Acquisition of Deterministic Exploration and Purposive Memory through Reinforcement Learning with a Recurrent Neural Network

Kenta GOTO and Katsunari SHIBATA
Dept. of Electrical & Electronic Engineering, Oita University, 700 Dannoharu, Oita 870-1192, JAPAN
Email: shibata@oita-u.ac.jp

Abstract—**The authors have propounded that various functions emerge purposively and harmoniously through reinforcement learning with a neural network. In this paper, emergence of deterministic "exploration" behavior, which is different from the stochastic exploration and needs higher intelligence, is focused on. In order to realize the intelligent exploration behaviors, it becomes a key point whether the recurrent neural network memorizes necessary information and utilizes it to generate appropriate actions. In the simulation of 3×3 grid world with an invisible goal task, by introducing a recurrent neural network for Q-learning, an agent can represent more accurate Q-values considering the past experiences, and that is suggested to enable to learn appropriate actions. The acquired knowledge can be generalized in some unknown environment to some extent. In another task in a simple environment with a random-located branch, it is also shown that the recurrent neural network cleverly memorizes and keeps the branch position to represent accurate Q-values after learning.**

Keywords*—*reinforcement learning; recurrent neural network; deterministic exploration; memory; function emergence.

I. Introduction

Although the intelligence of modern intelligent robots has been growing, it has to be said that the flexibility in the robots is still far poorer than that in humans. When we see the processing system for both, it is easily known that our brain is massively parallel and cohesively flexible, while the robot's process usually consists of a series of functional modules developed by humans as a designer. Unlike recognition or control, higher functions usually do not have direct connections from sensors or to actuators, and so it is difficult even to decide the inputs or outputs. If they are given in advance, the flexibility will be lost. From these views, the authors have thought that to develop human-like flexibility or intelligence in robots, the architecture and learning should be inspired more by humans. Therefore, the authors have propounded that various functions emerge purposively and harmoniously through reinforcement learning with a neural network, and have investigated whether each of the functions actually emerges or not. One of such functions is "deterministic exploration" [1][2][3].

In reinforcement learning research field, "exploration" generally means stochastic action selection using random numbers. However, human exploration seems to be highly strategic and to need higher intelligence. For efficient learning, active exploration considering the past experience such as the number of occurrences of each state has been investigated since a long time before [4][5]. However, there are no works in which the acquisition of exploration strategy through learning is aimed, as far as the authors know.

When we are looking for a lost key, we don't act randomly, but act very strategic. We may check the places where we think the key is highly possible to exist and/or close to us. For such strategic exploration, many things should be considered in parallel including the present situation and past trace. The authors think that such intelligent exploration should be acquired through learning. Inspired by the big gap between the human exploration and that in reinforcement learning, the authors have explored the possibility of emergence of intelligent exploration through reinforcement learning with a recurrent neural network, and have shown some strategic exploration behaviors emerge [1][2][3].

In order to realize intelligent exploration, it becomes a key point that how the recurrent neural network memorizes necessary information and utilizes it to generate appropriate exploration behaviors. In [1], an agent learns exploration behaviors in a grid-world environment where there are two landmarks of goal that are located randomly at each episode and one of them is a real goal and the other is a fake. The agent acquires the exploration behavior; going to one of the landmarks at first, and if it is not the real goal, going to the other one. Furthermore, in the recurrent neural network after learning, it is observed that a neuron represents whether one of the landmarks has been visited already or not.

Then, in this paper, a more confusing task is introduced, and acquired exploration behaviors are shown. Furthermore, from the viewpoint of purposive memory, it is shown that the learning to realize more precise value function enables the agent to memorize the branch position that changes randomly at the beginning of each episode.

In many works, a recurrent neural network is used in reinforcement learning, and it has been shown that memorization of necessary information emerges [6][7][8][9][10]. However, in most of them, memorized information takes a binary value, such as the direction of an arrow that is used to choose a way to go at the branch place appeared after some while. In [9][10], memorization of predicted useful non-binary information and relay of the information among a couple of hidden neurons were shown. As well as [9][10], this paper also has a purpose to show the ability to memorize necessary non-

