Reinforcement Learning by Policy Improvement Making Use of Experiences of The Other Tasks

Hajime Kimura and Shigenobu Kobayashi
Tokyo Institute of Technology, JAPAN
gen@fe.dis.titech.ac.jp, kobayasi@dis.titech.ac.jp

Abstract. We propose a new reinforcement learning algorithm based on a policy gradient method combining with importance sampling. Stochastic gradient methods can easily be applied to continuous state-action control problems, and also it can improve policies under some class of POMDP environments, however, the algorithms are limited to on-policy learning, i.e., the improvement is executed by the experiences under the only target policy. Importance sampling techniques enable the gradient methods to make use of all experiences for the learning. We develop the algorithm for continuing (not episodic) tasks under discounted reward criteria, and demonstrate through simulations of a maze world and a two-link arm robot problem.

1 Introduction

Reinforcement learning (RL) [6] is a promising approach for robots to obtain control rules through a process of trial and error. Learning control in real world applications is to deal with both continuous state-action spaces, and partially observability that causes non-Markovian effects. Stochastic gradient methods [7] are one approach to overcome such problems. In many policy gradient methods, a parameterized stochastic policy is updated according to the gradient of the value function with respect to the policy parameters. It is easy to implement multidimensional continuous action by using some continuous distribution (e.g. normal distribution) for the policy function [1][2]. Stochastic gradient methods can be classified to direct policy search algorithms [9] that would not make use of Markov properties of the environments. Therefore, the stochastic gradient methods can improve policies under some class of POMDP environments.

A drawback of the conventional stochastic gradient methods is that the learning is too slow. One reason is that the algorithms are limited to on-policy learning, i.e., the improvement can be executed by the experiences under the only target policy. In many real-robot applications, the same robot must experience trial and error to learn many policies for the respective tasks. Therefore, sharing the same experiences for the learning of all policies will save great amount of trials.

Importance sampling techniques enable the policy search algorithms to execute off-policy policy improvement [3], but these methods are limited to applying episodic tasks. In this paper, we propose a new reinforcement learning algorithm based on a policy gradient method combining with importance sampling, and develop it for continuing (not episodic) tasks under discounted reward criteria.

2 Background

2.1 Environment Model

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 general, the reward and the next state may be random, but their probability distributions are assumed to depend only on s_t and a_t in Markov decision processes (MDPs), in which many reinforcement learning algorithms are studied. In MDPs, 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 according to 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 reinforcement learning is to construct a policy that maximizes the agent's performance. A policy π is defined as probability distribution over action space under given state. A natural performance measure for finite (or infinite) horizon tasks is the cumulative discounted reward:

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

where the discount factor, $0 \leq \gamma < 1$ specifies the importance of future rewards, and V_t is the value at time t. In MDPs, the value under policy π is defined as a function of state:

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

where $E\{\cdot\}$ denotes the expectation. The objective in MDPs is to find an optimal policy that maximizes the value of each state s defined by Equation 2.

In this standard notation of MDPs, the reward is defined as scalar, but in this paper, we give reward signal as a vector, and each component in the reward vector is given for each different task respectively, and each task holds its policy.

2.2 Importance Sampling

Importance sampling is a classical technique for handling a mismatch of distributions, that is, we would like data drawn from the distribution of the target policy π to estimate values (or its gradient), but the data is drawn from the distribution of the other behavior policy π'. Let $H = \{< s_0, a_0, r_0, \ldots s_T, a_T, r_T, s_{T+1} >\}$ denote the set of all possible experience sequence of length $T+1$, and let a history $h(s) \in H$ with $s_0 = s$. $R(h)$ denotes the cumulative discounted rewards: $R(h) = \sum_{t=0}^{T} \gamma^t r_t$. Consider a sample history drawn from the distribution induced by the behavior policy π' executing multiple times in the environment. Using a standard importance sampling method [3] [5], the estimated state value is given by

$$\hat{V}^{\pi}(s) = R(h)\frac{\Pr(h|\pi)}{\Pr(h|\pi')} = R(h)\frac{\pi_0\pi_1\cdots\pi_T}{\pi'_0\pi'_1\cdots\pi'_T} = \left(r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \gamma^T r_T\right)\frac{\pi_0\pi_1\cdots\pi_T}{\pi'_0\pi'_1\cdots\pi'_T}, \tag{3}$$

