2011 10th International Conference on Machine Learning and Applications

Simple Reinforcement Learning for Small-Memory Agent

Akira Notsu, Katsuhiro Honda, Hidetomo Ichihashi, Yuki Komori
Department of Computer Science and Intelligent Systems
Osaka Prefecture University
1-1 Gakuen-cho, Nakaku, Sakai, Osaka 599-8531, Japan
Email: notsu@cs.osakafu-u.ac.jp

***Abstract*—In this paper, we propose Simple Reinforcement Learning for a reinforcement learning agent that has small memory. In the real world, learning is difficult because there are an infinite number of states and actions that need a large number of stored memories and learning times. To solve a problem, estimated values are categorized as "GOOD" or "NO GOOD" in the reinforcement learning process. Additionally, the alignment sequence of estimated values is changed because they are regarded as an important sequence themselves. We conducted some simulations and observed the influence of our methods. Several simulation results show no bad influence on learning speed.**

***Index Terms*—Reinforcement learning, Q-learning, State-action set categorize.**

I. Introduction

Reinforcement learning is chiefly applied to problems in the finite Markov decision process with a finite state and a finite action. However, recently, research on methods of expressing a continuous state and a continuous action, which is more similar to the real-world situation, has progressed. For example, studies have examined a control problem in order to enable a robot to walk. However, there are an infinite number of states and actions in the real world; so, the learning agent needs a lot of memory to learn optimum solutions in the Markov decision process.

Q-learning [1], [2] and SARSA are typical reinforced study methods that target discrete actions. Solving a problem that has a continuous state and a continuous action requires discrimination of the continuous state and continuous action.

In this paper, we propose Simple Reinforcement Learning (SRL) for a reinforcement learning agent that has a small number of stored memories and learning times. In our method, several estimated values are categorized into "GOOD" in the reinforcement learning process[3], [4]. Additionally, the alignment sequence of estimated values is rearranged as their priorities represented by the sequences themselves in the process.

We conducted some simulations and observed the influence of our methods. Simulation results show no bad influence on learning speed in the multi-agent hunter game.

II. Reinforcement Learning

Reinforcement Learning presupposes that agents can visit a finite number of states, and when visiting a state, a numerical reward will be collected; negative numbers represent punishments [5]. Each state has a changeable value attached to it. From every state, subsequent states can be reached by actions. The value of a given state is defined by the averaged future reward that can be accumulated by selecting actions from this particular state.

Q-learning is a 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 strength of Q-learning is that it can compare expected utility Q of available actions without requiring a model of the environment.

Q-learning process

```
procedure Q(0)-learning
begin
 initialize Q-table Q, ∀s ∈ S, ∀a ∈ A;
 set initial state s_0 and goal state s_n;
 for cycle:= 1 to MAXCYCYLE do s := s_0
  while s ≠ s_n do
   a :=ActionSelect(Q, s);
   r :=GetReward(s, a);
   s' :=GotoNextState(s, a);
   Q(s,a) := Q(s,a)
             +α [r + γ max_{a'∈A(s')} Q(s',a') − Q(s,a)];
   s := s';
  end
 end
end
```

Expected utility Q is iteratively updated. For each state s from state set S, and for each action a from action set A, we update Q-value Q, which depends on current state s of the agent, action a selected by the agent, next state s' of the agent after taking action a, and reward r received by the agent after the action. An update is calculated with 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.

978-0-7695-4607-0/11 $26.00 © 2011 IEEE
DOI 10.1109/ICMLA.2011.127

After Q-learning has finished, the optimal policy and the optimal value function have been found without continuously updating the policy during learning.

III. Simple Reinforcement Learning (SRL)

To categorize the $Value\,(s,a)$ into "GOOD" or "NO GOOD" requires little memory in comparison to Q-learning. Speaking in the extreme, precisely expected utility Q is not always required in the policy for modestly successful action selection.

SRL process

```
procedure Simple RL-learning
begin
 set initial state s_0 and goal state s_n;
 allocate memory Value(s_0,:);
 initialize Value(s_0,:);
 for cycle:= 1 to MAXCYCYLE do s := s_0
  while s ≠ s_n do
   a :=ActionSelect(Value, s);
   r :=GetReward(s, a);
   if r == positive and s is new state,
    allocate memory Value(s,:);
    initialize Value(s,:);
    Value(s,a) :="GOOD";
   end    s' :=GotoNextState(s, a);
   if Value(s,:) is newer allocated state than Value(s',:)
and max_{a'∈A(s')} Value(s',a) == "GOOD"
     reallocated memory Value(s,:) next to the memory
Value(s',:) ;
    Value(s,a) := "GOOD";
    delete inferior state Value(s,:)
   end
   s := s';
  end
 end
end
```

In our SRL process, if $Value\,(s',:)$ is a newer allocated state than $Value\,(s,:)$ and $\max_{a'\in A(s')} Value\,(s',a)$ == "GOOD" (: the $Value\,(s,a)$ quickly leads to a positive reward), "GOOD" is applied to $Value\,(s,a)$ and allocated next to $Value\,(s',:)$ (figures 2-4). In this way, the alignment sequence of estimated values is rearranged **as their priorities**.

