Contextual Behaviors and Internal Representations Acquired by Reinforcement Learning with a Recurrent Neural Network in a Continuous State and Action Space Task

Hiroki Utsunomiya and Katsunari Shibata

Oita University 700 Dannoharu Oita Japan
shibata@cc.oita-u.ac.jp

Abstract. For the progress in developing human-like intelligence in robots, autonomous and purposive learning of adaptive memory function is significant. The combination of reinforcement learning (RL) and recurrent neural network (RNN) seems promising for it. However, it has not been applied to a continuous state-action space task, nor has its internal representations been analyzed in depth. In this paper, in a continuous state-action space task, it is shown that a robot learned to memorize necessary information and to behave appropriately according to it even though no special technique other than RL and RNN was utilized. Three types of hidden neurons that seemed to contribute to remembering the necessary information were observed. Furthermore, by manipulate them, the robot changed its behavior as if the memorized information was forgotten or swapped. That makes us feel a potential towards the emergence of higher functions in this very simple learning system.

1 Introduction

It goes without saying that it is important to memorize useful information from time series of sensor signals and utilizing it when required. If such functions could be learned autonomously, it would be a great progress in developing human-like intelligence in robot. In reinforcement learning (RL) research, such functions have been discussed as one of the ways to solve POMDP (Partially Observable Markov Decision Problem) or perceptual aliasing problems. Many techniques have been proposed to solve this problem by introducing some memory system, and among them, utilizing a recurrent neural network (RNN) seems the most promising way because of its autonomous ability to extract important information effectively and to utilize it. After the proposition of the "Recurrent-Q" technique[1], several works have followed[2][3]. Most of them deal with tasks with a discrete state space. Against such a trend, there are some works which tackled to the continuous state space problems[4][5][6]. Onat et al.[4] showed that a dynamical system could be controlled without perceiving velocity information. In the work of Bakker et al[5], a robot learned to memorize and utilize important binary information to get a reward in a T-maze problem even though it had to

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 970–978, 2009.

make another decision before it made the decision that required the memorized information. They also showed that their method worked on an experiment using a real robot[6]. However, the action space is still discrete and the memorized information was not investigated in much depth.

In this paper, we applied RL with a RNN to a robot navigation task in which a robot has to choose one of two goals according to the flag signals perceived on a switch. The situation is similar to that in [5], but the exploration field is a two-dimensional free space, and the robot, goals, and switch are located randomly in the field at every episode. The state and action spaces are both continuous. Another big difference is that the processing system is as simple as just a RNN trained by RL, while in [5], special technique named "ARAVQ" was used to extract events in episodes. A significant point in this paper is that without employing any special techniques, interesting contextual behaviors are acquired through learning. In order to extract important events from episodes, some relevant criterion to judge the importance is usually introduced. However, when considering that RL has its criterion for the system optimization to increase its reward and decrease its punishment as much as possible, the criterion should be consistent in one system. We think that we should leave as much of robot process as possible to the autonomous learning by the combination of RL and NN. Furthermore, the hidden representation acquired through learning is observed, and an examination of the change in contextual behavior through manipulations of the outputs of hidden neurons allows us to investigate the properties of the memorized information in depth.

2 Learning System

The learning system is very simple. From sensors to motors, there is only one popular Elman-type recurrent neural network (RNN)[7] whose hidden outputs are fed back to the input layer at the next time step. It is trained by popular reinforcement learning (RL) technique, that is Actor-Critic with TD-Learning[8]. The present observation vector $\boldsymbol{x}_t$ is the network input. The output neurons are divided into one critic (state value) output and the other actor (motor commands) outputs. The motor command vector $\tilde{\boldsymbol{A}}_t$ is derived by adding an uniform random number vector $\boldsymbol{rnd}_t$ to the actor output $\boldsymbol{A}(\boldsymbol{x}_t)$. After the robot has moved according to $\tilde{\boldsymbol{A}}_t$, a new observation vector is obtained. After forward computation of the RNN for the new input signals $\boldsymbol{x}_{t+1}$, the training signal $C_{d,t}$ for the critic and $\boldsymbol{A}_{d,t}$ for the actor output vector for the previous time step are generated autonomously based on Temporal Difference learning as

$$C_{d,t} = C(\boldsymbol{x}_t) + \hat{r}_t = r_{t+1} + \gamma C(\boldsymbol{x}_{t+1}) \tag{1}$$

$$\boldsymbol{A}_{d,t} = \boldsymbol{A}(\boldsymbol{x}_t) + (\tilde{\boldsymbol{A}}_t - \boldsymbol{A}(\boldsymbol{x}_t))\hat{r}_t = \boldsymbol{A}(\boldsymbol{x}_t) + \boldsymbol{rnd}_t \cdot \hat{r}_t \tag{2}$$

$$\hat{r}_t = r_{t+1} + \gamma C(\boldsymbol{x}_{t+1}) - C(\boldsymbol{x}_t) \tag{3}$$

where r_t is a given reward, γ is a discount factor, $\hat{r}_t$ is the TD error, and C is the critic output. After one more forward computation for the input signals at the