where π_t denotes probability (or density) of executing action a_t under given policy π, r_t denotes the reward in the sequence h at time t. Intuitively, the likelihood ratio $\frac{\pi_0\pi_1\cdots\pi_T}{\pi'_0\pi'_1\cdots\pi'_T}$ on reward r_t should not depend on the future after time t. Based on this idea, per-decision importance sampling [4] is shown by:

$$\hat{V}^{\pi}(s) = \left(r_0\frac{\pi_0}{\pi'_0}\right) + \left(\gamma r_1\frac{\pi_0\pi_1}{\pi'_0\pi'_1}\right) + \cdots + \left(\gamma r_{T-1}\frac{\pi_0\pi_1\cdots\pi_T}{\pi'_0\pi'_1\cdots\pi'_T}\right) = \sum_{t=0}^{T}\gamma^t r_t \prod_{k=0}^{t}\frac{\pi_k}{\pi'_k} \tag{4}$$

It is shown that these algorithms are unbiased, that is, $\mathrm{E}\{\hat{V}^\pi(s)\} = V^\pi(s)$ where $T \to \infty$. For the policy improvement, we should calculate the gradient of the value function. Assume that the parameterized policy function π is given by the policy parameter θ, and the behavior policy π' is given by a parameter θ', then

$$\frac{\partial}{\partial\theta}V^\pi(s) = \mathrm{E}\left\{\frac{\partial}{\partial\theta}\sum_{t=0}^{T}\gamma^t r_t \prod_{k=0}^{t}\frac{\pi_k}{\pi'_k}\right\} = \mathrm{E}\left\{\sum_{t=0}^{T}\gamma^t r_t \frac{\frac{\partial}{\partial\theta}\prod_{k=0}^{t}\pi_k}{\prod_{k=0}^{t}\pi'_k}\right\} \tag{5}$$

The calculation of the gradient of the action likelihood with respect to the policy parameters $\frac{\partial}{\partial\theta}\prod_{k=0}^{t}\pi_k$ is given by:

$$\frac{\partial}{\partial\theta}\prod_{t=0}^{k}\pi_k = \left(\prod_{k=0}^{t}\pi_k\right)\frac{\partial}{\partial\theta}\ln\left(\prod_{t=0}^{k}\pi_k\right) = \left(\prod_{k=0}^{t}\pi_k\right)\left(\sum_{k=0}^{t}\frac{\partial}{\partial\theta}\ln\pi_k\ .\right) \tag{6}$$

This calculation can easily be done incrementally by accumulating the eligibility $\frac{\partial}{\partial\theta}\ln\pi_t$.

3 Gradient Method using Importance Sampling

3.1 The Algorithm

We develop a policy improvement algorithm based on the value gradient shown in the previous section. Figure 1 shows the outline of the algorithm. When the behavior policy π' is equal to the target policy π, then the algorithm is exactly the same as the standard stochastic gradient methods. Multiple policies can be applied to the target policy, however, each policy needs memory space for the policy parameter vector θ, its eligibility trace $\bar{e}(t)$ and discounted sum of the likelihood ratio $L(t)$, that is a scalar variable, respectively. A very similar idea is already proposed by [8] except for applying continuing tasks.

The policy parameter vector θ would be updated appropriately by the sum of $\Delta\theta_t$; In our experiments, $\Delta\theta$ is averaged according to $\Delta\bar{\theta}_t \leftarrow \Delta\theta_t + \gamma\Delta\bar{\theta}_{t-1}$ and update policy parameters by $\theta \leftarrow \theta + \alpha\frac{\Delta\bar{\theta}_t}{|\Delta\bar{\theta}_t|}$ for each time step, where α is a small positive constant, and $|\cdot|$ denotes L_1 norm.

