The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

A Reward Allocation Method for Reinforcement Learning in Stabilizing Control Tasks

Shu Hosokawa[1], Joji Kato[2], and Kazushi Nakano[12]

[1] Dept. of Electronic Eng. , The University of Electro-Communications, Chofu, Tokyo 182-8585, Japan
[2] Dept. of Mechanical Eng. and Intelligent Systems, The University of Electro-Communications, Chofu, Tokyo 182-8585, Japan
(Tel : 81-42-443-5190; Fax : 81-42-443-5183)
(hosokawa@francis.ee.uec.ac.jp)

Abstract: Reinforcement learning is a machine learning method that does not require a detailed teaching signal by a human, which is expected to be applied to real robots. In its application to real robots, the learning processes are required to be finished in a short learning period of time. A reinforcement learning method of non-bootstrap type has fast convergence speeds in the tasks such as Sutton's maze problem that aim to reach the target state in a minimum time. However, these methods are difficult to learn either task to keep a stable state as long as possible. In this study, we improve the reward allocation method for stabilizing control tasks. The validity of our method is demonstrated through simulation for stabilizing control of an inverted pendulum. Our proposed method can acquire a policy in order to keep a stable state within a short learning period of time.

Keywords: Reinforcement Learning, Stabilizing Control Tasks, Profit Sharing, SMDP

1 INTRODUCTION

The machine learning methods have attracted a lot of attention by a characteristic that robot's adaptive actions are able to get from action results of itself. Reinforcement learning is a method of machine learning, which does not require a detailed teaching signals by a human. Reinforcement learning can be divided into two types of bootstrap and non-bootstrap. As well-known non-bootstrap methods, there are Profit sharing [1] , Adaptive immunity - based reinforcement learning [2] and so on. Those types of learning methods are faster than famous reinforcement learning methods such as Q-learning [3] in learning convergence speeds, when applying to the problem of minimizing the amount of time to achieve the goal (e.g., maze searching, swing-up control for an invert pendulum and so on). Those methods are not able to get the optimal policy, but they can get the policy that achieves to the goal state by satisfying the rationality theorem proposed by Miyazaki et al [4].

However, when applying that method to stabilizing control tasks, we cannot acquire a policy to achieve the goals. The bootstrap methods can acquire the policy of the stabilizing control by giving a negative reward at a change from a stable state to an unstable state [5]. On the other hand, since the non-bootstrap learning method cannot deal with negative a reward value, the reward value has to take a positive value [6]. In this case, there is a great risk of learning an undesirable behavior of changing from a stable state to an unstable state according to reward values.

In this study, we improve the reward allocation method for the stabilizing control tasks. In the stabilizing control tasks, we need to model an environment by Semi-Markov decision process (SMDP) because a time of doing actions is not constant by each state. The validity of our method is demonstrated through simulation for stabilizing control of an inverted pendulum.

2 PROFIT SHARING

Reinforcement learning method is for learning by updating to the Q-value that is an estimated value of action in a state. The reinforcement learning method of non-bootstrap type does not require Q-values of other states in Q-value updating process. The following equation shows how to update the Q-values of the Profit Sharing.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r(t) - Q(s_t, a_t) \right] \tag{1}$$

where t is the time, s_t is the state in time t, a_t is the selected action in time t, $\alpha (0 < \alpha < 1)$ is the learning rate and $r(t)$ is the reward function. The Q-value is updated based on the reward function of the actions that has been selected before receiving a reward. Following equation is generally to use as a reward function that satisfied the rational theorem [4], when applying to the problem of minimizing the amount of time to achieve the goal.

$$r(t) = R \left(\frac{1}{S} \right)^{T-t} \tag{2}$$

where, $S (\geq L + 1)$ is the constant value, L is the number of selectable actions, T is the episode time and R is received reward.

In many cases, action selection methods in Profit Sharing use the roulette selection, which is based on the probability

$P(s,a)$ with weight of the Q-values.

$$P(s,a) = \frac{Q(s,a)}{\sum_{a \in A} Q(s,a)} \tag{3}$$

Here, since Eq. (3) cannot take zero or negative Q-values, the Q-values must be greater than zero in updating process.