Example of memory table

The following examples of memory table are demonstrated. In the example problem, there are 5 states and 2 actions. An agent gets rewards 10 (-10) at state s5 (state s1) and goes back to initial state s3.

Table I shows the Q-table of the example problem after sufficient learning ($\gamma = 0.9$) and table II shows the Simple RL Value-table of the example problem after sufficient learning.

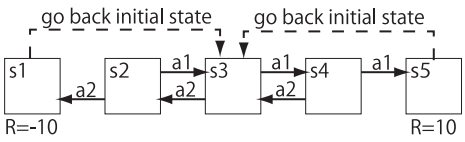

Fig. 1. Example problem

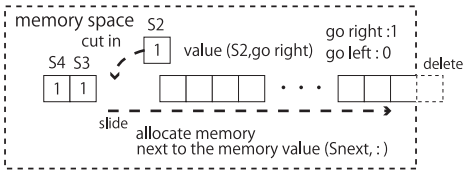

Fig. 2. Newer allocated "good" state (cut in)

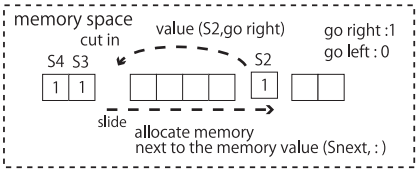

Fig. 3. Reallocated "good" state (order change)

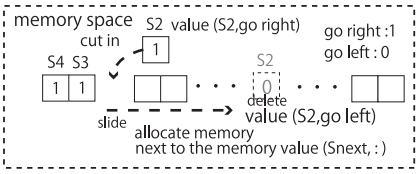

Fig. 4. Reallocated "good" state (rewrite)

TABLE I
Final Q-table of example problem

Q	a1	a2
s1	0	0
s2	8.1	-10
s3	9	7.29
s4	10	8.1
s5	0	0

TABLE II
Final SRL Value-table of example problem

$Value$	a1	a2
s2	GOOD	(unallocated)
s3	GOOD	(unallocated)
s4	GOOD	(unallocated)

IV. Simulation experiments

We simulated standard games named "Goal search" and "Hunter game" as a test case.

A. Goal search in 2D cell

In this game, agent actions are up, down, right and left. If the agent goes to the goal, it gets a reward and goes back to the initial state (start state).

The number of states is $n \times n$. The agent can observe its state completely.

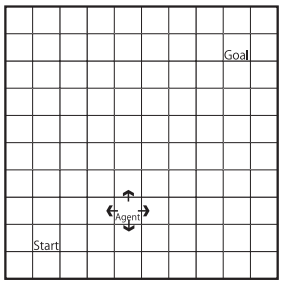

Fig. 5. Goal search in 2D cell

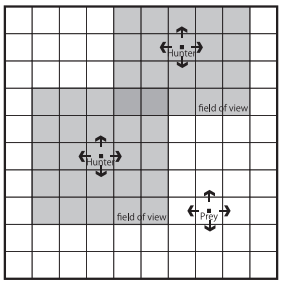

Fig. 6. Hunter game in 2D cell

B. Hunter game (pursuit game)

In this game[6], [7], agents' actions are going up, down, right, left and stay. If the agent is next to the prey agent, it gets a reward and goes back to the random state. The prey agent acts randomly (goes up, down, right, left and stays). Each agent cannot go through another agent.

The number of states is $m \times m$ (a prey agent position in the agent's field of view) $\times m \times m$ (another hunter agent position in the agent's field of view).

C. Simulation parameters

Agents' policies are ϵ-greedy. The following parameters are set.

- Learning rate $\alpha = 0.1$ in Q-learning expression.
- Discount ratio $\gamma = 0.9$ in Q-learning expression.
- Start state is $(2, 2)$ in Goal search.
- Goal state is $(9, 9)$ in Goal search.
- $n \times n = 5 \times 5, 10 \times 10$ (The number of states in Goal search)
- $m \times m = 9 \times 9$ (The agent's field of view) in Hunter game.