3.2 Features

- Since the likelihood ratio $\frac{\pi_0\pi_1\cdots\pi_T}{\pi'_0\pi'_1\cdots\pi'_T}$ is calculated incrementally on each time step by $L(t)$, the learner does not need to hold state-action-reward sequences. It is incorporated into the calculation of the eligibility trace in Figure 1.
- Although the behavior policy is limited to one on each step, multiple target policies can be learned concurrently. If the reward signal is given by a vector, and each component of the vector is assigned to the corresponding task, then the agent can learn multiple tasks from the same trial-and-error experiences. Especially, when all policies are similar at the early stage of the learning, the policies can share the same experiences, and the learning efficiency will be proportional to the number of the tasks.
- However, the learning proceeds and the differences of the action selection probability between the tasks are getting large, then the weight of sharing the experiences tends to get small because the likelihood ratio π'/π would approach to zero.

1. Observe state s_t, choose action a_t with probability (or density) following a behavior policy $\pi'(a_t, \theta, s_t)$, and perform it.

2. Calculate π_t that is a probability of selecting action a_t under the condition of the target policy π in the state s_t.

3. Calculate the eligibility of the target policy parameter θ: $e(t) = \frac{\partial}{\partial \theta} \ln \pi_t$

4. Update the eligibility trace according to:

$$\begin{aligned} L(t) &= (1 + \gamma L(t-1)) \frac{\pi_t}{\pi'_t} \\ \bar{e}(t) &= L(t)e(t) + \gamma \frac{\pi_t}{\pi'_t} \bar{e}(t-1), \end{aligned} \tag{7}$$

   where γ is a discount factor, and $L(t)$ is an accumulated likelihood ratio.

5. Observe immediate reward r_t and update policy parameters:
   $\Delta\theta_t = (r_t - b)\bar{e}(t)$, where b is a reinforcement baseline parameter.

6. Let $t \leftarrow t + 1$, and go to step 1.

Figure 1: The policy improvement algorithm combined with importance sampling.

3.3 Analysis of the Algorithm

As sown in Figure 1, the discounted sum of the likelihood ratio $L(t)$ is given by

$$L(t) = (1 + \gamma L(t-1))\frac{\pi_t}{\pi'_t} = \frac{\pi_t}{\pi'_t} + \gamma \frac{\pi_t \pi_{t-1}}{\pi'_t \pi'_{t-1}} + \gamma^2 \frac{\pi_t \pi_{t-1} \pi_{t-2}}{\pi'_t \pi'_{t-1} \pi'_{t-2}} + \cdots + \gamma^{t-1} \frac{\pi_t \pi_{t-1} \cdots \pi_0}{\pi'_t \pi'_{t-1} \cdots \pi'_0} \tag{8}$$

The backups of the policy parameter $\Delta\theta$ are given by

$$\begin{aligned} \Delta\theta_t &= (r_t - b)\bar{e}(t) = (r_t - b)\left(L(t)e(t) + \gamma \frac{\pi_t}{\pi'_t} \bar{e}(t-1) \right) \\ &= (r_t - b)\left(L(t)e(t) + \gamma \frac{\pi_t}{\pi'_t} L(t-1)e(t-1) + \cdots + \gamma^{t-1} \frac{\pi_t \pi_{t-1} \cdots \pi_0}{\pi'_t \pi'_{t-1} \cdots \pi'_0} L(0)e(0) \right) \end{aligned} \tag{9}$$

