. Then we introduce two constraints by extrinsic rewards with thresholds G^2 = G^3 = 0. Fig.7 (b) shows the experimental results. At the early stage of learning, the robotic arm touched the aversive star object. Then, it learned to avoid the aversive star object after about 1x10^3 episodes. The bottom of Fig.7 (b) shows a typical learned behaviour. The end-effector of the robot arm was initially located in the neighbourhood of the aversive star object, but the arm touched the triangular object.

Finally, we strengthen the constraint by setting G^3 = 0.5 and observe the behaviours shown in Fig.7 (c). Since the robotic arm has to touch the square object in order to obtain a positive r^2, the number of touches on the square objects increases while those on other objects are gradually reduced to zero. It is revealed that G^2 is a sensitive threshold that affects the resultant behaviours. The obtained behaviour at the end of episode is shown in the bottom of Fig.7 (c).

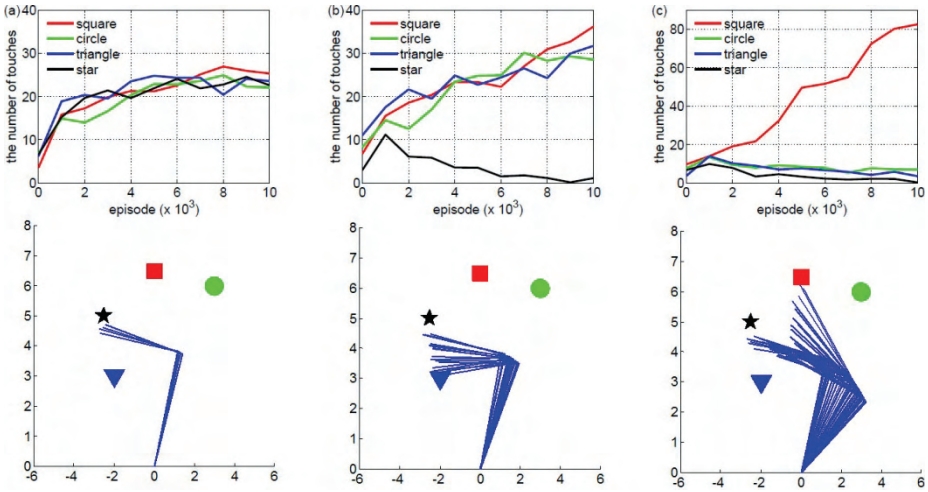

Fig. 7. Number of touches on the objects and learned behaviours. (a) No constraints. (b) Normal constraints: $G^2 = G^3 = 0$. (c) Tight constraints: $G^2 = 0$ and $G^3 = 0.5$.

5. Conclusion

In this chapter we have proposed the CPGRL that maximizes the long-term average reward under the inequality constraints that define the feasible policy space. Experimental results encourage us to conduct the robotic experiments because one of our interests is to design the developmental learning methods for real hardware systems. Although we could not discuss the design principles of intrinsic and extrinsic rewards to establish a sustainable and scalable learning progress, this is very important. We think that the CPGRL gives the first step towards developmental learning. We develop the experimental setup that integrates the CPGRL and the technique of the embodied evolution in our multi-robot platform named "Cyber Rodents" (Doya & Uchibe, 2005). In this case, the intrinsic reward is computed from sensor outputs while the extrinsic rewards are given according the external events such as collisions with obstacles, capturing a battery pack, and so on. We have reported that good exploratory reward is acquired as the intrinsic reward through the interaction among three mobile robots (Uchibe and Doya, to appear). We also plan to test other types of intrinsic rewards used in previous studies (Singh et al., 2005; Oudeyer & Kaplan, 2004).

Finally, we describe three foreseeable extensions of this study. At first, we improve the efficiency of numerical computation. It is known that the learning speed of standard PGRL can be slow due to high variance in the estimate. Then, the Natural Policy Gradient (NPG) method (Morimura et al., 2005) supported by the theory of information geometry is implemented to accelerate the speed of learning. Secondly, we develop a method to tune the thresholds used in the inequality constraints during learning processes. As shown in section 4, the learned behaviours were strongly affected by the setting of the thresholds. From a viewpoint of constrained optimization problems, G^i is just a meta-parameter given by the experimenters. However, the learning agent will show a variety of behaviours by changing these thresholds. We think that CPGRL has a potential to create new behaviours through the interaction between intrinsic and extrinsic rewards.

6. References

Barto, A.G.; Singh, S. & Chentanez, N. (2004). Intrinsically Motivated Learning of Hierarchical Collections of Skills, *Proceedings of International Conference on Developmental Learning*

Baxter, J. & Bartlett, P.L. (2001). Infinite-horizon gradient-based policy search. *Journal of Artificial Intelligence Research,* Vol. 15, pages 319-350

Deci, E.L. & Flaste, R. (1996). *Why we do what we do: understanding self-motivation*, Penguin books

Dolgov, D. & Durfee, E. (2005). Stationary deterministic policies for constrained MDPs with multiple rewards, costs, and discount factors, *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 1326-1331

Doya, K. & Uchibe, E. (2005). The Cyber Rodent Project: Exploration of adaptive mechanisms for self-preservation and self-reproduction. *Adaptive Behavior*, Vol. 13, pages 149-160

Feinberg, E. & Shwartz, A. (1999). Constrained dynamic programming with two discount factors: Applications and an algorithm. *IEEE Transactions on Automatic Control*, Vol. 44, pages 628-630

Konda, V.R. & Tsitsiklis, J.N. (2003). Actor-critic algorithms. *SIAM Journal on Control and Optimization,* Vol. 42, No. 4, pages 1143-1166

Meeden, L.A. ; Marshall, J.B. & Blank, D. (2004). Self-Motivated, Task-Independent Reinforcement Learning for Robots, *Proceedings of 2004 AAAI Fall Symposium on Real-World Reinforcement Learning*

Morimura, T. ; Uchibe, E. & Doya, K. (2005). Utilizing the natural gradient in temporal difference reinforcement learning with eligibility traces, *Proceedings of the 2nd International Symposium on Information Geometry and its Application*, pp. 256-263

Oudeyer, P.-Y. & Kaplan, F. (2004). Intelligent adaptive curiosity: A source of self-development, *Proceedings of the 4th International Workshop on Epigenetic Robotics*, pp. 127-130

Oudeyer, P.-Y., Kaplan, F. & Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 2, pages 265-286

Rosen, S.A. (1960). The gradient projection method for nonlinear programming --- part I: linear constraints. *Journal of the Society for Industrial and Applied Mathematics*, Vol. 8, No. 1, pages 181-217

Singh, S. ; Barto, A.G. & Chentanez, N. (2005). Intrinsically motivated reinforcement learning, *Advances in Neural Information Processing Systems 17*, pp. 1281-1288, MIT Press

Stout, A.; Konidaris, G.D. & Barto, A.G. (2005). Intrinsically motivated reinforcement learning: A promising framework for developmental robot learning, *Proceedings of the AAAI Spring Symposium Workshop on Developmental Robotics*

Sutton, R.S. & Barto, A.G. (1998). *Reinforcement Learning: An Introduction,* MIT Press

Sutton, R.S.; Precup, D. & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence,* Vol. 112, pages 181-211

Uchibe, E. & Doya, K. (to appear). Finding Intrinsic Rewards by Embodied Evolution and Constrained Reinforcement Learning. *Neural Networks*