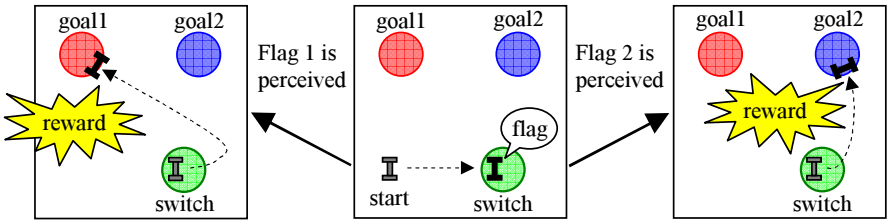


Fig. 1. The mission of the robot is to step on the switch at first and then to go to the correct goal according to the flag signals that can be perceived only on the switch

previous time step, the training signals are given to the corresponding output, and the network is trained by BPTT (Back Propagation Through Time)[9]. The output function used in the hidden and output neurons is the sigmoid function whose value ranges from -0.5 to 0.5, and the training signals are bounded from -0.4 to 0.4. To adjust the offset between actual critic and network output, they are shifted up or down by 0.5. The network output 0.0 corresponds to critic 0.5.

3 Experimental Results

In this experiment, a task in which a robot requires contextual behavior was employed. As shown in Fig.1, on a two dimensional 15 × 15 continuous, flat square space of arbitrary distance units, the robot, one switch and two goals are located randomly at the beginning of every episode. They do not overlap with each other. The goals and switch are circles of radius 0.5. At first, the robot has to step on a switch before it approaches to a goal. Only when the robot is on the switch, it can perceive the two flag signals, and one of them chosen randomly is 1 and the other is 0. When the robot is not on the switch, they are always 0. At each episode, the robot can know from the flag signals which goal it should go to.

As shown in Fig.2, the observation vector has 13 elements in total. 5 signals represent the distances to the objects and wall. 6 signals represent the angles to the objects. 2 signals represent the flag information. A transformation using the exponential function on the distance signals magnifies the range for small distances. The signals are input into the RNN. Before learning, the robot does not know the meaning of the switch, goals, or flag signals. The robot has two wheels. It can rotate its right and left wheels separately, and so the robot can move any directions except sideways. Each element of $\boldsymbol{A}_t$ multiplied by 1.25 indicates the distance that each wheel moves in one time step. The value is bounded from -0.5 to 0.5. The interval between two wheels is 0.32. The robot can move up to 0.5 forward or backward, and can rotate up to 180 degrees in one time step. Whether the robot steps on a goal or switch is judged actually by whether the center of the robot is on the area of goal or switch.

The task is episodic. Before each episode, all the hidden outputs are reset to 0.0, and the critic value after reaching a goal is 0.0. Only when the robot steps on