From Equation 8 and 9, we get

$$\begin{aligned} \Delta\theta_t &= (r_t - b)\left[e(t)\left(\frac{\pi_t}{\pi'_t} + \gamma \frac{\pi_t \pi_{t-1}}{\pi'_t \pi'_{t-1}} + \gamma^2 \frac{\pi_t \pi_{t-1} \pi_{t-2}}{\pi'_t \pi'_{t-1} \pi'_{t-2}} + \cdots + \gamma^{t-1} \frac{\pi_t \pi_{t-1} \cdots \pi_0}{\pi'_t \pi'_{t-1} \cdots \pi'_0} \right) \right. \\ &\quad + \gamma e(t-1) \frac{\pi_t}{\pi'_t} \left(\frac{\pi_{t-1}}{\pi'_{t-1}} + \gamma \frac{\pi_{t-1} \pi_{t-2}}{\pi'_{t-1} \pi'_{t-2}} + \cdots + \gamma^{t-2} \frac{\pi_{t-1} \pi_{t-2} \cdots \pi_0}{\pi'_{t-1} \pi'_{t-2} \cdots \pi'_0} \right) \\ &\quad + \gamma e(t-2) \frac{\pi_t \pi_{t-1}}{\pi'_t \pi'_{t-1}} \left(\frac{\pi_{t-2}}{\pi'_{t-2}} + \gamma \frac{\pi_{t-2} \pi_{t-3}}{\pi'_{t-2} \pi'_{t-3}} + \cdots + \gamma^{t-3} \frac{\pi_{t-2} \pi_{t-3} \cdots \pi_0}{\pi'_{t-2} \pi'_{t-3} \cdots \pi'_0} \right) \\ &\quad \left. + \cdots + \gamma^{t-1} e(0) \frac{\pi_t \pi_{t-1} \cdots \pi_1}{\pi'_t \pi'_{t-1} \cdots \pi'_1} \left(\frac{\pi_0}{\pi'_0} \right) \right] \\ &= (r_t - b)\left[\frac{\pi_t}{\pi'_t} e(t) + \gamma \frac{\pi_t \pi_{t-1}}{\pi'_t \pi'_{t-1}} (e(t) + e(t-1)) + \gamma^2 \frac{\pi_t \pi_{t-1} \pi_{t-2}}{\pi'_t \pi'_{t-1} \pi'_{t-2}} (e(t) + e(t-1) + e(t-2)) \right. \end{aligned}$$

$$+\cdots+\gamma^{t-1}\frac{\pi_t\pi_{t-1}\cdots\pi_0}{\pi'_t\pi'_{t-1}\cdots\pi'_0}(e(t)+e(t-1)+\cdots+e(0))\Bigg] \tag{10}$$

By Equation 6 and 10, the sum of all backups is given by

$$\begin{aligned}\sum_{t=0}^{T}\Delta\theta_t &= (r_0-b)\frac{\pi_0}{\pi'_0}e_0+\cdots+\gamma^{T-1}(r_T-b)\frac{\pi_0\pi_1\cdots\pi_T}{\pi'_0\pi'_1\cdots\pi'_T}(e_0+\cdots+e_T)\\ &\quad+(r_1-b)\frac{\pi_1}{\pi'_1}e_1+\cdots+\gamma^{T-2}(r_T-b)\frac{\pi_1\pi_2\cdots\pi_T}{\pi'_1\pi'_2\cdots\pi'_T}(e_1+\cdots+e_T)\\ &\quad+\cdots+(r_T-b)\frac{\pi_T}{\pi'_T}e_T=\sum_{t=0}^{T}\sum_{k=t}^{T}\gamma^k(r_k-b)\frac{\pi_t\pi_{t+1}\cdots\pi_k}{\pi'_t\pi'_{t+1}\cdots\pi'_k}(e_t+e_{t+1}+\cdots+e_k)\\ &= \sum_{t=0}^{T}\sum_{k=t}^{T}(r_k-b)\frac{\gamma^k}{\pi'_t\pi'_{t+1}\cdots\pi'_k}\frac{\partial}{\partial\theta}(\pi_t\pi_{t+1}\cdots\pi_k)=\sum_{t=0}^{T}\frac{\partial}{\partial\theta}V_t\end{aligned} \tag{11}$$

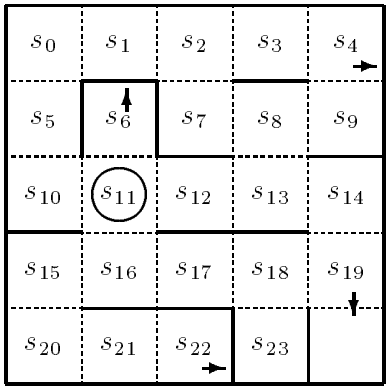
The last transformation of Equation 11 is obtained by the derivative of Equation 1 with respect to θ. Equation 5 and 11 derive the following results:

$$\mathrm{E}\left\{\lim_{T\to\infty}\sum_{t=0}^{T}\Delta\theta_t\right\}=\sum_{t=0}^{T}\mathrm{E}\left\{\lim_{T\to\infty}\sum_{t=0}^{T}\frac{\partial}{\partial\theta}V_t\right\}=\sum_{t=0}^{T}\frac{\partial}{\partial\theta}V^{\pi}(s_t) \tag{12}$$

It is shown that the target policy parameter can be improved towards the sum of the gradient of the value function on each time step. Note that this result does not depend on the Markov property, therefore it can be extended to non-Markovian environments.

