International Journal of Fuzzy Logic and Intelligent Systems, vol. 12, no. 1, March 2012, pp. 36-41
http://dx.doi.org/10.5391/IJFIS.2012.12.1.36 pISSN 1598-2645 eISSN 2093-744X

Automatic Adaptive Space Segmentation for Reinforcement Learning

Yuki Komori, Akira Notsu*, Katsuhiro Honda, and Hidetomo Ichihashi

Department of Computer Sciences and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, Osaka, 599-8531, JAPAN

Abstract

We tested a single pendulum simulation and observed the influence of several situation space segmentation types in reinforcement learning processes in order to propose a new adaptive automation for situation space segmentation. Its segmentation is performed by the Contraction Algorithm and the Cell Division Approach. Also, its automation is performed by "entropy," which is defined on action values' distributions. Simulation results were shown to demonstrate the influence and adaptability of the proposed method.

Keywords : Reinforcement learning, Space segmentation, Entropy.

1. Introduction

Reinforcement learning is a framework for a strategy to learn through trial and error according to rewards provided by the environment [1]. In reinforcement learning, if the purpose of study is set, the agent automatically determines the best way to achieve the goal.

Q-learning [2], [3] is a typical reinforced study method that targets discrete states and actions. Solving a problem that has a continuous state and a continuous action requires discrimination of the continuous state and continuous action. We introduced the Contraction Method [5], and the Segmentation and Integration Method [6], into Learning Residual Entropy [7]. The Contraction Method contracts a state. The Segmentation and Integration Method segments a state and integrates two adjacent states. Learning Residual Entropy shows the degree of actions' converging.

In this paper, we investigated these methods. We considered how to divide in order to find the optimum solution faster, the technique by which an agent can acquire the best approach to automatically divide states, and the headstand stabilization problem of the pendulum.

2. Q-learning

Q-learning is a well-known reinforcement learning technique that works by learning an action-value function that gives the expected utility of taking given action a in given state s and following a fixed policy thereafter. One advantage of Q-learning is that it can compare the expected utility Q of available actions without requiring a model of the environment (usually, the agent selects the action with the maximum expected utility). Expected utility Q is updated iteratively. For each state s from state set S, and for each action a from action set A, we update Q-value Q that depends on current state s of the agent, action a selected by the agent, the next state s' of the agent after taking action a, and reward r received by the agent after taking the action. An update is calculated by the following expression:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a' \in A(s')} Q(s',a') - Q(s,a) \right], \quad (1)$$

where α is the learning rate and γ is the discount factor. Once Q-learning has finished, the optimal policy and optimal value function have been found without continuously updating the policy during learning.

Q-learning algorithm:

```
procedure Q-learning
begin
  initialize Q-table Q, ∀s ∈ S, ∀a ∈ A;
  set initial state s0 and goal state sn;
  for cycle := 1 to MAXCYCLE do s := s0
    while s ≠ sn do
      a := ActionSelect(Q,s);
      r := GetReward(s,a);
      s' := GotoNextState(s,a);
      Q := UpdateQ(Q,s,a,trace);
      s := s' ;
    end
  end
end
```

3. Learning Residual Entropy

The timing of the change of how to divide state space

Manuscript received Feb. 28, 2012; revised Mar. 5, 2012; accepted Mar. 12, 2012

*Corresponding author: Akira Notsu (Phone: +81-72-254-9355, fax: +81-72-254-9915, E-mail: notsu@cs.osakafu-u.ac.jp)

ⓒ The Korean Institute of Intelligent Systems. All rights reserved.

It is the extended version of a paper awarded as a best presentation in ISIS2011.

influences long-term reward and the speed of learning. However, finding the appropriate timing is difficult. Therefore, "Learning residual entropy" was proposed [7]. It expresses the grade of learning processes. For each state S, learning residual entropy $I(S)$ is defined as follows from action choice probability $p(S, a)$.

$$I(S) = -\frac{1}{\log|a|}\sum_{a} p(S,a)\log(p(S,a)) \quad , \qquad (2)$$

where $p(S, a)$ is a probability of the internal state S and action a. Now, $|a|$ is a number of possible actions. $I(S)$ is an index of to what degree uncertainty of actions remains at each state. In an unlearned case, $I(S)$ = log $|a|$ / log $|a|$ = 1. In an advanced learning case, particular action tends to be chosen in some cases. Then the more learning processes there are, the more $I(S)$ approaches 0.

