Q-learning with exploration driven by internal dynamics in chaotic neural network

1st Toshitaka Matsuki
Department of Innovative Engineering
Faculty of Science and Technology
Oita University
Oita, Japan
matsuki@oita-u.ac.jp

2nd Souya Inoue
Department of Innovative Engineering
Faculty of Science and Technology
Oita University
Oita, Japan

3rd Katsunari Shibata
Department of Innovative Engineering
Faculty of Science and Technology
Oita University
Oita, Japan
katsunarishibata@gmail.com

***Abstract*—This paper shows chaos-based reinforcement learning (RL) using a chaotic neural network (NN) functions not only with Actor-Critic, but also with Q-learning. In chaos-based RL that we have proposed, exploration is performed based on internal dynamics in a chaotic NN and the dynamics is expected to grow rational through learning. Q-learning is a very popular RL method and widely used in several researches. We focused on whether Q-learning can be adopted to chaos-based RL. Then we demonstrated the agent can learn a goal task in a grid world environment with chaos-based RL using Q-learning. It was also shown that, as learning progresses, irregularity in the network outputs originated from the internal chaotic dynamics decreases and the agent can automatically switch from exploration mode to exploitation mode. Moreover, it was confirmed that the agent can adapt to changes in the environment and automatically resume exploration.**

***Index Terms*—Q-learning, exploration, chaotic neural network, chaos-based reinforcement learning, reservoir network**

I. Introduction

Deep Learning (DL) breakthroughs have shown a deeply layered Neural Network (NN), which is trained with a large amount of data, can surpass existing image recognition systems [1] [2] [3]. DL has also been applied in several engineering fields, and many studies have identified DL as a superior approach to handcrafted systems. More recently, DL has been applied to series data processing such as speech recognition or natural language processing [4] [5] [6]. Therefore, recurrent NNs (RNN), which have a recursive structure, have been focused on. The successes of DL suggest phenomenal ability of parallel processing in the brain and difficulty in understanding and handcrafting such a massively parallel system.

Reinforcement learning (RL) is a machine learning approach in which an agent learns without any direct teacher [7]. In RL, an agent learns in a task environment by trial-and-error or exploration driven by external random noise. For a long time, End-to-End RL approach has been studied [8] [9] [10]. In End-to-End RL, an agent learns the entire process from raw input signals to action output based on RL in an NN as a parallel learning system. Mnih et al. showed that an NN can play Atari games successfully and sometimes beat human experts with RL [11]. It was also shown that an RNN can learn tasks that require memory in which past information is integrated through time with RL [12] [13] [14]. In End-to-End RL, an NN automatically learns necessary functions as a process to generate action outputs from raw input signals through the interaction with the environment rather than given data sets. This approach is expected as a key to realize artificial general intelligence including higher functions.

In the general approach using RNN, the network requires convergence dynamics to an appropriate state to extract and memorize necessary features from the past inputs and to generate appropriate outputs. Differing from the RNN usage, the biological brain is not driven only by inputs. We consider that the state of the brain autonomously and rationally transitions as internal dynamics. We expect that organizing such dynamics in an artificial NN through learning must be a key to develop a thinking machine.

Freeman investigated chaotic dynamics that emerges in olfactory system of a rabbit [15]. In this research, it is found that olfactory bulb and cortex have many chaotic attractors that the system settles into when rabbits are held under the influence of particular odorant stimulus. When the rabbit establishes a new conditioned response to a new odorant, reorganization of existing attractors is observed and a new attractor corresponding to the new odorant is added after the emergence of chaotic dynamics. Freeman insisted that the chaos in the brain continually produces novel activity patterns that are critical to develop new nerve cell assemblies. Osana et al. demonstrated a numerical simulation of self-associative memory with a chaotic NN model that shows the similar phenomenon to the dynamics that is observed in biological experiment [16]. Freeman suggested that the ability to create activity patterns may underlie the brain's ability to generate insights and trials in trial-and-error problem solving [17].

In order to realize both convergence and transition internal dynamics in a network, which look incompatible with each other, we have investigated an approach of training an NN that has chaotic dynamics. We have set up a hypothesis that the exploration driven by disordered and autonomous state transition in a chaotic NN could develop into rational transition among multiple states through learning and the network could acquire dynamics like thinking [18]. On the hypothesis, we

This work was supported by JSPS KAKENHI Grant Number JP15K00360.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

have proposed chaos-based RL in which exploration is driven by chaotic dynamics in an NN and necessary functions are acquired based on reward and punishment [19]. It has been demonstrated that a fully connected RNN having chaotic dynamics by introducing strong interconnection weights or neural refractoriness can learn an easy goal task or avoiding obstacle task with the Actor-Critic method that is a kind of RL method [20] [21]. In chaos-based RL, it was confirmed that internal chaotic dynamics initially drives exploration and the network can establish ordered dynamics to carry out a task as learning progress. Additionally, the network can automatically resume exploration when the agent meets changes in the environment. In other words, it was shown that the agent can adapt to changes in the environment and automatically switch between exploration and exploitation modes.