4 Experiments

4.1 Maze World: A MDP Environment



state S : 24 states
action A : To move North, East, South or West (4 actions)

The agent cannot move across the wall.
When the agent enters any goal,
the agent jumps all states with uniform distribution.
Reward $R^a(s,s')=100$ if the agent enters the goal,
or 0 otherwise.
The symbol O denotes the agent.
The current state is, $s_t=s_{11}$.

Figure 2: A maze problem with four tasks. Task 0: Goal if South is selected at s_{19}, Task 1: Goal if East is selected at s_{22}, Task 2: Goal if East is selected at s_4, Task 3: Goal if North is selected at s_6.

The algorithm is demonstrated in a maze world with four tasks as shown in Figure 2. In the importance sampling by switching policies, the agent changes its behavior policy for Task 0 (policy 0), Task 1 (policy 1), Task 2 (policy 2) or Task 3 (policy 3) in turn at intervals of 500 steps. In the importance sampling by a random policy, the agent keeps its initial behavior policy. In the importance sampling methods, the learning for all policies are concurrently executed sharing the same experience under

the behavior policy. It is compared with a simple policy gradient method, in which the learning is executed only in the target policy that equals to the behavior policy. That is, the policy 0 in the simple gradient method is updated by using the data of only 500 steps whereas the agent performs 2000 steps. The simple policy gradient method can be easily arranged by modifying our algorithm to $\pi = \pi'$ for each task. The agent observes the position in the maze (24 states), and selects one action from North, East, South, or West. When the agent enters one of the goal, the only corresponding policy receives positive reward $r_t = 100$, or $r_t = 0$ for other policies, and then the agent jumps next state with uniform distribution.

The agent calculates action-probability of policies in the current state s by:

$$P^j(a_i|s) = \frac{\exp(\theta_i^j(s))}{\exp(\theta_0^j(s)) + \exp(\theta_1^j(s)) + \exp(\theta_2^j(s)) + \exp(\theta_3^j(s))} ,$$

where $P^j(a_i|s)$ denotes the probability of selecting action a_i in state s following the policy π^j, and $\theta_i^j(s)$ is the policy parameter of π^j. If the action a_i is selected, the eligibility $e_k^j(s)$ for the policy parameter $\theta_i^j(s)$ is given by

$$e_k^j(s) = \frac{\partial}{\partial \theta_k^j(s)} \ln P^j(a_i|s) = \begin{cases} 1 - P^j(a_i|s) & \text{if } k = i \\ -P^j(a_i|s) & \text{if } k \neq i \end{cases}$$

We adopted this scheme to the calculation of π_t, π'_t and $e(t)$ in Figure 1.

The simulation results are shown in Figure 3. The discount factor γ equals 0.9 and learning rate α equals 0.01 for both algorithms. The learning performance of the proposed method is about two or three times as good as that of the standard gradient method, except for Task 1. The reason is that since the goal of Task 1 is located in the depth of the maze, the number of visiting to the goal of Task 1 tends to small.

4.2 Locomotion Tasks on a Two-link Arm Robot: Applying Continuous Action Space

We applied the algorithm to a robot shown in Figure 4. The objective of learning is to find control rules to move forward (Task 0) or backward (Task 1). The joints are controlled by servo motors that react to angular-position commands. At each time step, the agent observes angular-position of two motors and a touch sensor attached to the leg top according to (ϕ_1, ϕ_2, ϕ_3), where each component is normalized between $0 \leq \phi_1, \phi_2 \leq 1$, and selects action according to the behavior policy. The angular-position ϕ_1, ϕ_2 is defined in two dimensional continuous space. The touch sensor ϕ_3 takes only 0 or 1. The action is a destination of angular-position. Let a vector $(a_{(1)}, a_{(2)})$ be the destination of angular positions of two-motors as an action output. It is also defined in two dimensional continuous space. When the agent select action, the the motors move towards the commanded positions. When the joint-angles move to the commanded position, or changing the sensor variables, then the reward is given as the result of the transition, and the time step proceeds to the next step. When the sensor variable changes in the way of moving joint-motors, the angular-position would not correspond to the destination position. For this reason, there exists uncertainty of the state transition. For Task 0, the immediate reward is the distance of the body movement caused by the previous action. When the robot moves backward, the policy for Task 0 receives negative reward. For Task 1, all rewards are simply given by inverting the sign of the reward of Task 0. The function of π and π' are given by Gaussian: The action $a_{(i)}$ for the motor (i) is selected following normal distribution $N(\mu_i, \sigma_i)$, where