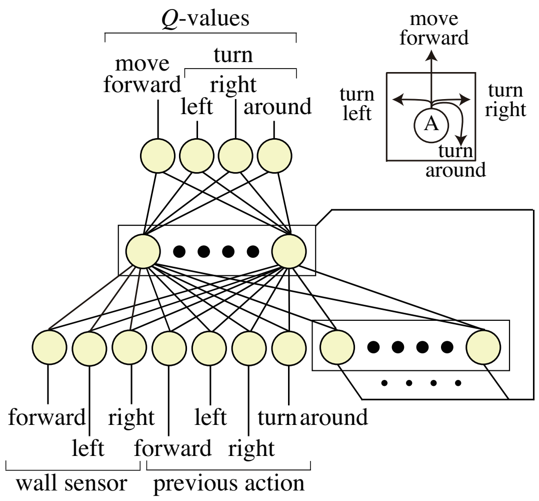


Fig. 1. Learning system consisted of one Elman-type recurrent neural network and its inputs and outputs.

binary information through reinforcement learning without providing any knowledge about which information is valuable to be memorized.

II. Learning System

The learning system is quite simple. As shown in Fig. 1, there is one recurrent neural network, and perceptual signals of an agent are the input of the network. The recurrent network here is a popular 3-layer Elman network whose hidden outputs are fed back as a part of the input at the next time step. In this paper, since a discrete space task is employed, Q-learning [11] is used as a reinforcement learning algorithm. The number of outputs of the recurrent network is the same as the number of possible actions for the agent. The outputs of the network are used as Q-values after linear transformation. Here, the output function of each neuron is the sigmoid function with the value range from -0.5 to 0.5. When an output is used as the Q-value for the corresponding action, 0.4 is added, while the ideal Q-value is used actually as a training signal after 0.4 is subtracted. Boltzmann selection [12] is used as an action selection. Only the output corresponding to the executed action a_t is trained using the training signal $T_{a_t,t}$ generated based on Q-learning as

$$T_{a_t,t} = r_{t+1} + \gamma \max_{a'} Q_{a'}(\mathbf{S}_{t+1}) \tag{1}$$

where r indicates a reward, γ indicates a discount factor, Q_a indicates the Q-value for the action a, and $\mathbf{S}$ is the input vector of the network. The network is trained by BPTT (Back Propagation Through Time) [13] supervised learning algorithm. Note that the system and learning are very general without any special techniques for learning of exploration.

III. Simulation

Here, discrete grid world tasks are employed. At first, the learning results in a 3×3 maze task is introduced. In this task, since the agent can change its direction by a "turn" action, the environment is more confusable for the agent than the 3×3 maze task in the previous work in which the agent direction was not considered [1]. After that, to investigate the acquisition of memory function, another simple environment that has always only one branch position is prepared.

A. Task Setting

In 3×3 maze problem, 9 squares are arranged in the shape of 3×3 as shown in Fig. 2. At each episode, an agent always starts from the center square. The goal state is randomly located on one of the squares except the center, and is fixed during one episode. Except for the 12 walls that form the boundary between the 3×3 world and the outer world, 4 walls are allocated randomly at each episode on the condition that the agent can visit all the squares from the center square that is the start location. There are 192 wall allocation patterns, and 1536 combinations when both wall and goal allocations are considered. The agent can perceive only local information, that is whether a wall exists or not in each of "forward", "left" and "right" directions of the agent. Possible actions of the agent are "move forward", "turn left", "turn right", and "turn around". The state transition is deterministic. The important setting is that the agent has no way to know where the goal is located, and so the agent has to explore to find the goal. If the agent reaches the goal state, it gets a reward, and the episode terminates. No penalty is imposed even when it collides a wall.

The input signals include 3 signals from a wall sensor and each of them represents the existence of a wall at the corresponding direction. The input signals also include 4 signals to represent the previously executed action. Each input takes a binary value. For the agent, the Q-value becomes larger for the state and action when it reaches the goal state in smaller number of steps, and so the agent learns to reach the goal as early as possible. In other words, the agent learns to find invisible goal state effectively considering the context using the recurrent neural network.

The other parameter settings are seen in Table I. Stable and efficient error propagation is expected by the special connection weights for the feedback connections. Since learning is more stable in the branch task, the learning rate is larger to get more accurate Q-values.

