Optimization of Learning Cycles in Online Reinforcement Learning Systems

Akira Notsu
Graduate School of Humanities and Sustainable System Sciences
Osaka Prefecture University
Sakai, Osaka 599-8531, Japan
Email: notsu@cs.osakafu-u.ac.jp

Koji Yasuda, Seiki Ubukata and Katsuhiro Honda
Graduate School of Engineering
Osaka Prefecture University
Sakai, Osaka 599-8531, Japan
Email: {subukata, honda} @cs.osakafu-u.ac.jp

***Abstract*—In reinforcement learning, since the learning cycles are set by the designer, this drawback greatly affects the learning efficiency. In this research, we adaptively change the learning cycles of online reinforcement learning systems to acquire a necessary and sufficient set of states for them. We used a growing self-organizing map to estimate the state for fast learning speed. In conventional methods, a calculation is performed more often than necessary, but in our proposed method, calculations that are unnecessary for learning are minimized based on the state transition. We demonstrate that the proposed method finishes learning quickly by the inverted pendulum task and, based on our experiment results, find that an approach that adaptively shortens or lengthens the learning cycle is suitable for reinforcement learning.**

***Index Terms*—Growing self-organizing map, Reinforcement learning, Learning optimization, Online learning**

I. Introduction

Reinforcement learning [1] is a framework for obtaining optimum output by trial and error based on information obtained from the environment. In recent years, value estimation methods are producing better results than humans in such board games such as Go and Atari 2600 using Deep Neural Networks [2], [3]. In the approach that recognizes reinforcement learning as a value function optimization, since a large amount of state and behavior related information is stored and used for optimization, the computation time and required memory are increased. Considering that quantization [4] and online learning [5], [6] are also becoming important in neural network research, using such online algorithms is valuable even in reinforcement learning.

On the other hand, we use a growing self-organizing map (GSOM) to estimate the state to improve the learning speed and reduce the calculation amount. Since a growing self-organizing map can adapt information without breaking the learning result, it is possible to advance a search without breaking the previous reinforcement learning result.

Even though a learning cycle is set by the designer, this greatly affects the learning efficiency. When the learning cycle is shortened, the number of states becomes too large and the search efficiency drops. However, if it is too large, the learning agent cannot learn. In the function optimization approach, it is possible to deal with appropriately obtained information by deleting it, but many unnecessary calculations are required.

In this research, we adaptively change the learning cycle of online reinforcement learning systems to acquire a necessary and sufficient set of states for reinforcement learning. In online reinforcement learning, state estimation is performed by a growing self-organizing map. The timing of the growing is judged by the difference of the sensor information.

II. Reinforcement learning

Reinforcement learning is formulated based on the Markov decision process. We denote the set of possible states of the environment by $S = s_1, s_2, ..., s_n$, and the set of behaviors the agent can take is $A = a_1, a_2, ..., a_n$. In state $s \in S$ in the environment, when the agent executes action a, the environment transitions to state$s' \in S$ in a probabilistic manner. At this time, reward R is stochastically given from the environment to the agent. Decision making at each of the agent's time point is based on an estimation of an action's value in the state. Q value $Q(s_t, a_t)$, which evaluates the value of action a_t in state s_t at time t, is updated using the information given on the environment and reward R [7]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_{QL}\left[R + \gamma \max_{a' \in A(s_{t+1})} Q(s_{t+1}, a') - Q(s_t, a_t)\right], \tag{1}$$

where $0 < \alpha_{QL} < 1$ is the learning rate and $0 < \gamma < 1$ is the discount rate. In experiments described below, we used ε-greedy as a policy.

III. Growing self-organizing map

A self-organizing map is a learning algorithm for the quantization and visualization of information suggested by Kohonen [8]. Neurons naturally align by repeating learning in the algorithm procedure. The following procedure updates weight vector w_i of neuron i based on input vector I with learning rate α_i:

$$\begin{aligned} w_{i,t+1} &\leftarrow (1 - \alpha_i)w_{i,t} + \alpha_i I, \\ \alpha_i &= \alpha_{som} \exp(-\frac{\|r_{win} - r_i\|^2}{c}), \end{aligned} \tag{2}$$

where $\|r_{win} - r_i\|^2$ is the square distance on the network between the winning neuron and neuron i, α_{som} is the learning rate, and c is a parameter that indicates the spread of the neighborhood. With this algorithm, since each neuron represents the prototype of a part of the data, we can quantize it and cluster the data.