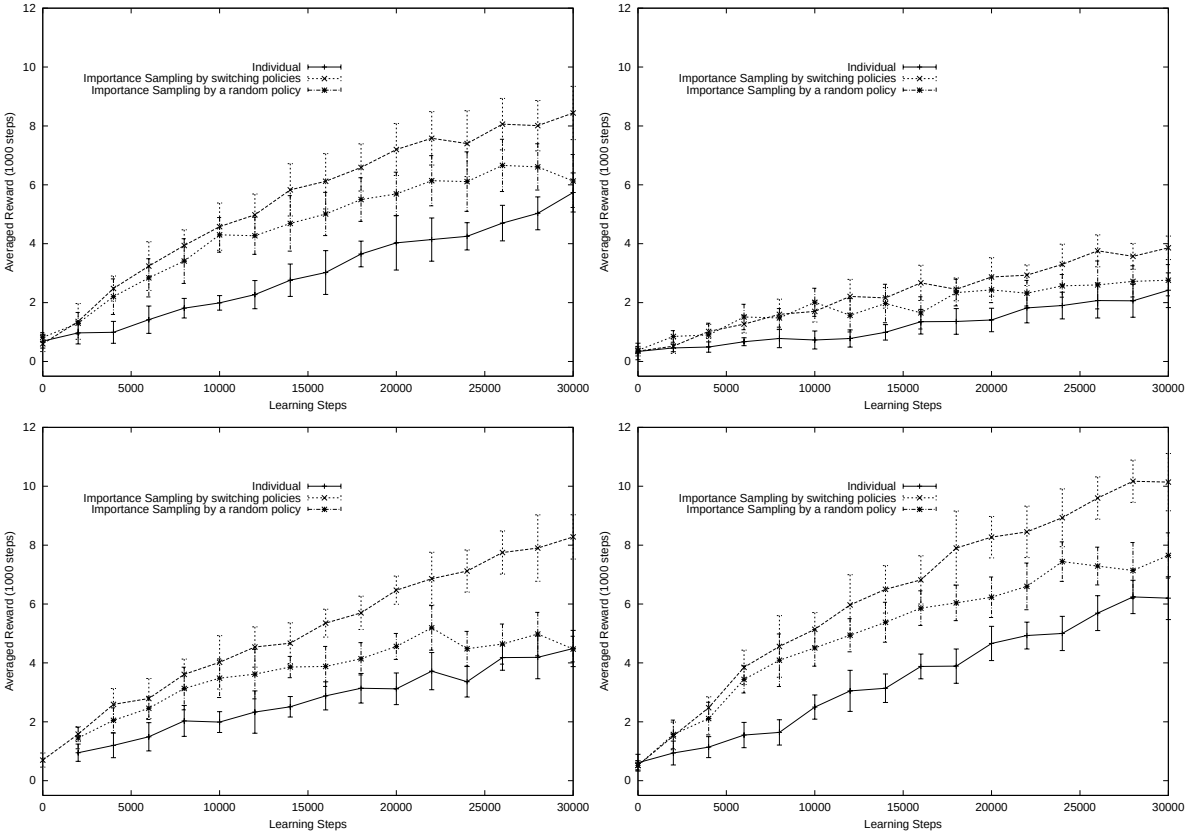


Figure 3: The performance of the algorithm compared with a standard gradient method in the maze world, averaged over 10 trials. The errorbar denotes the standard deviation. The top left shows Task 0, top right is Task 1, bottom left is Task 2, bottom right is Task 3.