In addition, the average of $I(S)$ in every state which is included in episode E at every episode and is called "Average of learning residual entropy" I, is as follows:

$$I = \frac{1}{|E|}\sum_{S\in E} I(S) \qquad (3)$$

Now, $|E|$ is a number of states included in episode E. By using the learning residual entropy, there is no guarantee of always getting appropriate timing. However, this can be obtained with only information from the environment, so this can become an extremely powerful index.

4. Single Pendulum Simulation

We simulated the single pendulum standing task (Figure 1). This physical calculation is performed by the Runge-Kutta 4th order method.

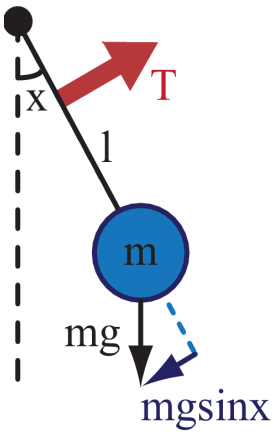


Fig. 1. Single pendulum

The following motion equation is simulated in increments of 0.05 seconds [4]:

$$m\ddot{x} = -mgl\sin x - k\dot{x} + T \qquad (4)$$

where the parameters and variables are defined as follows:

m = 1(kg)
l = 1(m)
k = 0
g = 9.8(m/s^2)
$-\pi < x \le \pi$
$-3\pi < \dot{x} \le 3\pi$
$-20 \le T \le 20$
1 iteration = 20 seconds
1 learning cycle = 0.2 seconds
$x_0 = \pi$, $\dot{x}_0 = 0$ in the initial state
reward $r_t = \frac{|x|}{\pi} - 0.5\frac{|T|}{20}$
learning rate α=0.25
discount factor γ=0.9
temperature τ = 1 at the first learning cycle, τ = 0.001 at the last learning cycle (linearly decreasing)
1000 episodes × 100
angle x and angular velocity $\dot{x}$ are segmented into 10 segments
torque T is segmented into six segments
x is segmented evenly, in the initial state
$\dot{x}$ and T are segmented evenly
contraction ratio is 0.05.

4.1 Contraction Method

We investigated an automatic contraction method through simulation [5].

```
procedure Q-learning with the contraction method
begin
  initialize Q-table Q, ∀s ∈ S, ∀a ∈ A;
  set initial state s0;
  for cycle := 1 to MAXCYCLE do s := s0
    for time := 0:0.05:20
    s´ := GetNextState(s,a);
      if mod(time,0.2)==0
        sr´ = s´ ;
        r := GetReward(sr´ ,a);
        Q := UpdateQ(Q,sr ,sr´ ,a);
        a := ActionSelect(Q,sr´ );
        sr= sr´ ;
      end
      s= s´ ;
    end
    if mod(cycle,FREQUENCY)==0
      (S,Q):=Contract(S,Q);
    end
  end
end
```

In this method, the space is initially evenly segmented, and contraction is performed at regular cycles by the Contract(S,Q) functions. The state in which the visited frequency is the most is contracted by the Contract(S,Q) function. Then, other states are expanded with the same ratio as before. Expected utilities are undisturbed. Each various episode was contracted. These

average rewards from 951 episodes to 1000 episodes are shown in TABLE I.

TABLE I
Frequency of contraction and average reward
(average of 100 trials)

Frequency of contraction	Average reward	Contraction count
every 5 episodes	54.6862	200
every 10 episodes	74.8555	100
every 15 episodes	76.5142	66
every 20 episodes	**76.9315**	**50**
every 25 episodes	76.1738	40
every 30 episodes	75.3698	33
no contraction	73.2671	0

As a result, "every 20 episodes" had the best reward.