3 REWARD FUNCTIONS IN STABILIZING CONTROL TASKS

In this section, we discuss about the method of non-bootstrap reinforcement learning for applying it to the stabilizing control tasks. We considered about conditions of the reward allocations for obtaining the policy of the stabilizing control tasks. In addition, we propose a reward allocation method by use of those conditions.

3.1 Semi-Markov Decision Process

The problem of minimizing amount of time to achieve the goal dose not need to deal with the actions of keeping same state. On the other hand, there are no problems that deal with actions of keeping the same state in the stabilizing control tasks. However, Markov decision process (MDP) must to select an action at the each time of the constant control cycles. If the state is divided roughly, each action selection may include multiple effective action or illegal actions in the same state. It can be resolved by dividing the state into smaller pieces, but it requires longer time to learning.

In this study, we model an environment by SMDP. SMDP selects an action after the state transition has occurred. It cannot completely resolve the trade-off for the state space division, but there is no need to handle the autoregressive action to the same state. Therefore, it is possible to make a relatively rough state division.

3.2 Reward allocation

The non-bootstrap method updates all estimate values of state and action when receive a reward. Therefore, we can use the duration of actions in the episode unlike bootstrap methods such as Q-learning. However, if using the duration of episode as the reward, we have a very large reward. Overflow may occur when performing the actual calculation, or if the estimated value has a very small initial one, there is a risk that state solution search is not performed sufficiently. Therefore, we give the reward of constant value (R=1), which is allocated from duration time of action. In addition, we assume that each episode starts from the stable region, and discuss about a reward allocation method in stabilizing control tasks by using an example shown in Fig. 1.

First of all, we give an example of the state transition in Fig. 1.

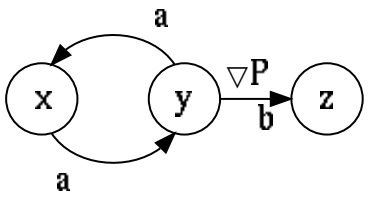

Fig. 1. An example of state transition

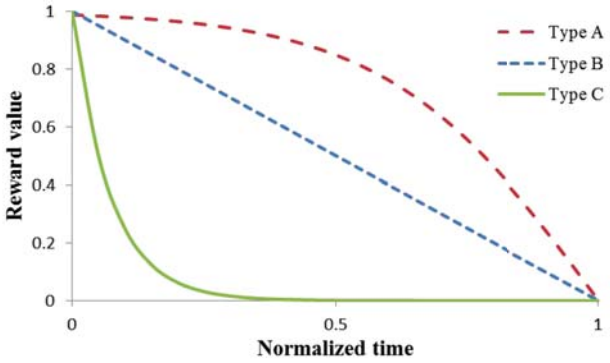

Fig. 2. An example of reward allocation

State transition example(1)

$x_a \rightarrow y_b \rightarrow z$

State transition example(2)

$x_a \rightarrow y_a \rightarrow x_a \rightarrow y_b \rightarrow z$

where x and y are the stable states, and z is the unstable state their subscripts is show selected actions. The above example means to suppress the action b of under going a transition from the state y to the state z. For the stabilizing control tasks, the reward allocation value should be zero or small value because action taken just before receiving the reward should not be selected. On the other hand, the reward allocation value may be higher values because actions taken long before receiving the reward one useful for stabilization. Therefore, the reward of actions allocates higher value around the initial time (t_a), and does lower values around the episode end time (t_b).

$$r(t_a) > r(t_b)$$

Figure 2 shows three types of functions. The horizontal axis in this figure shows the normalized time duration episode, the vertical axis does the value of reward allocation. In initial learning is short, it is better to reduce the action selections for changing to unstable state as soon as possible. We want to assign a small reward for it. The type C function is not suitable from the above view. And also, the episode duration time becomes longer on the way of learning. In this

case, since it may also contribute on the way of learning, it is better allocate the reward equivalent to that in the initial episode. The type A functions allocates a small reward to actions just before Stabilization state end, and dose a small discounted reward to actions in the initial-middle stage of the episode. The following equation is an example of the type A functions shown in Fig. 2.

$$r(t) = 2\left(0.5 - \frac{1}{1+\exp(-(t-1)/T_a)}\right) \tag{4}$$

where t is the normalized time by episode time and T_a is the constant value for decide to functions gradient.