$\mu_i = 1/(1+\exp(-\sum_k x_k\theta_{i,k})$ and $\sigma_i = 1/(1+\exp(-\theta_i)$. In this experiment, we tried two ways of generating the feature vector X: One is broad state generalization that takes $X = (x_1, \ldots x_6) = (\phi_1, \phi_2, \phi_3, 1-\phi_1, 1-\phi_2, 1-\phi_3)$. The other is a fixed-grid approximation, which consists of 8x8x2 boxels over the state space. The details of the calculation for π, π' and $e(t)$ are mostly omitted here for the reason of the limited space, but it is the same as the implementation for a four-legged robot in our works [2]. In this experiment, we use the same parameters in all the algorithms, $\gamma = 0.9$ and $\alpha = 0.02$.

Figure 5 and 6 show learning curves for the robot on the task 0 and 1. The importance sampling methods are too bad and quite unstable with using broad state generalization. Using the grid approximation, the proposed methods outperformed than the standard gradient method, however, the acquired control rules are inferior to those of the standard method with using broad state generalization.

5 Discussion and Future Work

We propose a policy gradient method combining with importance sampling. It enables off-policy policy improvement under the environments that are continuing (non-episodic) tasks, and/or continuous-action space. The effect of sharing the experiences among different policies is clearly observed in the maze problem. However, it is weak when the the agent adopts broad state generalization as the state representation. The reason is that since the state occupancy destribution is different between off-policy and on-policy, the off-policy learning with broad state generalization tends to spoil good

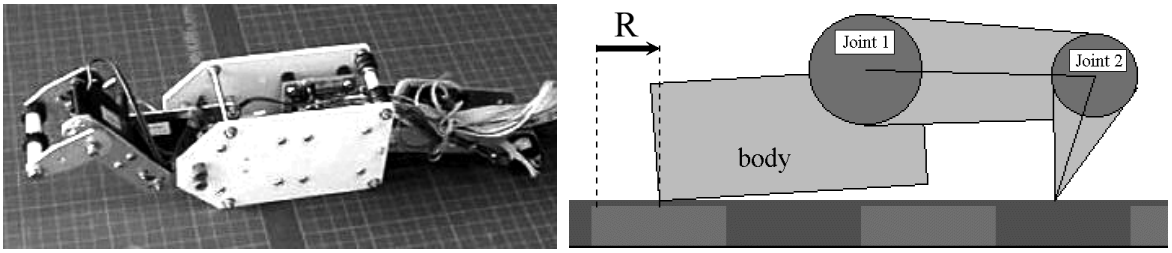


Figure 4: The left is a real two-link robot. The right-side shows a simulator of the robot used in this experiment. The state-action space is continuous.

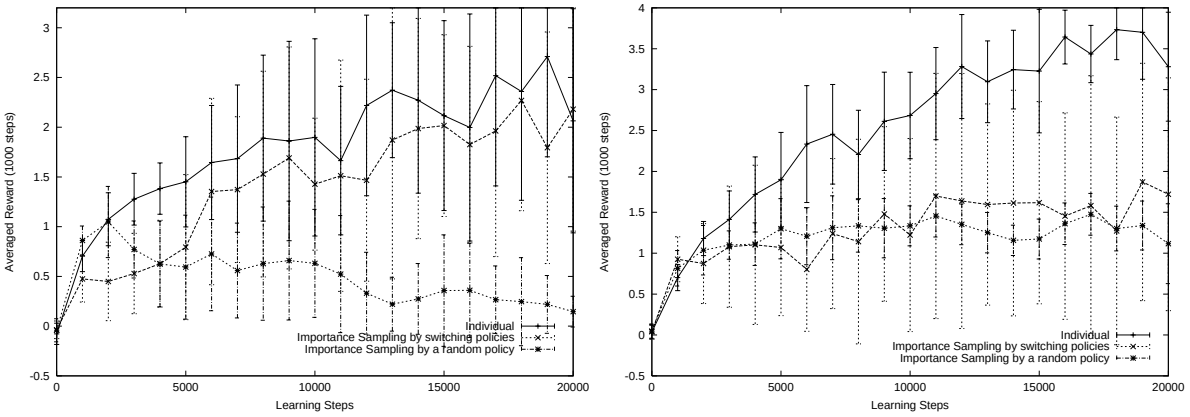


Figure 5: Learning curves of the algorithms using broad state generalization averaged over 10 trials on the robot simulator. The left is Task 0, the right is Task1. The importance sampling methods are inferior to the standard gradient methods.