If the agents get a reward (go to goal state in Goal search or hunt down a prey agent in Hunter game) or does 1,000 actions, their episode ends and the next one begins where agents are set to the initial state.

V. Simulation results

Following figures show learning processes. In addition, goal search examples of agents' actions at the 100th episode and hunter game examples of agents' actions at the 1,000th episode are described in Figures 7, 8.

A. Goal search results

Goal search results are indicated in Figure 9. The vertical axis shows the goal turn and the horizontal axis shows the number of episodes. The blue, green, red and cyan lines respectively correspond to Q-learning agent, SRL agent, SRL32 agent that has only 32 states of memory and SRL16 agent that has only 16 states of memory.

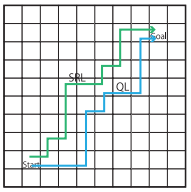

Fig. 7. Examples of learned actions at 100th episode

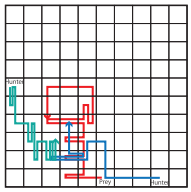

Fig. 8. Examples of learned actions (SRL) at 1,000th episode

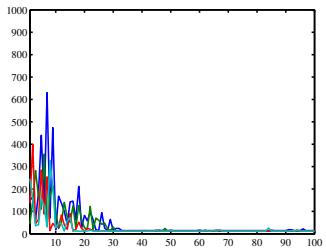

Fig. 9. Examples of learning process ($\epsilon = 0.1$)

The mean and the median of the number of learned actions during 100 episodes are shown in table III. SRL16 indicates an SRL agent that has only 16 states of memory.

Table IV shows the necessary amount of memory for action-state value table.

B. Hunter game

Hunter game results are described in Figures 10-15. The vertical axis shows goal turn and the horizontal axis shows the number of episodes. The blue, green, red and cyan lines respectively correspond to Q-learning agent, SRL agent, SRL1024 agent that has only 1,024 states of memory and SRL512 agent that has only 512 states of memory.

The mean of the number of learned actions during 1,000 episodes is shown in Tables V, VI.

C. Discussion

The results implied that SRL works as well as normal Q-learning in Goal search (Perfectly Observable Markov Decision Processes). However, some SRL agents do not work well in the Hunter game (Partly Observable Markov Decision Processes) and need more episodes to learn (convergence). The reason for this is that our method put aside trivial information. However, there have also been SRL agents that work better than normal Q-learning agents.

If an agent does not have enough memory to learn, it gets lost on the way at the first part of each episode. This is because SRL stocks $Value\,(s, a)$ in goal-nearest order preferentially.

On the other hand, our mechanism can be affected by action selection policy more than QL (Tables III, V and VI; Figures 12-14). Too many cuts in processes caused a lack of stability of the action chain.

TABLE III
MEAN (AND MEDIAN) OF THE NUMBER OF LEARNED ACTIONS DURING 100 EPISODES (100 AGENTS)

ϵ	QL	SRL	SRL32	SRL16
0.01	14.62 (14)	14.62 (14)	14.48 (14)	16.24 (14)
0.02	14.62 (14)	14.44 (14)	14.56 (14)	20.04 (14)
0.05	14.68 (14)	14.48 (14)	14.50 (14)	29.86 (18)
0.1	14.34 (14)	14.32 (14)	14.34 (14)	52.54 (44)
0.2	14.26 (14)	14.28 (14)	14.18 (14)	69.92 (50)
0.5	14.08 (14)	24.06 (14)	31.96 (22)	91.56 (80)

TABLE IV
AMOUNT OF MEMORY (GOAL SEARCH)

QL	SRL	SRL32	SRL16
sizeof(float)$\times 4 \times 10 \times 10$	2(bit) $\times 10 \times 10$	2(bit) $\times 32$	2(bit) $\times 16$

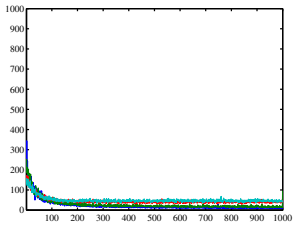

Fig. 10. $\epsilon = 0.1$, $n = 5$, 100 mean

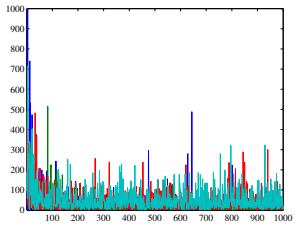

Fig. 11. Examples of learning process ($\epsilon = 0.1$, $n = 5$)