In a growing self-organizing map, a neuron is newly added when the difference between its weight vector and the input exceeds a threshold value [9]. This accommodates new data without significantly changing the previous neuron learning. This is suitable for online learning. Growing self-organizing maps have been studied, such as adding rows and columns of two-dimensional neurons [10], extensions to hypercubes [11], or with a hierarchical structure [12].

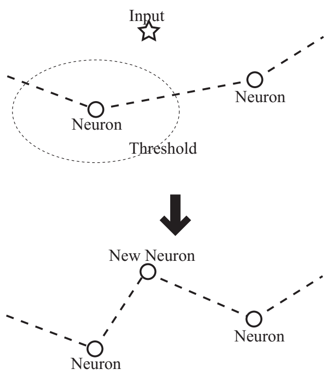

Fig. 1. Growing self-organizing map

We used a one-dimensional growing self-organizing map to estimate the state in reinforcement learning [13]. Each neuron is regarded as one state. Our previous research confirmed that a one-dimensional SOM is suitable for the reinforcement learning process. If the state is not given directly (but indirectly) by sensor information, however, updating the learning cycle becomes a big problem. If the information is too fine, the estimated number of states increases exponentially. Conversely, if it is too large, the obtained information becomes too rough, and it is impossible to perform good control.

IV. Proposed method

We adaptively changed the learning cycle. When the learning is successful, the learning cycle is not changed; if the learning fails, it is changed. Strictly speaking, a more accurate estimation method of a batch type can be conceived, but this time our main purpose is to improve online reinforcement learning.

We have two proposed methods: decreasing a large learning cycle and increasing a small learning cycle. If the learning is not good in the trial, we multiply the learning cycle by β. Specifically, an inflation approach increases the cycle when the highest reward is not updated and a deflation approach reduces it when the highest reward is less than a minimum goal and is updated.

The following is the procedure of the proposed method: :

Proposed algorithm

```
1)  Initialize each variable;
2)  Let the first input vector be the weight vector of the first neuron and the first winning neuron.
3)  repeat
4)      Determine an action based on the Q values of the winning neuron.
5)      Output an action.
6)      Observe the next input vector and reward.
7)      Determine the next winning neuron based on the input, SOM, and the differential. If the input exceeds the threshold value, then generate a new neuron that becomes the next winning neuron.
8)      Update SOM, the Q value, and the differential.
9)      If the obtained reward is not good, update the learning cycle
10) until the terminal condition.
```

In the proposed method, average input differential $D_{j,t}$ of the jth sensor input $I_{j,t}$ at time t is updated:

$$D_{j,t+1} \leftarrow \begin{cases} D_{j,t}, & \text{if } |I_{j,t+1} - I_{j,t}| < \delta D_{j,t} \\ |I_{j,t+1} - I_{j,t}|, & \text{else if } D_{j,t} = D_{j,0} \\ (1-\alpha_D) D_{j,t} + \alpha_D |I_{j,t+1} - I_{j,t}|, & \text{else} \end{cases}, \quad (3)$$

where α_D is the learning rate and δ is a threshold parameter. The first case in the updated formula eliminates the influence of noise and no state transition.

The distance between a weight vector of neurons $w_t = (w_{1,t}, w_{2,t}, \ldots, w_{n,t})$ and input data $I_t = (I_{1,t}, I_{2,t}, \ldots, I_{n,t})$ is normalized:

$$||I_t - w_t||^2 = \sum_j \left(\frac{I_{j,t} - w_{j,t}}{D_{j,t}} \right)^2. \quad (4)$$

If the normalized distance exceeds the threshold value, then a new neuron is generated.

Another simple way to increase the number of neurons based on the mean distance is considered for comparison. In this method (GSOM-MD), neurons are added under the following conditions:

$$||I - w_{win}|| > \kappa \text{MD}, \quad (5)$$

where κ is a parameter and MD is the mean distance between neurons in SOM.

V. Numerical experiments and results

We simulated the swing-up problem of an inverted pendulum (Fig. 2). In this setting, a 1-m rod is connected to a motor and moves by the application of torque. The inputs are angle θ ($-\pi < \theta \leq \pi$ rad) and angular speed $\dot{\theta}$ ($-5\pi < \dot{\theta} \leq 5\pi$ rad/s), and the outputs are torques (T) -10, 0, and 10 Kgf·m.