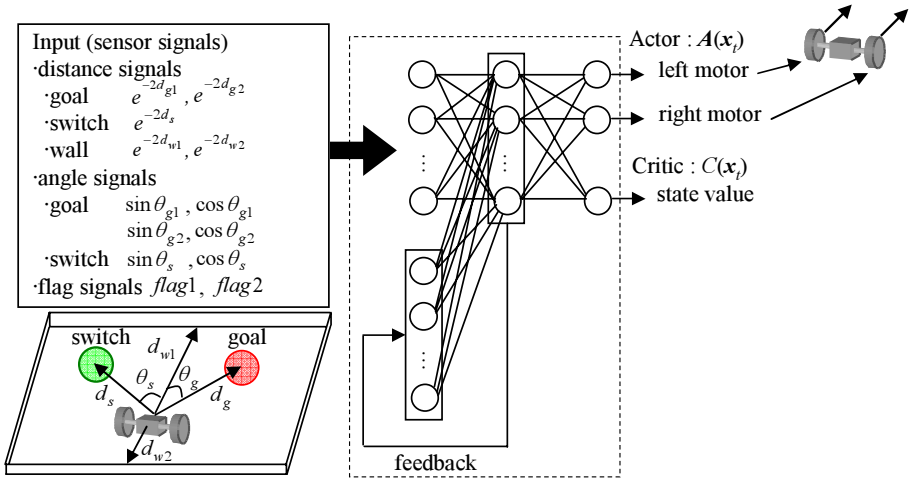


Fig. 2. Input and output signals of the recurrent neural network

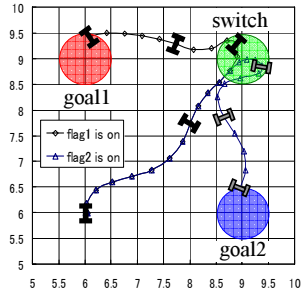
the correct goal after stepping on the switch it can get a reward $r = 0.9$, and the episode terminates. However, when the robot goes to the incorrect goal or goes to the correct goal before stepping on the switch, it does not get any reward, but a penalty of $r = -0.1$, and the episode does not terminate. Moreover, if the robot bumps a wall, it also gets a penalty of $r = -0.1$.

The task is divided into 3 levels, and the learning becomes increasingly difficult as shown in Table 1. Initial location range means all the objects are located in this area. Exploration range means the range of the random numbers in $\boldsymbol{rnd}$ that is added to the actor output as trial and error factors. If the condition that the robot reaches the correct goal within 50 time steps is satisfied for 100 successive episodes, the robot moves to the next level. The learning terminates when the robot is in the level 3 and the condition is satisfied for 100 episodes.

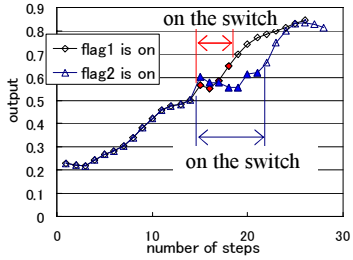
The RNN has three layers and 50 hidden neurons. The maximum time steps traced back through time for BPTT is 20. The discount factor γ is 0.96. The initial weight for each hidden-output connection is 0.0 and that for each non-feedback input-hidden connection is chosen randomly from -1.0 to 1.0. For the self-feedback connections, the initial weight is 4.0, while that for the other

Table 1. Incremental learning difficulties. x indicates the number of episodes in the level.

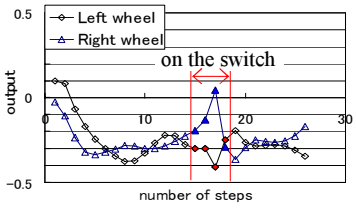
	Level1	**Level2**	**Level3**
Initial location range	2×2	3×3	4×4
Exploration range	$\pm\frac{1}{2}e^{-5\times10^{-5}x}$	$\pm\frac{2}{5}e^{-5\times10^{-5}x}$	$\pm\frac{3}{10}e^{-5\times10^{-5}x}$
Upper bound of episode length	10000	5000	3000



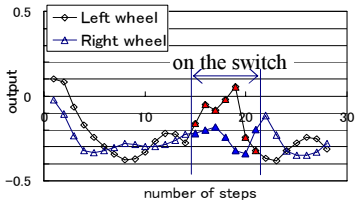
(a) An example of the robot's trajectory



(b) The temporal change of the critic output



(c) The actor output (flag 1 is on)



(d) The actor output (flag 2 is on)

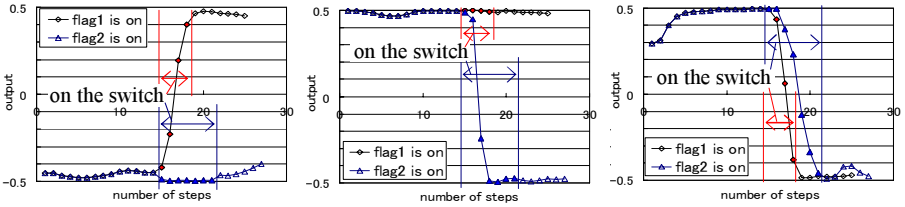
Fig. 3. A sample of the learning result for both flag states after learning

feedback connection is 0.0, which makes the learning of memory function easy. The learning constant is 0.05 for the feedback connections, and 0.2 for the others. More than 10 runs with different random number sequences were performed, but the learning results were similar to each other. One of them is introduced in the following.

Learning terminates after 31131 episodes. The robot can go to the switch at first, and then it can approach the correct goal corresponding to the flag signals perceived only at the switch for any initial locations such as Fig.3 (a). In Fig.3 (b), the critic output increases in both cases, but in Fig.3 (c) and (d), we can see big differences in the actor outputs especially on the switch that make the difference in the direction of the robot's approach after stepping on it.