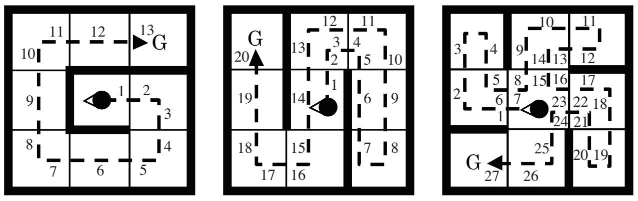
B. 3×3 *maze problem*

At first, a 3×3 maze problem is learned. Figure 3 shows the learning curve. The average number of steps to the goal during learning is plotted at every 1,000 episodes. It should be noted that the number is also influenced by the random factor in Boltzmann selection where the temperature is reduced gradually during learning.

Figure 2 shows the agent behaviors after learning for sample 3 wall allocations. The agent initially faces the left direction. In these cases, the goal was not allocated until the agent visited the last square. Figure 4 shows the change of the maximum Q-value at each step for the cases of Fig. 2. The line with no plots indicates the ideal Q-value. Since it is difficult to calculate the ideal value for all the states, the line is drawn

TABLE I
PARAMETER SETTINGS

Number of episodes	500,000
The maximum number of steps when the agent cannot reach the goal	250
Temperature in Boltzmann selection	0.05 → 0.0025
Discount factor γ	0.92
Reward r at the goal state	0.8
Number of external inputs	7
Number of hidden neurons	40
Number of outputs	4
Initial output of each hidden neuron at each episode	0.0
Truncated propagation step in BPTT	30
Initial Connection Weight	
Hidden → output	0.0
External output → hidden	random from -0.5 to 0.5
Self feedback	4.0
Other feedback	0.0
Learning rate	
For Feedback connections (3 × 3 task)	0.1 → 0.05
For other connections (3 × 3 task)	0.2 → 0.1
For all connections (branch task)	0.5 → 0.25



(a) environment 1 (b) environment 2 (c) environment 3

Fig. 2. Simulation environments and agent's behaviors after learning for the cases of sample three wall allocations. The wall allocation is randomly chosen at the beginning of each episode under the condition that the agent can visit all the squares. The small numbers indicate what number of step the action is in the episode.

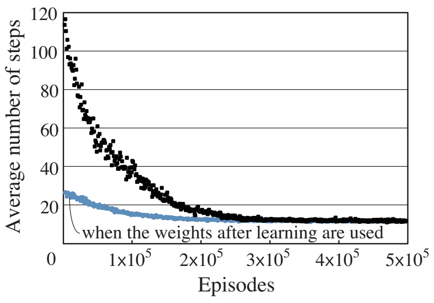
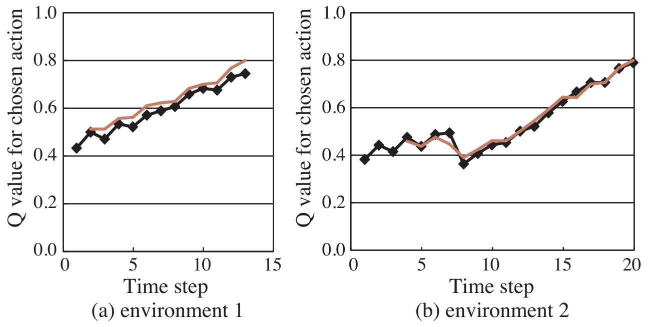


Fig. 3. Learning curve in the 3 × 3 maze task. Since the curve includes the influence of random exploration, the performance change only by the exploration factor is plotted by using the connection weights after learning to see the effect of learning.



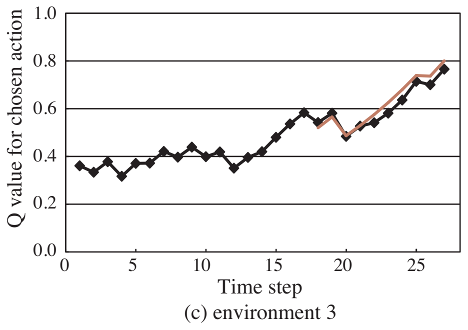
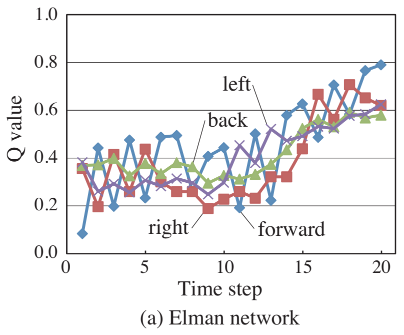


Fig. 4. Change of the maximum Q-value in one episode after learning for the 3 cases in Fig. 2. The line with no plot marker indicates the ideal Q-value.