As the duration of episode becomes to be longer with learning, action selection is to be performed multiple times in the same state. However, the type A function cannot make an evaluation of difference between actions for moving to unstable state and others. Because each evaluation is decided by normalized time of action, and each value has only small difference. Here, we consider about the number of selected actions when transitions among some states are undergone periodically. We set that the action for keeping stable state is a_{loop}, and the action for moving to unstable state is a_{open}. It is clear that the number of action selection for keeping stable state is longer than that of the action for moving to unstable state ($Num(a_{loop}) \geq Num(a_{open})$, $Num()$ is action selected numbers). And also, the actions for keeping stable state are made earlier than the actions for moving to unstable state. From those considerations, we can make a difference in the update process by use of the sum of the reward. Therefore, the update formula of Profit Sharing method for stabilizing control task is shown as.

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha \left[\sum_t r(t|s_i, a_i) - Q(s_i, a_i)\right] \tag{5}$$

The reward function $r(t)$ is defined in the range $[0, 1)$, the initial Q-values have to set 1 and more. This provides a searching in the environments is performed well by Boltzmann and roulette selection.

4 SIMULATION

This section shows a simulation result of our proposed method applying to stabilizing control of the inverted pendulum (Fig. 3). The motion equation of the inverted pendulum is expressed as

$$(M+m)\ddot{x} + ml\cos\theta\ddot{\theta} + D_x\dot{x} + ml\sin\theta\dot{\theta} = a \tag{6}$$

$$-ml\cos\theta\ddot{x} + I\ddot{\theta} + D_\theta\dot{\theta} - mgl\sin\theta = 0 \tag{7}$$

where, M is the mass of the cart, m is the weight of the pendulum, l is the length of the pendulum to the gravity center,