The goal is to stop the rod by pointing it straight up by controlling the torque. The rod weighs 1.5 kg, and the torque is not large, so even continuous output in one direction of rotation will not make the rod vertical. Therefore, it must

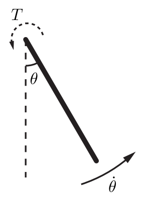

Fig. 2. Inverted pendulum

be swung to gain momentum. We simulated the inverted pendulum using the 4th-order Runge-Kutta method in 0.025-second increments and 5-second iterations. Time length (s) of $|\theta| \geq 0.98\pi$ is given as a reward.

The parameters are summarized in Table I. Agents chose an action according to the ε-greedy method. Initial Q value Q_0 is set to 2 to avoid local solutions. In addition to the proposed method (GSOM-DL), we compared the following states: agents with states given (GRID), states estimated by SOM, and states estimated by GSOM-MD. Agents with states given are written as GRID 40^2 and GRID 50^2 for state spaces partitioned into 40×40 and 50×50 discrete states, respectively.

First, to confirm that the learning is finished quickly by the proposed method, we experimented with a learning cycle set to 0.2 seconds. The result is shown in Fig. 3. The vertical axis represents the average inverted time for 100 trials. Only the proposed method finished learning within 8000 times.

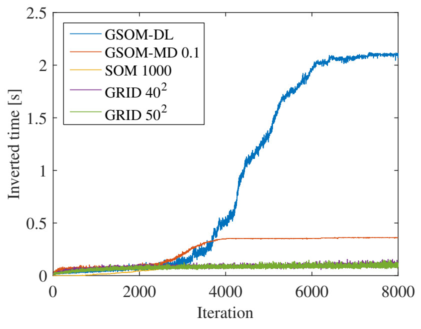

Fig. 3. Learning results

An example of the obtained control is shown in Fig. 4. The vertical axis is the angle, and the horizontal axis is the elapsed time. Once it goes up half way, it gains momentum, and is inverted.

The mean ($\pm$ SD) number of states constructed by our method is 666.65 ($\pm$ 39.47), which is much smaller than that of GRID 40^2, i.e., 1600. In Fig. 5, the state space based on the acquired self-organizing map is indicated by a Voronoi diagram. The horizontal axis shows the angular velocity, and the vertical axis shows the angle. The arrow indicates the direction of the learned behavior. By our method, the space is asymmetrically divided based on the estimation of the state transition. This is the reason why it learned efficiently.

Next we shown the results when the learning cycle is shortened and extended. Figs. 6 and 8 show the average inverted times when the learning cycle is 0.05 and 0.5, respectively. Figs. 7 and 9 are the acquired self-organizing maps.

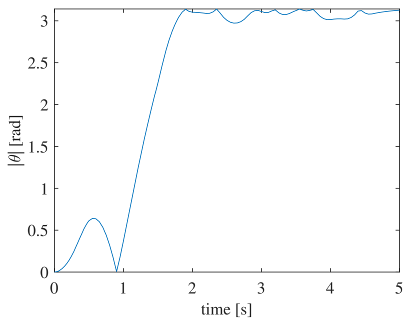

Fig. 4. Example of control

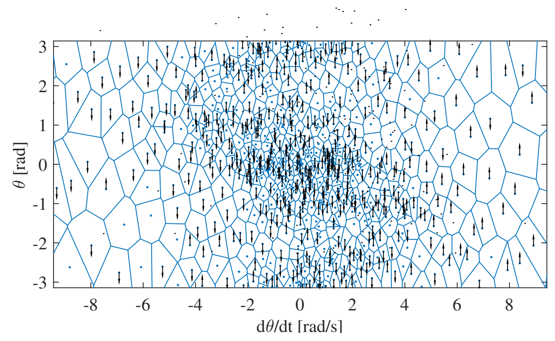

Fig. 5. Example of growing SOM

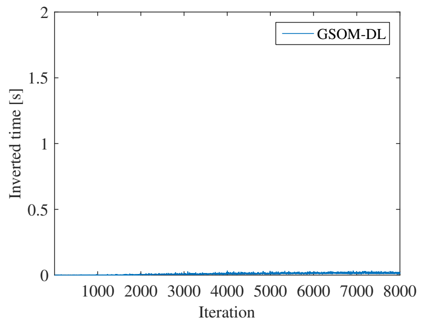

Fig. 6. Learning results in short cycle (learning cycle = 0.05)

In Fig. 6, the states are finely defined, and no learning has been completed at all. From Fig. 8, it is impossible to define a detailed state near the start or near the goal, and no delicate control can be performed.