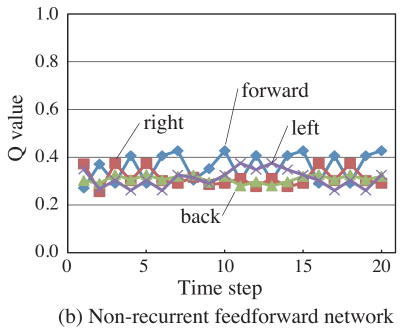


Fig. 5. Comparison of the Q-value change in the case of the environment 2 between the cases of an Elman network and a non-recurrent feedforward network.

only at the steps close to the final step. It is seen that the Q-value is quite similar to the ideal value in all the three cases. In the environment 1, although the agent cannot get a reward when it turns the direction, the agent can reach the last square with no turnaround. Therefore, the possibility that the next square is the goal becomes larger gradually; it is $1/8 = 12.5\%$ at first, and 100% when the agent visits the last square. It is seen that the actual Q-value after learning increases almost monotonously and finally it reaches around 0.8 that is the reward value. In the environment 2, the agent has to turn around at one end of the path, and during some steps after the turn, it has to passes 4 squares with no possibility of the goal existence before it reaches the next square with a possibility of the goal. Therefore, the Q-value increases at first, but it decreases suddenly at the 8th step, and it increases again.

In the environment 3, at the center square, the agent can choose one among 4 moving directions. Because the change of direction spends one more step, the agent, in the optimal path, should go straight when it returns to the center square for the first time, change its direction for the second time, and go straight for the third time. However, the agent took the path as shown in Fig. 2 (c). It takes 27 steps, while 25 steps in the case of the optimal path. The maximum number of steps to visit all the 9 states when the wall allocation is varied is 28, while 26 steps in the case of the optimal path.

10 simulation runs with a different random sequence for initial connection weights and random exploration are performed. There is no case in which the agent can always take the optimal action. In 2 of the 10 runs including the above result, the maximum number of steps is 28. In 5 of the 10 runs, the maximum number of steps is less than 40. In 1 of the 10 runs, the maximum number of steps is more than 100, but less than 1000.

When a non-recurrent regular 3-layer neural network is used, in all of 10 simulation runs with a different random sequence, the agent falls into the infinite loop in the environment 3 after learning. In fact, for all the cases of wall allocation, the agent must be able to take non-optimal but appropriate path without using any memories, which we call ”wall following”. It might occur that inaccurate Q-values due to the state confusion disturb to choose appropriate actions.

Figure 5 shows the change of all the Q-values in one episode in the environment 2 for both recurrent and non-recurrent neural network cases. It is seen that in both cases, the action selection is appropriate, but in the case of non-recurrent neural network, the Q-values do not increase as time goes by. In contrast, in the case of recurrent neural network, the change of Q-value seems to consider the possibility that a goal appears. When the agent visits the last 9th square, it is always the goal state and gets the reward 0.8. It is interesting that the agent seems to know the visit of the final 9th square even though the agent perceives only wall assignment around it and the previous action.

C. Behaviors in Some Unknown Environments

Next, the agent after learning of 3 $\times$ 3 maze was put on some unknown environments, and the agent's behaviors were observed to examine how the acquired knowledge is generalized to unexpected environments. At first, the agent was put on the environment where 9 squares are arranged in a row. The agent behavior when it was located initially at the second square from the left end is shown in Fig. 6, and the change of Q-value for the chosen action is shown in Fig. 7.

It is seen that the agent goes straight to the right end of the row, turns around at the end, and then goes straight again until it reaches the unvisited square at the left end of the row. The Q-value tends to increase at first, and when it reaches the right end, the Q-value decreases suddenly. After that, the Q-value increases again until it reaches the last unvisited square.

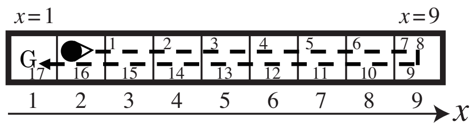


Fig. 6. The agent behavior in the environment where 9 squares are allocated in a row. The agent has not experienced the environment during learning.

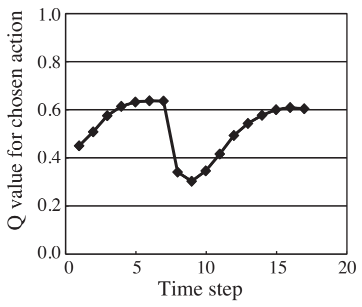


Fig. 7. Change of the maximum Q-value in one episode after learning for the environment in Fig. 6.