4 Contextual Behaviors and Hidden States: Tests and Observations

How the robot remembers the information of flag signals in the RNN after learning is analyzed here. First, we observed the temporal change of each hidden neuron during one episode for several switch and goal configurations. There are three types of hidden neurons that are considered to contribute to remembering the flag signals after leaving the switch. Fig.4 shows the response of a hidden neuron for each type to each flag state in the example of Fig.3. The type 1 neurons changed its output only when the flag 1 was on, and the type 2 neurons changed its output only when the flag 2 was on. On the other hand, the type 3



(a) Type 1 (responds only to flag 1) (b) Type 2 (responds only to flag 2) (c) Type 3 (responds to both flags)

Fig. 4. Three types of hidden neurons that seem to contribute to keeping the flag state

neurons changed its output at the switch for both cases. There are 7 type 1, 4 type 2, and 4 type 3 neurons out of the total of 50 hidden neurons.

4.1 Manipulating of Hidden State during an Episode

In order to make sure that these neurons remember the flag state, the hidden neuron outputs were manipulated. Before the robot reached the switch, the outputs of type 1 hidden neurons that respond only to the flag 1 were stored, and after leaving the switch, the outputs of type 1 hidden neurons were reset to those stored before the switch.

When the flag 1 was on, the robot returned to and stepped on the switch again and then approached the goal 1 as shown in Fig.5 (a). However, when the flag 2 was on, the value of the hidden neuron did not change noticeably after the resetting. The robot's trajectory did not seem to change, and it went to the goal 2 without returning to the switch. When the outputs of type 2 hidden neurons that respond only to the flag 2 were stored and reset, the robot returned to the switch when the flag 2 was on, but when the flag 1 was on, it approached to the goal 1 without returning to the switch. It is thought that the division of roles in memory among hidden neurons was acquired through RL.

Next, in only one of the type 1 hidden neurons, the value was stored before stepping on the switch and was reset to the stored one afterwards. Then, soon after the reset, the hidden neuron retrieved the value just before the reset, and the robot's trajectory does not change noticeably. It is thought that the robot memorized the information about the flag with a distributed representation among some neurons, and the function of associative memory using the recurrent structure emerged through RL.

In the third test, the flag 1 was on at first. After the robot left the switch, the outputs of all the three types of hidden neurons were stored. Then the robot was taken back to the initial location with all hidden neurons being reset to 0.0 and began to move again. However, this time, the flag 2 was on and the flag 1 was off on the switch. As shown in Fig.6(a), after stepping on the switch, it began to approach the goal 2 because the robot perceived the flag 2 on the switch, but after that, the outputs of all the three types of hidden neurons were reset to the stored ones when the flag 1 had been on. After that, the robot changed its

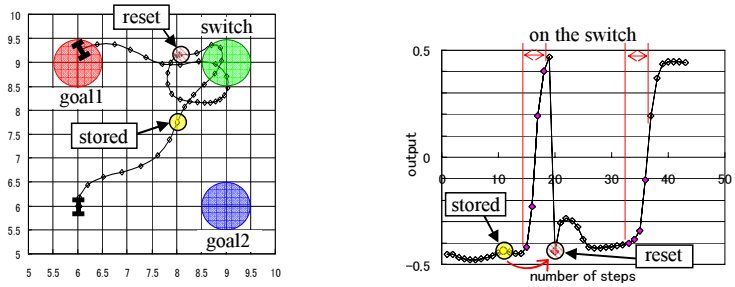


Fig. 5. (a) Robot's trajectory when the output of type 1 hidden neurons were reset to those stored before stepping on the switch and (b) output change of one of the type 1 hidden neurons for the case of (a)

direction to the goal 1. However, since that was not the correct goal, no reward was given and the episode did not terminate there. Surprisingly, as shown in Fig.6(b), the robot then returned to the switch again and approached the goal 2 that is the correct one.