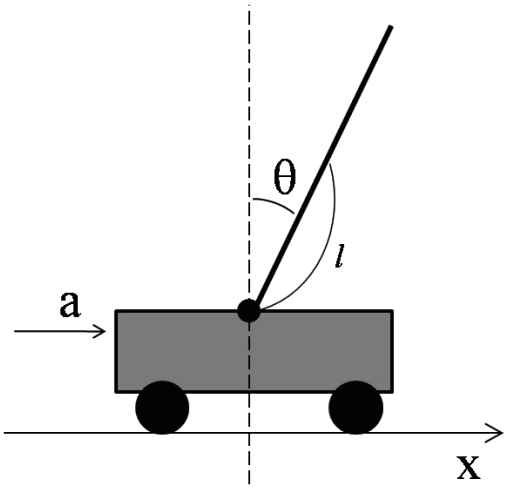

Fig. 3. Inverted pendulum

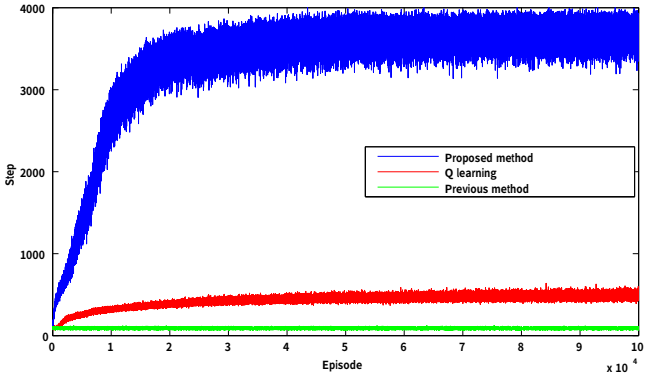

Fig. 4. Results of convergence speeds

D_x is the frictional coefficient on the cart, D_θ is the friction of the pendulum rotation, I is the moment of inertia of the pendulum. The states $x, \dot{x}, \theta$ and $\dot{\theta}$ are inputted to the learning module, whose values are observed with the normalized random values ($N(\mu, \sigma^2), \mu = 0, \sigma = 0.0001$) added. And also, we divide the states as 10 for x, 10 for $\dot{x}$, 50 for θ and 50 for $\dot{\theta}$. In this simulation, we perform a learning of the stabilizing control of the inverted pendulum according to action list $\boldsymbol{A} = [-10, -1, -0.1, 0, -0.1, 1, 10]$. If the pendulum is not around the upright position ($|\theta| > 0.1$[rad]), or the cart position is out of the range ($|x| > 3$[m]), we give the reward, and restart and proceed to the next episode. Also, if we can reach to 4,000 simulation steps, we deemed that it was possible to get the stabilizing control policy; we give the reward and end the episode.

Figure 4 shows the results of convergence speeds in learning by using our proposed reward allocation method, the previous reward allocation method (Eq. (2)) and the Q-learning

method. In the case of the Q-learning, we give the penalty reward ($R = -10$), when the pendulum is not around upright position, or the cart position is out of the range. The previous reward allocation method was not able to give the policy of the stabilizing control. It was able to give the policy that the pendulum fall down as soon as possible instead. On the other hand, the Q-learning was able to give the stabilizing control policy, but only in a short time this success was made. Our method was able to produce the stabilizing control policy with a shorter convergence time than other methods.

5 CONCLUSION

In this paper, we improved the reward allocation method in the non-bootstrap reinforcement learning through the stabilizing control tasks. We showed the conditions of reward allocation for the stabilizing control tasks, and introduced an example of reward allocate function for it. Since the reward is allocated only from the duration time of action, we do not need to change the reward value according to environments.

In our future works, we are to build a learning method that can deal with combined problems such as swing-up and stabilizing controls of the inverted pendulum. And also, we have studied on the reinforcement learning method imitating the human adaptive immunity [2][7]. Those methods have fast convergence speed in the problems of minimizing the amount of time to achieve the goal with a very few configuration parameters. It is possible to construct a learning method with fast convergence speed by integrating those methods.

REFERENCES

[1] J. J. Grefenstette. Credit assignment in rule discovery systems based on genetic algorithms. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning*, pp. 524–534. Kaufmann, San Mateo, CA, 1988.

[2] Jungo Ito, Kazushi Nakano, Kazunori Sakurama, and Shu Hosokawa. Adaptive immunity based reinforcement learning. *Artificial Life and Robotics*, Vol. 13, No. 1, pp. 188–193, 2008.

[3] Christopher J. C. H. Watkins and Peter Dayan. Technical note: q-learning. *Mach. Learn.*, Vol. 8, No. 3-4, pp. 279–292, 1992.

[4] Miyazaki Kazuteru, Yamamura Masayuki, and Kobayashi Shigenobu. A theory of profit sharing in reinforcement learning. *Journal of Japanese Society for Artificial Intelligence*, Vol. 9, No. 4, pp. 580–587, 1994-07-01. (in japanese).

[5] Yu Zheng, Siwei Luo, and Ziang Lv. Control double inverted pendulum by reinforcement learning with double cmac network. In *Proceedings of the 18th International Conference on Pattern Recognition*, Vol. 04 of *ICPR '06*, 2006.

[6] Atsushi Suzuki, Tohgoroh Matui, and Hirohisa Seki. Profit sharing considering penalty. *The 17th Annual Conference of the Japanese Society for Artifical Intelligence*, pp. 3F4–02, 2003. (in japanese).

[7] Shu Hosokawa, Kazushi Nakano, and Kazunori Sakurama. A consideration of human immunity-based reinforcement learning with continuous states. *Artifical Life and Robotics*, Vol. 15, No. 4, pp. 560–564, 2010.