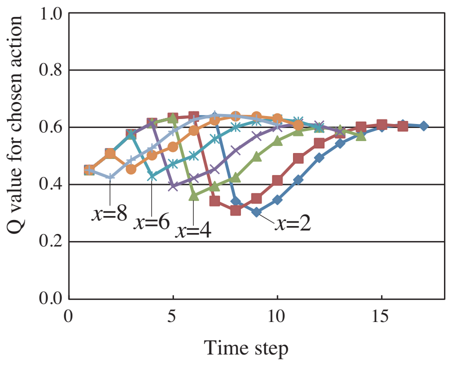


Fig. 8. Comparison of Q-value Change for chosen action when the initial agent location is varied.

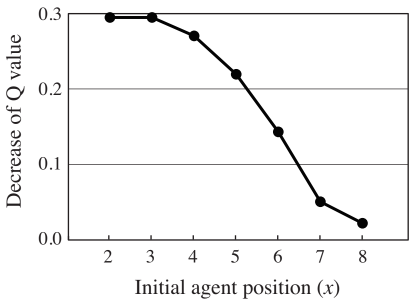


Fig. 9. Change of the Q-value decrease at the right end in Fig. 6 according to the initial agent location.

Although the Q-value does not reach 0.8, but the trend of the change seems rational because at the right end, the goal will not appear for some steps. It is also rational that the decrease of Q-value is larger than that in the environment 2, as shown in Fig. 4(b). In the environment 2, when the agent goes to the bottom-right square, the possibility that the square is the goal is 20%, while in this case, the possibility is 50% because only

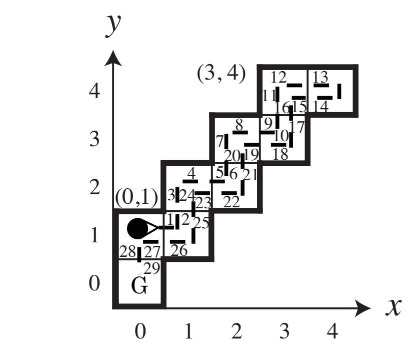


Fig. 10. The agent behavior in the environment where 9 squares are allocated in a zigzag manner. The agent has not experienced the environment during learning.

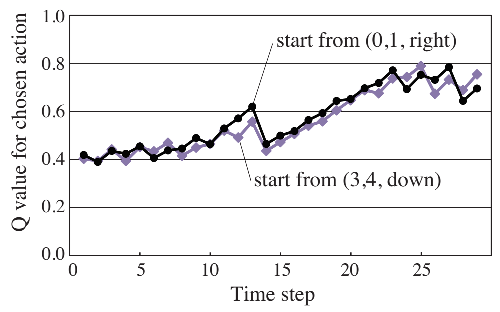


Fig. 11. Change of the maximum Q-value in one episode after learning for the environment in Fig. 10.

one square remains at the left end if the right end square is not the goal.

Then, to examine whether or not the decrease of Q-value at the right end considers the past state transition even in an unknown environment, the decrease of Q-value is observed when the initial location of the agent changes. Figure 8 shows the change of Q-value for the 7 cases of the initial agent locations. In all the cases, the Q-value decreases at the right end, and increases after that. Figure 9 shows the decrease of Q-value as a function of the initial agent x position. It is seen that the decrease of Q-value decreases as the initial location is closer to the right end. If the initial location is closer to the right end, the possibility of the goal at the right end decreases, and the number of steps to reach the next square with a possibility of goal from the right end also decreases. Accordingly, it seems that the agent considers the past state transition to evaluate its present situation.

Secondly, 9 squares are arranged in a zigzag manner as shown in Fig. 10. The behavior that the agent takes when it starts from the square at $(0, 1)$ with the direction "right" is also shown in Fig. 10. The change of the maximum Q-value in one episode for that case is shown in Fig. 11. In this figure, the change of Q-value when the agent starts from the square at $(3, 4)$ with the direction of "down" is also plotted.

The agent behavior is appropriate. The decrease of Q-value at the upper-right end is smaller than expected, but the Q-value increases gradually before and after the turnaround.