acton-distribution around on-policy states. One way to avoid this undesirability is to use localized state representation such as grid approximation methods. However, the grid approximation methods cause problems arising from curse of dimensionality. Designing appropriate state representation for importance sampling policy improvement methods in the high-dimensional state space is challenging future work. We are trying to extend our method to an actor-critic algorithm that estimates state value for each target policy and makes use of the value for the policy improvement. We also intend to apply the algorithm to a real one leg to four leg robots as future work.

References

[1] Kimura, H. & Kobayashi, S.: An Analysis of Actor/Critic Algorithms using Eligibility Traces: Reinforcement Learning with Imperfect Value Function, Proc. of 15th International Conference on Machine Learning, pp.278–286 (1998).

[2] Kimura, H., Yamashita, T. & Kobayashi, S.: Reinforcement Learning of Walking Behavior for a Four-Legged Robot, 40th IEEE Conference on Decision and Control (CDC2001), pp.411–416 (2001).

[3] Peshkin, L. & Shelton, C.R.: Learning from Scarce Experience, Proc. of 19th International Conference on Machine Learning (ICML2002), pp.498–505.

[4] Precup, D., Sutton, R.S. & Singh, S.: Eligibility Traces for Off-Policy Policy Evaluation, Proc. of the 17th International Conference on Machine Learning, pp.759–766 (2000).

[5] Shelton, C.R.: Policy Improvement for POMDPs using Normalized Importance

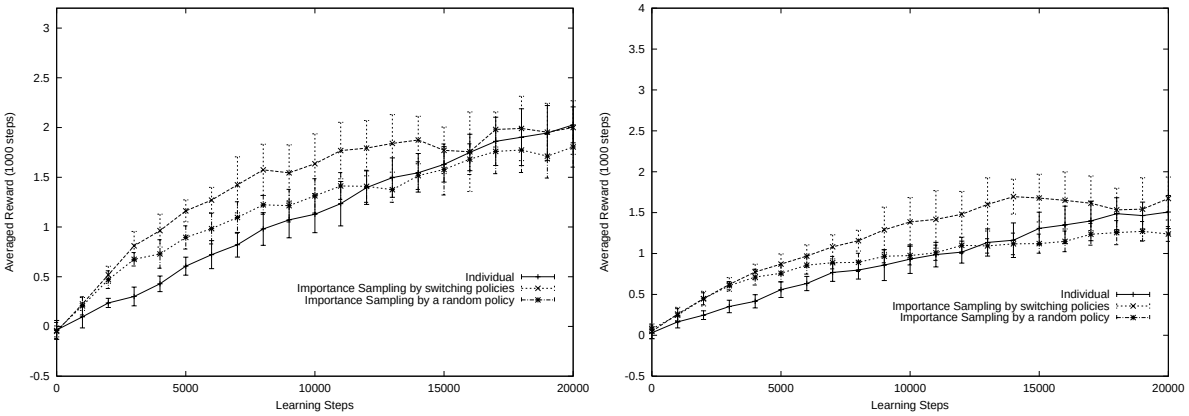


Figure 6: Learning curves of the algorithms using the grid state approximation (8x8x2) averaged over 10 trials on the robot simulator. The left is Task 0, the right is Task1. The importance sampling methods are outperformed than the standard gradient methods.

Sampling, Proc. of the 17th Conference on Uncertainty in Artificial Intelligence, (UAI2001) pp.496–503 (2001).

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

[7] Sutton, R.S., McAllester, D., Singh, S. & Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation, Advances in Neural Information Processing Systems 12 (NIPS12), pp. 1057–1063 (2000).

[8] Uchibe, E. & Doya, K.: Reinforcement Learning using Importance Sampling for Multiple Learning Modules, Technical report of the Institute of Electronics, Information and Communication Engineerings (2003, in Japanese).

[9] Williams, R.J.: Simple Statistical Gradient Following Algorithms for Connectionist Reinforcement Learning, Machine Learning 8, pp. 229–256 (1992).