4.2 Segmentation and Integration Method

We investigated an automatic segmentation and integration method through simulation[6].

```
procedure Q-learning with the segmentation and integration method
begin
  initialize Q-table Q, ∀s ∈ S, ∀a ∈ A;
  set initial state s0;
  for cycle := 1 to MAXCYCLE do s := s0
    for time := 0:0.05:20
    s´ := GetNextState(s,a);
      if mod(time,0.2)==0
        sr´ = s´ ;
        r := GetReward(sr´ ,a);
        Q := UpdateQ(Q,sr sr´ ,a);
        a := ActionSelect(Q,sr´ );
        sr= sr´ ;
      end
      s= s´ ;
    end
    if mod(cycle,FREQUENCY)==0
      (S,Q):=Combine(S,Q);
      (S,Q):=Divide(S,Q);
    end
  end
end
```

In this method, the space is initially evenly segmented, and segmentation and integration are performed at regular cycles by the Combine(S,Q) and Divide(S,Q) functions. The neighboring states in which an average of the visited frequency is the least are integrated into a single state that has an average expected utility of the states by the Combine(S,Q) function. The state in which the visited frequency is the most is equally segmented into two states that have the same expected utility by the Divide(S,Q) function. Other expected utilities are undisturbed.

In each various episode, the states were combined and divided. These average rewards from 951 episodes to 1000 episodes are shown in TABLE II.

TABLE II
Frequency of combining and dividing and the average reward
(average of 100 trials)

Frequency of combining and dividing	Average reward	Contraction count
every 50 episodes	74.0592	20
every 100 episodes	77.1982	10
every 200 episodes	**78.5920**	**5**
every 300 episodes	76.2746	3
every 400 episodes	75.6055	2
every 500 episodes	74.6342	2
no combining and dividing	73.2671	0

As a result, "every 200 episodes" had the best reward.

4.3 Comparison of "Contraction method" and the "Segmentation and Integration Method"

Learning curves of the "Contraction Method" and "Segmentation and Integration Method" are shown in Figure 2.

This figure shows that the two methods got better reward than no-change states. In the "Segmentation and Integration Method," the reward was increasing especially every 200 episodes.

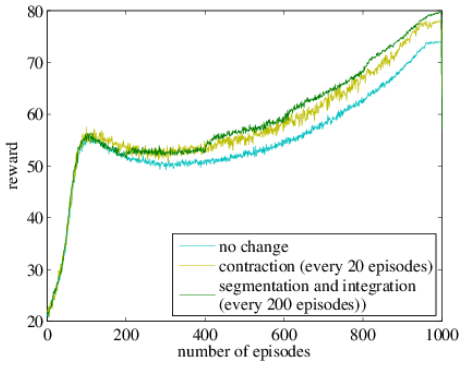


Fig. 2. Comparison of these methods (average of 100 trials)

4.4 Learning Residual Entropy Approach

We investigated an automatic learning residual entropy approach with the "Contraction Method" or "Segmentation and Integration Method" through simulation.

```
procedure Q-learning with the learning residual entropy approach
begin
  initialize Q-table Q, ∀s ∈ S, ∀a ∈ A;
  set initial state s0;
  set CONT_FLAG=1; or SEG_FLAG=1;
  set ENTROPY THRESHOLD;
    for cycle := 1 to MAXCYCLE do s := s0
    for time := 0:0.05:20
    s´ := GetNextState(s,a);
      if mod(time,0.2)==0
        sr´ = s´ ;
```

```
        r := GetReward(s_r', a);
        Q := UpdateQ(Q, s_r s_r', a);
        a := ActionSelect(Q, s_r');
        s_r = s_r';
      end
      s = s';
    end
    I=Entropy(Q);
    if mod(cycle,20)==0 && CONT_FLAG==1
    && I<ENTROPY_THRESHOLD
        (S,Q):=Contract(S,Q);
    elseif SEG_FRAG==1 && I<ENTROPY_
    THRESHOLD
        (S,Q):=Combine (S,Q);
        (S,Q):=Divide(S,Q);
    end
  end
end
```

In this method, the "Contraction Method" or "Segmentation and Integration Method" is done when learning residual entropy is under the entropy threshold. However, in the "Contraction Method," the average entropy of the previous 20 episodes is used because contractions make little change in action values' distributions; namely, learning residual entropy changes little. In the "Segmentation and Integration Method," the entropy of the previous episode is used. These average rewards from 951 episodes to 1000 episodes are shown in TABLES III and IV, and the entropy method's rewards are shown in Figure 3. Average entropy and an example of entropy are also shown in Figure 4 and 5.