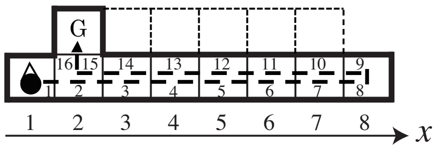


Fig. 12. Simulation environment with a random branch and the agent's behaviors after learning.

Particularly, the way of increase is interesting. Before the turnaround at the upper-right end, when the agent visits a new square, there is a possibility that the square is the goal state, but when it selects the "turn right" action, there is no possibility of the goal state. If the agent knows that, the Q-value for the action to go to a square with a possibility of goal is large, and the Q-value becomes smaller after it reaches the square but no reward. That matches the actual Q-value change with up and down before the turnaround as shown in Fig. 11. On the other hand, after the turnaround, for any actions, there is no possibility of goals for some time steps because the squares have been visited already. The Q-value is expected to increase monotonously towards the final goal. That also matches the actual change in Q-value after the turnaround even though unexpected up and down appears after the 22nd step. It seems that the agent knows that the goal does not appear for some while after the turnaround at the upper-right end.

When the agent starts from the square at $(3, 4)$, the inputs of the neural network before the turnaround are the same as those after the turnaround when the agent starts from the square at $(0, 1)$. In the same way, the inputs before the turnaround when it starts from the square at $(0, 1)$ are the same as those after turnaround when it starts from the square at $(3, 4)$. Nevertheless, before the turnaround, the Q-value increases with up and down, and while after the turn, it increases monotonously in both cases. The same test is executed in the other simulation run in which the performance in 3×3 maze environment is better, a similar tendency is observed, although it is not so clear as this result.

D. Branch-position memorization task

In order to investigate the acquisition of memory function more, the agent learned in a more simple environment as shown in Fig. 12. In this environment, there are a row of 8 squares and one branch square. The agent starts from the left-end square facing the up direction, and the location of the branch is chosen randomly at each episode between the 2nd and 7th squares, as shown in Fig. 12. The goal state is located randomly on one of the squares except for the left end square. All the parameters are the same as in the previous task except for the learning rate. The learning rate is larger because the learning is more stable and precise Q-value is required to show the acquisition of memory function clearly.

Figure 12 also shows the agent behavior when the branch is located at $x = 2$. At the first glance, the optimal agent behavior seems visiting the branch square at first, but to visit the branch, many turns are needed to change the agent's

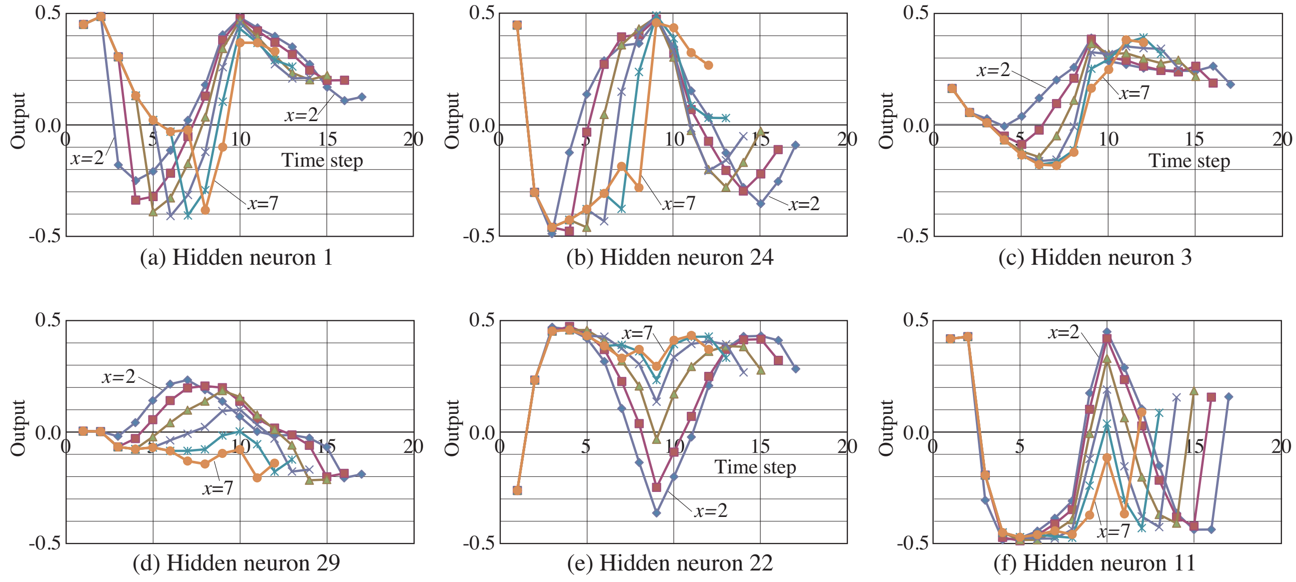
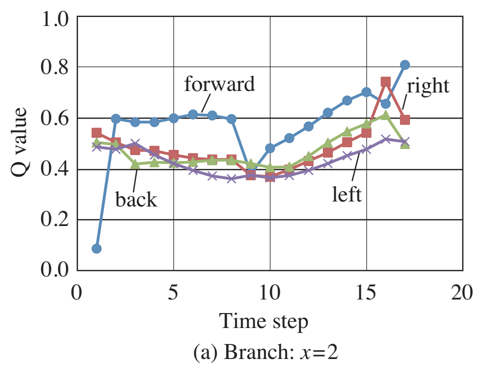