Finally, we conducted an experiment on adaptively changing the learning cycle. A learning cycle method was set to 0.5, and if the highest reward was less than 1 (=a minimum goal reward) and updated in the trial, we multiplied the learning cycle by $\beta(< 1)$ (deflation approach) and the method of setting the learning cycle to 0.05. If the highest reward was not updated in the trial, we multiplied the learning cycle by

TABLE I
PARAMETERS

Parameters	Grid	SOM	GSOM-MD	GSOM-DL
α_{QL}	0.2	0.2	0.2	0.2
γ	0.9	0.9	0.9	0.9
ε	0.001	0.001	0.001	0.001
α_{SOM}	-	0.01, 0.9	0.01	0.01
c	-	0.00001, 1	0.00001, 1	0.00001, 1
κ	-	-	0.02, 0.1, 0.5, 0.9	0.9
α_D	-	-	-	0.99
δ	-	-	-	0.3
$D_{j,0}$	-	-	-	0.00001
N	40^2, 50^2	250, 500, 1000	Growing	Growing

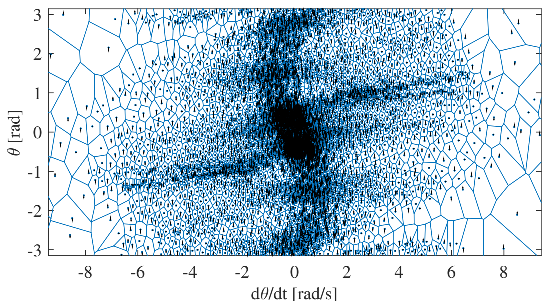

Fig. 7. Example of growing SOM in short cycle

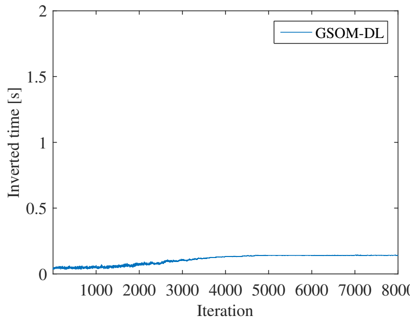

Fig. 8. Learning results in long cycle (learning cycle = 0.5)

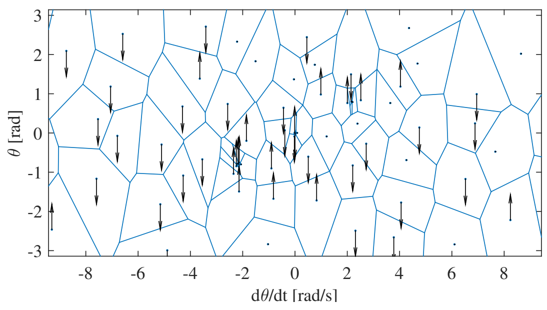

Fig. 9. Example of growing SOM in long cycle

$\beta(>1)$ (inflation approach). Both steps were simulated.

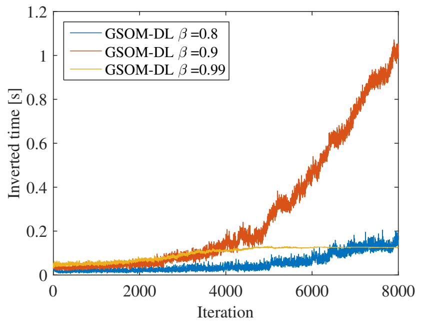

Fig. 10. Learning results in deflated long cycle

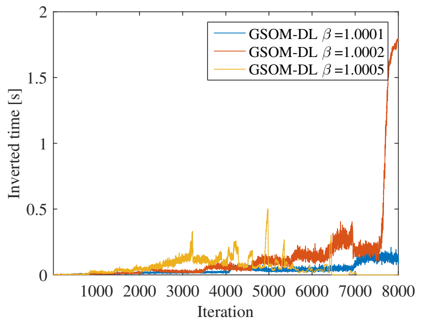

Fig. 11. Learning results in inflated short cycle

The results of the deflation and inflation approaches are shown in Figs. 12 and 13 as well as their inverted times. Figs. 12 and 13 are the acquired self-organizing maps.

Several deflation approaches have acquired state space, and the learning result is good and successful. On the other hand, several inflation approaches also succeeded within 8000 iterations. However, some learnings did not work very well, which reduced the average reward, also in the deflation approach which has best a parameter β.

As an approximate trend, if the inflation speed is too fast, learning fails. If the inflation or deflation speed is too slow, the learning speed also becomes slow.