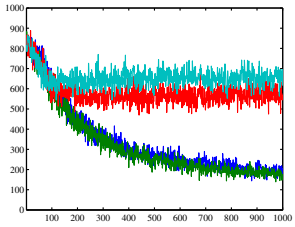

Fig. 12. $\epsilon = 0.01$, $n = 10$, 100 mean

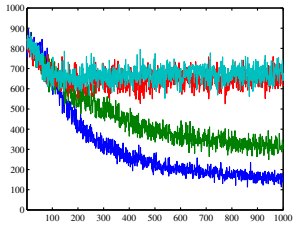

Fig. 13. $\epsilon = 0.1$, $n = 10$, 100 mean

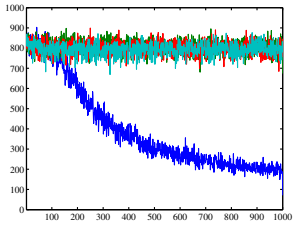

Fig. 14. $\epsilon = 0.5$, $n = 10$, 100 mean

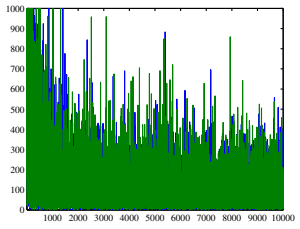

Fig. 15. Examples of learning process (QL and SRL, $\epsilon = 0.1$, $n = 10$)

VI. CONCLUSION

We proposed a novel reinforcement learning method (SRL) to create a novel reinforcement learning agent that does not need much memory. This method categorizes several state-action sets as "GOOD" and arranges alignment sequences of estimated values as their priorities.

The proposed method is applied to several simulations (Goal search (Perfectly Observable Markov Decision Processes) and Hunter game (Partly Observable Markov Decision Processes)) in order to demonstrate the adaptation ability of the model. Experimental results showed that our method is as useful as normal Q-learning and saves agent memory.

An experimental result implied that our mechanism can be affected by action selection policy. Appropriate policy should be set according to agent memory and ability.

Comparative study with other learning approaches remains for future work.

TABLE V
MEAN (AND MEDIAN) OF THE NUMBER OF LEARNED ACTIONS DURING 1,000 EPISODES ($n = 5$, 100 AGENTS)

ϵ	QL	SRL	SRL1024	SRL512
0.01	17.96 (13)	30.45 (20)	30.73 (20)	34.95 (21)
0.1	15.93 (13)	35.29 (24)	41.41 (36)	41.83 (31)
0.5	8.09 (7)	33.23 (21)	41.05 (22)	40.03 (31)

TABLE VI
MEAN (AND MEDIAN) OF THE NUMBER OF LEARNED ACTIONS DURING 1,000 EPISODES ($n = 10$, 100 AGENTS)

ϵ	QL	SRL	SRL1024	SRL512
0.01	207.04 (160)	177.23 (154)	559.27 (629)	680.79 (816)
0.1	153.03 (91)	299.96 (206)	619.38 (672)	678.45 (866)
0.5	70.91 (32)	674.05 (877)	755.10 (1000)	778.82 (1000)

TABLE VII
AMOUNT OF MEMORY (HUNTER GAME)

QL	SRL	SRL1024	SRL512
sizeof(float)$\times 5 \times 6561$	3(bit) $\times 6561$	3(bit) $\times 1024$	3(bit) $\times 512$

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.

REFERENCES

[1] Jaakkola,T., Jorban,M., and Singh,S: On the convergenece of stochastic iterative dynamic programming algorithms. Neural Computarion, 1994.

[2] C. J. C. H. Watkins and P. Dayan: "Technical Note:Q-learning", Machine Learning, Vol. 8, 1992.

[3] A. Notsu, K. Honda and H. Ichihashi: "Proposal for Notion Learning of Reinforcement Learning", Proc. of Social Intelligence Design 2009.

[4] A. Notsu, K. Honda and H. Ichihashi: "Particle Swarm for Reinforcement Learning", Proc. of Joint 5th International Conference on Soft Computinng and Intelligent Systems and 11th International Symposium on Advanced Intelligent Systems, 809-812, 2010.

[5] R. S. Sutton and A. G. Barto: Reinforcement Learning -An Introduction-, pp.32-33. The MIT Press,1998.

[6] Sttel, L. and Vogt, P. : "Grounding Adaptive Language Games in Robotic Agents", The Fourth European Conference on Artificial Life(ECAL '97), pp. 474-482, 1997.

[7] Tan, M. : "Multi-agent Reinforcement Learning: Independent vs. Cooperative Agent", The 10th International Conference on Machine Learning(ICNL' 93), pp. 330-337, 1993.