Fig. 16. Comparison of the output change due to the difference of the branch location in some hidden neurons.



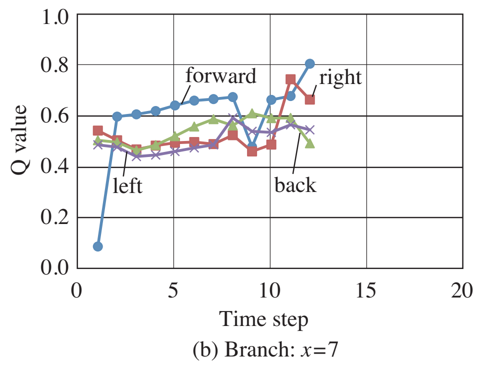


Fig. 13. Change of Q-values for all the actions for two cases of branch positions in the branch task.

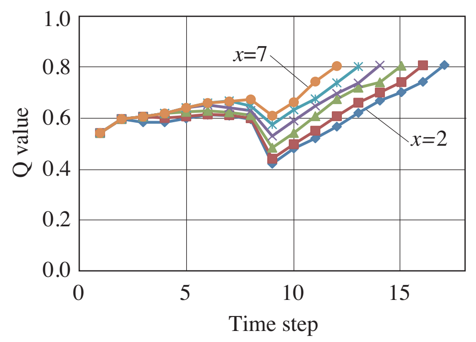


Fig. 14. Comparison of the Q-value change due to the difference of the branch location

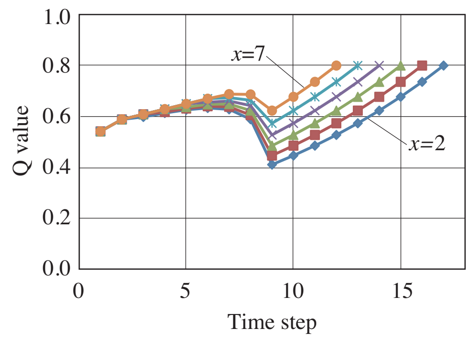


Fig. 15. Ideal Q-value change due to the difference of the branch location.

direction. Accordingly, in the case of discount factor $\gamma = 0.92$, the optimal path from the viewpoint of cumulative discounted rewards is to go straight at first, and if the goal exists on the branch square, it turns around at the right end not depending on the branch location.

Figures 13 shows the change of all the Q-values after learning when the branch exists at $x = 2$ or $x = 7$ respectively. After turning right at the initial location, the Q-value for “moving forward” is the maximum until it reaches the right end of the row. At the 9th step, at the right end, the Q-value for the forward motion is decreased, but after that, it increases before the branch comes again. At the branch, the Q-value for ”turn right” becomes maximum, and the Q-value for the final action is almost 0.8.

Figure 14 shows the change of Q-value of the chosen action for the 6 cases of branch location. Figure 15 shows the ideal Q-value change also for the 6 cases. It is seen that the actual Q-value change is quite similar to the ideal one. If the agent does not reach the goal before reaching the right end of the environment, the Q-value decreases once at the 9th step. However, it is interesting that the Q-value at the 9th step is larger as the branch position is closer to the right-end although the present external inputs are completely the same. This means that the agent learned to memorize the branch position in the recurrent neural network and to reflect it to the Q-value. In this case, just to realize the optimal actions, no memory is necessary. Actually, with a non-recurrent neural network, the optimal behaviors can be achieved in this task. This means that the branch position is memorized to obtain precise Q-values.