As seen in Figs. 12 and 13, although the learning results were relatively similar, the densities of the obtained growing

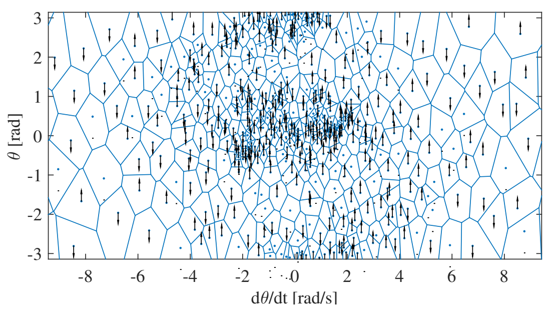

Fig. 12. Example of growing SOM in deflated long cycle

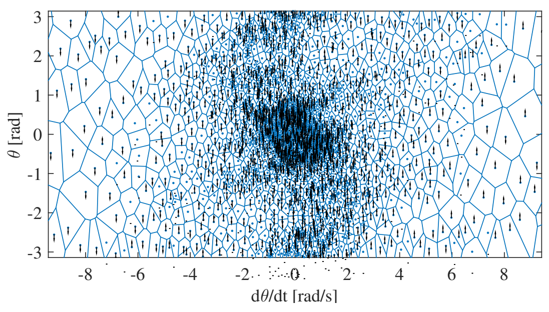

Fig. 13. Example of growing SOM in inflated short cycle

self-organizing maps were different. For learning a fine state, gradually increasing the learning cycle will produce stable results. On the other hand, a detailed condition can be learned from the recognition of rough states. The number of states is smaller for the latter approach and more economical.

VI. Conclusion

We introduced a method to acquire state space by a growing self-organizing map in reinforcement learning and experimented with a method that adaptively adjusts the learning cycles. We confirmed that the proposed method finishes learning quickly and also found that an approach that shortens or lengthens the learning cycle is applicable for reinforcement learning from our experiment results.

Since there is a time lag between getting a reward and the value estimation whether an action will be good for optimal control, it is difficult to judge whether it is learnable. We also examined a simple approach in this paper. Future research will discuss and test other optimization methods. Forgetting or ignoring useless states is also important for saving learning resources, particularly in the inflation approach. Furthermore, we will consider how to change the learning cycle in one trial and automatically allow finer online control.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP15K00344.

References

[1] R. S. Sutton, and A. G. Barto, Reinforcement learning: An introduction, Vol. 1, MIT press, Cambridge, 1998.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587), 2016, pp. 484-489.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540), 2015, pp. 529-533.

[4] M. Courbariaux, Y. Bengio, and J. David, Binaryconnect: Training deep neural networks with binary weights during propagations, Advances in neural information processing systems, 2015, pp. 3123-3131

[5] N. L. Roux, M. Schmidt, and F. R. Bach, A stochastic gradient method with an exponential convergence rate for finite training sets, Advances in Neural Information Processing Systems, 2012, pp. 2663-2671.

[6] R. Johnson, and T. Zhang, Accelerating stochastic gradient descent using pre-dictive variance reduction, Advances in neural information processing systems, 2013, pp. 315-323.

[7] C. J. Watkins, and P. Dayan, Q-learning, Machine learning 8 (3-4) (1992) pp. 279-292.

[8] T. Kohonen, The self-organizing map, Neurocomputing 21 (1), 1998, pp. 1-6.

[9] B. Fritzke, A growing neural gas network learns topologies, Advances in neural information processing systems, 1995, pp. 625-632.

[10] B. Fritzke, Growing grid-a self-organizing network with constant neighborhood range and adaptation strength, Neural processing letters 2 (5) 420, 1995, pp. 9-13.

[11] H.-U. Bauer, and T. Villmann, Growing a hypercubical output space in a self-organizing feature map, IEEE Transactions on Neural Networks, 8 (2) , 1997, pp. 218-226.

[12] M. Dittenbach, D. Merkl, and A. Rauber, The growing hierarchical self organizing map, Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on, Vol. 6, IEEE,2000, pp. 15-19.

[13] A. Notsu, Y. Hattori, S. Ubukata, and K. Honda, Visualization of learning process in state and action space using self-organizing maps, Journal of advanced computational intelligence and intelligent informatics, 20 (6) , 2016, pp. 983-991. doi:10.20965/jaciii.2016.p0983. URL http://ci.nii.ac.jp/naid/40020996925/en/