TABLE III
Entropy approach with "Contraction Method"
(average of 100 trials)

Entropy threshold	Average reward	Contraction count
0.3	73.4586	0
0.6	75.4468	34.82
0.65	76.1829	40.69
0.7	**77.148**	**43.95**
0.75	76.2985	44.74
0.8	76.8721	45.98
0.9	76.7689	46.94
more than 1.0	--	50

TABLE IV
Entropy approach with "Segmentation and Integration Method"
(average of 100 trials)

Entropy threshold	Average reward	Contraction count
0.3	73.3312	0
0.5	74.0798	0.69
0.5	75.971	2.47
0.6	**77.1707**	**9.86**
0.65	73.5105	38.92
0.7	67.3112	127.66
0.9	58.5275	940.15
more than 1.0	--	1000

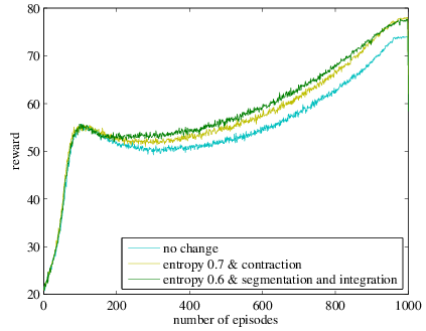


Fig. 3. Entropy methods reward (average of 100 trials)

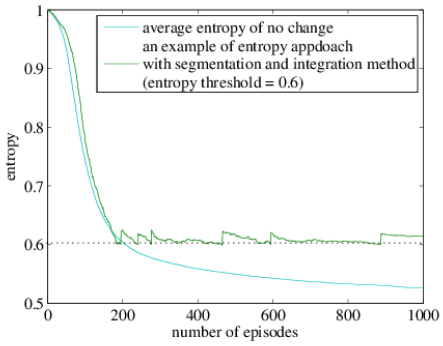


Fig. 4. Entropy example with "conventional approach" and "Segmentation and Integration Method"

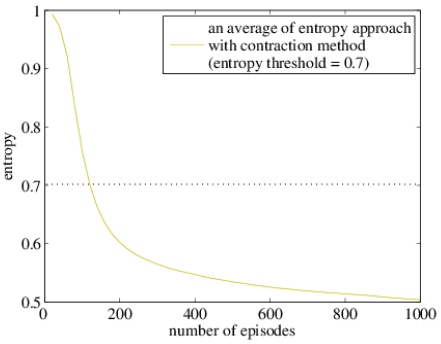


Fig. 5. Entropy example with "Contraction Method"

In the Contraction Method, contraction every 20 episodes was the best. In the Segmentation and Integration Method, segmentation and integration every 200 episodes was the best. Both methods could get better rewards than the case of evenly segmented states. In the Segmentation and Integration Method, the reward increased in particular for every episode of changing how to divide state space, which indicates that the method can be used to determine an appropriate segmentation method automatically by time. In the learning residual entropy approach with the Segmentation and Integration Method, states were segmented and integrated after approximately converging. Then, learning residual entropy increased, and after waiting for new converging, the next segmentation and integration were

done. Therefore, the first objective, which is to find good timing for changing state space, was accomplished.

However, in the learning residual entropy approach with the contraction method, learning residual entropy was not very increased after contraction, so we took an average of 20 episodes. From this, in the early stages of learning (about the first 100 episodes), contraction was not done and after approximately converging, contraction was done at fixed intervals. Thus, this method can also find good timing for changing state space.

5. Conclusions

Q-learning is a typical reinforced study method that targets discrete states and actions. However, finding appropriate space segmentation types is difficult. In this paper, we investigated the "Contraction Method," "Segmentation and Integration Method," and "Learning Residual Entropy Approach" with these two methods through single pendulum simulations. We showed changing space segmentation types while learning was effective and learning residual entropy could find good timing for changing state space.

Finding the appropriate entropy threshold automatically or finding more effective method to find good timing for changing state space remains as the future work.

Acknowledgment

This work was supported in part by the Ministry of Education, Culture, Science and Technology, Japan under Grant-in-Aid for Scientific Research No. 21700242 and funded by the Ministry of Internal Affairs and Communications, Strategic Information and Communications R&D Promotion Program.