When the 40 hidden neurons are observes, some neurons that seems to contribute the memory of the branch location were found. Figure 16 shows some examples. The hidden 1 (a) and hidden 24 (b) neurons responds to the branch location, but they becomes almost a same value when it reaches the right end at the 9th step. The hidden 3 (c) and hidden 29 (d) neurons seem to keep the values for a while. In the hidden 22 (e) and hidden 11 (f) neurons, the difference of the output at the branch is not so large, but at the right-end, the outputs represent the branch location. Before learning, the agent did not know that memorization of branch location is necessary to calculate accurate Q-values. It also does not know how the branch location is memorized and how the memorized information is reflected to the Q-values. However, the agent autonomously learns to memorize the branch location and to reflect it to the Q-values.

IV. Conclusion

In the deterministic exploration task with invisible goal state, the agent acquired appropriate behaviors through reinforcement learning with a recurrent neural network though the optimal behavior cannot be obtained. The Q-values after learning was similar to the ideal one that cannot be obtained with a non-recurrent feedforward neural network. The agent after learning seems to understand that after the turnaround at the end of the environment, the goal state does not appear for some steps even in some unknown environments. Furthermore, it was also shown that the memorization of necessary non-binary information such as the branch location emerges in the recurrent neural network to output accurate Q-values only through the learning from rewards.

Acknowledgment

This research was supported by JSPS Grant in-Aid for Scientific Research #19300070.

References

[1] K. Shibata: Acquisition of Deterministic Exploration Behavior by Reinforcement Learning, Proc. of the 11th Int'l Symp. on Artificial Life and Robotics (AROB) (2006)

[2] K. Shibata: Learning of Deterministic Exploration and Temporal Abstraction in Reinforcement Learning, Proc. of SICE-ICCAS, pp.4569-4574 (2006)

[3] K. Goto and K. Shibata: Learning of Exploration Behaviors Utilizing Memory by Reinforcement Learning with a Recurrent Neural Network, Record of Joint Conf. of Electrical & Electronic Engineers in Kyushu (2008) (in Japanese)

[4] Sebastian Thrun: Efficient Exploration In Reinforcement Learning, Tech. Report CMU-CS-92-102, Computer Science Department, Carnegie Mellon University (1992)

[5] Gang Zhao, Shoji Tatsumi & Ruoying Sun: RTP-Q: A Reinforcement Learning System with an Active Exploration Planning Structure for Enhancing the Convergence Rate, Proc. of IEEE SMC'99, fp063.pdf, pp. V-475-480 (1999)

[6] B. Bakker, V. Zhumatiy, G. Gruener, and J. Schmidhuber, A Robot that Reinforcement-Learns to Identify and Memorize Important Previous Observations Proc. of IROS 2003, 430-435 (2003)

[7] H. Utsunomiya and K. Shibata, "Contextual Behavior and Internal Representations Acquired by Reinforcement Learning with a Recurrent Neural Network in a Continuous State and Action Space Task", Advances in Neuro-Information Processing, Lecture Notes in Computer Science, Vol. 5507, pp. 970-978, 5507-0970. pdf (CD-ROM) (2009)

[8] K. Shibata: Discretization of Series of Communication Signals in Noisy Environment by Reinforcement Learning, Ribeiro et al(eds.), Adaptive and Natural Computing Algorithms, pp. 486-489 (2005)

[9] K. Goto, K. Matsumoto and K. Shibata: Emergence of prediction ability by reinforcement learning using a recurrent neural network, Proc. of SICE Kyushu Annual Conf. (2009) (in Japanese)

[10] K. Goto and K. Shibata: Emergence of prediction by reinforcement learning using a recurrent neural network, Journal of Robotics, Vol. 2010, Article ID437654 (2010) (to appear)

[11] Watkins, C.J.C.H. and Dayan, P.: Technical Note:Q-Learning, Machine Learning, Vol.8, pp.279-292 (1992)

[12] R. S. Sutton and A. G. Barto, Reinforcement Learning, MIT Press (1998)

[13] Rumelhart, D.E, Hinton, G.E., and Williams, R.J.: Learning Internal Representations by Error Propagation, Parallel Distributed Processing, The MIT Press, pp. 318-362 (1986)