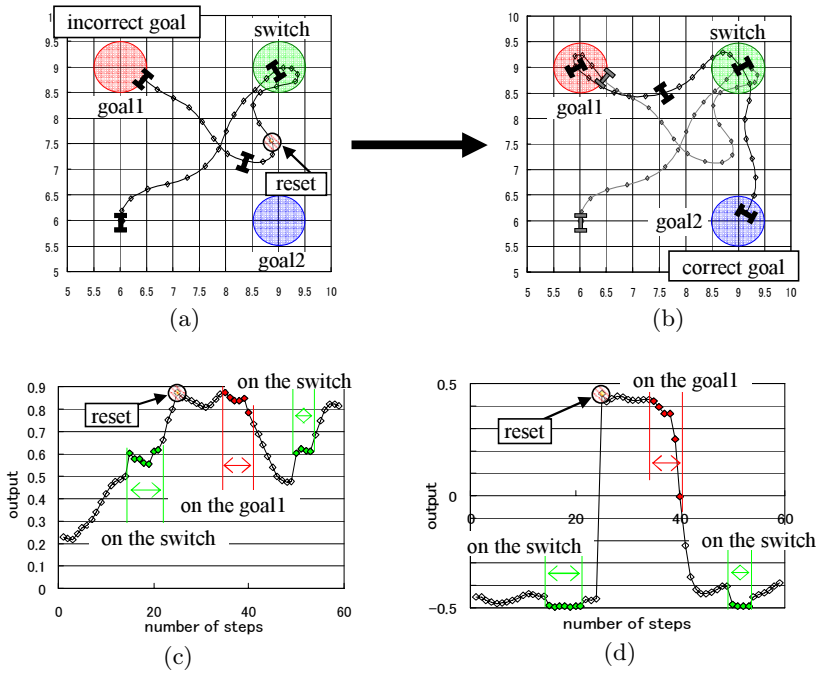


Fig. 6. Robot's trajectory (a) until it arrived at the incorrect goal and (b) after the arrival, when the flag 2 was on and the hidden neuron outputs were reset to those stored when the flag 1 had been on. Output change of (c) critic and (d) a type 1 hidden neuron for the case of both (a) and (b)

As shown in Fig.6(c) and (d), it is interesting that when the episode did not terminate on the goal 1, the outputs of the critic and type 1 hidden neurons that respond only to the flag 1 decreased suddenly. In the robot's experiences, when it stepped on the switch beforehand and went to the correct goal, the episode always terminated with a reward. When the episode did not terminate, it had missed to step on the switch in advance or had arrived to the incorrect goal. It is suggested that due to the generalization from such experiences, the robot misunderstood that it had forgotten to step on the switch even though it actually had. Such behaviors seem very human-like. The intelligent behaviors seem to be controlled depending on the abstract state that is constracted by memorizing the flag signals in the RNN based on understanding of their importance. We suppose it shows the potential toward the emergence of higher functions that the proposed very simple learning system has.

5 Conclusion

A robot with a very simple learning system consisting of a recurrent neural network trained using reinforcement learning without any other special techniques learned a continuous state-action space task that requires a memory function. After learning, it was observed that replacement of the hidden states in the middle of one episode lead to sudden change of the contextual behavior. It was also observed that no termination of one episode caused a misunderstanding in the robot, which behaved as if it had forgotten to take the necessary action, and returned to perform the action again. We think that the learning system is so simple and general that it can be applied widely to many tasks. The observed interesting behaviors suggest that the learning system is very simple but promising towards the emergence of higher functions.

Acknowledgments. This research was supported by JSPS Grant in-Aid for Scientific Research ♯15300064 and ♯19300070.

References

1. Lin, L., Mitchell, T.M.: Memory Approaches to Reinforcement Learning In Non-Markovian Domains. Technical Report CMU-CS-TR-92-138, CMU, Computer Science (1992)
2. Onat, A., Kita, H., Nishikawa, Y.: Recurrent Neural Networks for Reinforcement Learning: Architecture, Learning Algorithms and Internal Representation. In: Proc. of IJCNN 1998, pp. 2010–2015 (1998)
3. Mizutani, E., Dreyfus, S.E.: Totally Model-Free Reinforcement Learning by Actor-Critic Elman Network in Non-Markovian Domains. In: Proc. of IJCNN 1998, pp. 2016–2021 (1998)
4. Onat, A., Kita, H., Nishikawa, Y.: Q-Learning with Recurrent Neural Networks as a Controller for the Inverted Pendulum Problem. In: Proc. of ICONIP 1998, pp. 837–840 (1998)

5. Bakker, B., Linaker, F., Schmidhuber, J.: Reinforcement Learning in Partially Observable Mobile Robot Domains Using Unsupervised Event Extraction. In: Proc. of IROS 2002, vol. 1, pp. 938–943 (2002)
6. Bakker, B., Zhumatiy, V., Gruener, G., Schmidhuber, J.: A Robot that Reinforcement-Learns to Identify and Memorize Important Previous Observations. In: Proc. of IROS 2003, pp. 430–435 (2003)
7. Elman, J.L.: Finding structure in time. Cognitive Science 14, 179–211 (1990)
8. Barto, A.G., Sutton, R.S., Anderson, W.: Neuronlike adaptive elements can solve difficult learning control problems. IEEE Trans. on Systems, Man, and Cybernetics 13(5), 834–846 (1983)
9. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L., Group, P.R. (eds.) Parallel distributed processing. MIT Press, Cambridge (1986)