References

[1] R. S. Sutton, "Learning to Predict by Method of Temporal Differences," Machine Learning, vol. 3, 1, pp. 9-44, 1988.

[2] T. Jaakkola, M. Jordan, and S. P. Singh, "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms," *Neural Computation*, vol. 6, pp. 341–362, 1992.

[3] C. J. C. H. Watkins and P. Dayan, "Technical Note: Q-Learning," *Machine Learning*, vol. 8, pp. 56–68, 1992.

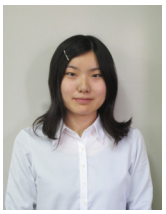
[4] Y. Kashimura, A. Ueno, and S. Tatsumi, "A Continuous Action Space Representation by Particle Filter for Reinforcement Learning," *JSAI2008*, pp. 118-121, 2008.

[5] A. Notsu, H. Honda, H. Ichihashi, and H. Wada, "Contraction Algorithm in State and Action space for Q-learning," Proc. of SCIS&ISIS, pp. 93-96, 2009.

[6] A. Notsu, H. Wada, H. Honda, and H. Ichihashi, "Cell Division Approach for Search Space in Reinforcement Learning," International Journal of Computer Science and Network Security, vol. 8, no. 6, 2008.

[7] A. Ito and M. Kanabuchi, "Speeding up Multi-Agent Reinforcement Learning by Coarse-Graining of Perception Hunter Game as an Example," IEICE Trans. D, vol. J84-D1, no. 3, pp. 285-293, 2001.

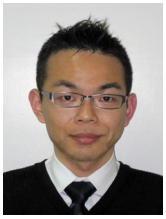
[8] M. Nagayoshi, H. Muraoand, and H. Tamaki, "Switching Reinforcement Learning to Mimic an Infant's Motor Development Application to Two-dimensional Continuous Action Space," Proc. SICE Annual Conference 2010 (SICE 2010), pp.243-246 (TA09-3(on DVD-ROM)), 2010.



Yuki Komori received the B.Eng. degree from Osaka Prefecture University in 2011.

She is with the Graduate School of Engineering, Osaka Prefecture University. Her research interests include reinforcement learning and multi-agent simulation.

E-mail: komori@hi.cs.osakafu-u.ac.jp



Akira Notsu received the B.E., M.I. and D. Informatics degrees from Kyoto University in 2000, 2002, and 2005, respectively.

He is currently an Assistant Professor, Department of Computer Science and Intelligent Systems, Osaka Prefecture University. His research interests include agent-based social simulation, communication networks, game theory, human machine interface, and cognitive engineering.

E-mail: notsu@cs.osakafu-u.ac.jp

Katsuhiro Honda (M'01) received the B.E., M.E. and D.Eng. degrees in industrial engineering from Osaka Prefecture University, Osaka, Japan, in 1997, 1999, and 2004, respectively.

He is currently an Assistant Professor, Department of Computer Science and Intelligent Systems, Osaka Prefecture University. His research interests include hybrid techniques of fuzzy clustering and multivariate analysis, data mining with fuzzy data analysis, and neural networks. He received a paper award and a young investigator award from the Japan Society for Fuzzy Theory and Intelligent Informatics (SOFT) in 2002 and 2005, respectively, and gave a tutorial titled "Introduction to Clustering Techniques" at the 2004 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE 2004).

E-mail: Honda@cs.osakafu-u.ac.jp

Hidetomo Ichihashi (M'94) received the B.E. and D.Eng. degrees in industrial engineering from Osaka Prefecture University, Osaka, Japan, in 1971 and 1986, respectively.

From 1971 to 1981, he was with the Information System Center of Matsushita Electric Industrial Co., Ltd., Tokyo, Japan. From 1981 to 1993, he was a Research Associate, Assistant Professor, and Associate Professor at Osaka Prefecture University, where he is currently a Professor in the Department of Computer Sciences and Intelligent Systems. His fields of interest are adaptive modeling of GMDH-type neural networks, fuzzy c-means clustering and classifier, data mining with fuzzy data analysis, human machine interface, and cognitive engineering.

E-mail: ichi@cs.osakafu-u.ac.jp