As well as Actor-Critic, Q-learning is a widely used popular RL method as in Deep Q-network [11]. Q-learning is a simple learning for discrete actions, and an agent learns action values for the sets of state and action. In this paper, we demonstrate that chaos-based RL can be performed with Q-learning method. Additionally, we show that an agent can automatically switch between exploration and exploitation modes with chaos based Q-learning.

Here, we employ a reservoir network (RN) that has chaotic dynamics instead of a fully connected RNN. RN was proposed as an echo state network (ESN) by Jaeger [22] or a liquid state machine by Maass [23], respectively. Its hidden layer is RNN called reservoir whose recurrent connections are randomly and sparsely connected. An RN can learn series data processing easily and stably with modifying only output layer called readout unit (RU). Hoerzer et al. showed that a chaotic RN can learn through reward-modulated Hebbian learning. This suggests that an RN can learn several tasks through exploration driven by external random noise and a sequential reward [24]. We demonstrated that an RN can learn a working memory task based on its internal chaotic dynamics and rewards, and can switch between exploration and exploitation modes automatically [25].

The remainder of this paper is organized as follows. We introduce the chaos-based RL and the experimental method in Section II. We present simulation results in Section III. Finally, we conclude these results in Section IV.

II. Method

A. Network

In this study, we employed an echo state network (ESN). Fig. 1 shows the network structure. This network consists of an RNN called reservoir and an output layer called readout unit. The reservoir has $N_{\mathrm{x}} = 200$ neurons. The network receives $N_{\mathrm{i}} = 121$ inputs as a vector $\boldsymbol{u}_t \in \mathbb{R}^{N_{\mathrm{i}}}$, and generates $N_{\mathrm{o}} = 4$ outputs as a vector $\boldsymbol{z}_t \in \mathbb{R}^{N_{\mathrm{o}}}$. The outputs $\boldsymbol{z}_t$ are fed back to the reservoir neurons. Then the internal state vector at time t is expressed as follows:

$$\boldsymbol{x}_t = g\boldsymbol{W}^{\mathrm{rec}}\boldsymbol{r}_{t-1} + \boldsymbol{W}^{\mathrm{in}}\boldsymbol{u}_t + \boldsymbol{W}^{\mathrm{fb}}\boldsymbol{z}_{t-1}. \tag{1}$$

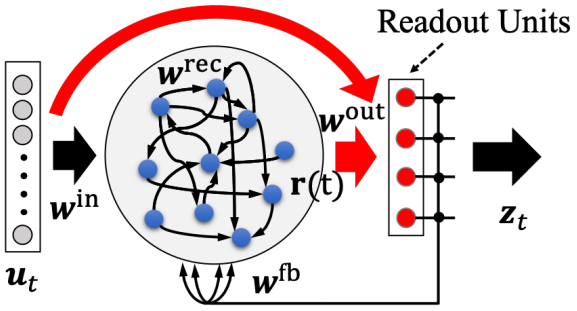


Fig. 1. Echo State Network (ESN). An ESN has a special hidden layer called "reservoir" whose neurons are randomly and sparsely connected and fixed. The output layer is called "readout units". The inputs are fed to reservoir and readout units, and the network outputs are generated as linear combinations of reservoir activity and network inputs. The network has a feedback pathway, and outputs are fed back to the reservoir.

The initial internal state $\boldsymbol{x}_0$ is set to zero vector. $\boldsymbol{W}^{\mathrm{in}} \in \mathbb{R}^{N_{\mathrm{x}} \times N_{\mathrm{i}}}$ is the connection weight matrix from the input to the reservoir. $\boldsymbol{W}^{\mathrm{fb}} \in \mathbb{R}^{N_{\mathrm{x}} \times N_{\mathrm{o}}}$ is the feedback weight matrix from the network output to the reservoir. $\boldsymbol{W}^{\mathrm{rec}} \in \mathbb{R}^{N_{\mathrm{x}} \times N_{\mathrm{x}}}$ is the recurrent connection weight matrix in the reservoir. The reservoir output vector $\boldsymbol{r}_t$ is expressed as follows:

$$\boldsymbol{r}_t = \tanh(\boldsymbol{x}_t) \tag{2}$$

$\boldsymbol{W}^{\mathrm{in}}$ and $\boldsymbol{W}^{\mathrm{fb}}$ are given randomly from a uniform distribution between -1 and 1 and their connection probability is 0.5. $\boldsymbol{W}^{\mathrm{rec}}$ is generated randomly from a uniform distribution with connection probability 0.1 and divided by its maximal absolute eigenvalue $\rho(\boldsymbol{W}^{\mathrm{rec}})$ (Spectral Radius) so that it becomes a matrix with a unit spectral radius. The normalized $\boldsymbol{W}^{\mathrm{rec}}$ is scaled with $g = 1.4$, and consequently, the spectral radius is given by g. The fact is well known that the reservoir dynamics is ordered with $g < 1$, whereas it is chaotic with $g > 1$ and the larger g is, the more irregular the chaotic activity is.

The network output $\boldsymbol{z}_t$ is given with the reservoir outputs $\boldsymbol{r}_t$ and the input $\boldsymbol{u}_t$ as follows:

$$\boldsymbol{z}_t = \boldsymbol{W}^{\mathrm{out}} \begin{pmatrix} \boldsymbol{u}_t \\ \boldsymbol{r}_t \end{pmatrix}, \tag{3}$$

where $\boldsymbol{W}^{\mathrm{out}} \in \mathbb{R}^{N_{\mathrm{o}} \times (N_{\mathrm{i}} + N_{\mathrm{x}})}$ is a fully connected weight matrix from the input and reservoir to the readout units, which is indicated as red arrows in Fig. 1. $\boldsymbol{W}^{\mathrm{out}}$ is initialized randomly from a uniform distribution between -0.01 and 0.01.

In the reservoir computing approach, the readout weight $\boldsymbol{W}^{\mathrm{out}}$ is only modified to learn generating appropriate outputs. Note that, although the RN is a kind of RNN and is generally used to process series data, we use the RN as a network that has chaotic dynamics in this study. We use supervised learning to modify $\boldsymbol{W}^{\mathrm{out}}$, and then we define the loss function as follows:

$$\boldsymbol{E}(\boldsymbol{u}_t, \boldsymbol{r}_t; \boldsymbol{W}^{\mathrm{out}}) = \frac{1}{2}(\boldsymbol{T}_t - \boldsymbol{z}_t)^2 \tag{4}$$

where $\boldsymbol{T}_t$ is the training signal vector that is generated automatically based on Q-learning. $\boldsymbol{W}^{\text{out}}$ is modified to minimize the loss function with gradient decent as follows:

$$\begin{aligned}\boldsymbol{W}^{\text{out}} \leftarrow & \boldsymbol{W}^{\text{out}} - \eta \nabla_{\boldsymbol{W}^{\text{out}}} \boldsymbol{E}(\boldsymbol{u}_t, \boldsymbol{r}_t; \boldsymbol{W}^{\text{out}}) \\ = & \boldsymbol{W}^{\text{out}} - \eta(\boldsymbol{T}_t - \boldsymbol{z}_t) \begin{pmatrix} \boldsymbol{u}_t \\ \boldsymbol{r}_t \end{pmatrix}^{\text{T}}\end{aligned} \tag{5}$$

where $\eta = 0.01$ is the learning rate that modulates the step size of the weights update.

B. Task

Here, an agent learns to clear a simple goal task in grid world environment. The outline of the task is shown in Fig. 2. At the beginning of one episode, the agent is randomly placed on one of the four start cells in 11×11 grid world. Then, at each step, the network receives agent state input $\boldsymbol{u}_t$ and generates output vector of action values $\boldsymbol{z}_t = [Q_t^1, \cdots, Q_t^4]^{\text{T}}$ whose elements are corresponding to four actions "go north", "go east", "go south", "go west", respectively. When the agent is at (i, j) in the grid world, the input vector $\boldsymbol{u}_t$ is given as only one corresponding input to the agent location is 1 as follows:

$$\begin{aligned}\boldsymbol{u}_t &= [u_t^1, u_t^2, \cdots, u_t^k, \cdots, u_t^{121}]^{\text{T}} \\ u_t^k &= \begin{cases} 1, & k = 11(i-1) + j \\ 0, & otherwise \end{cases}\end{aligned} \tag{6}$$

Action selection is not stochastic, but always completely greedy. Then, an agent action at time t is expressed as follows:

$$a_t = \arg\max_a Q_t^a \tag{7}$$

In general RL, to find actually better actions, an agent sometimes explores with stochastic selection by taking an action that is currently estimated not to be best. For example, in ε-greedy policy, the agent takes random action with probability $0 \leq \varepsilon \leq 1.0$. Such exploration in RL is driven by random numbers generated outside the learner agent. Meanwhile, in chaos-based RL, exploration is driven by internal chaotic dynamics in a chaotic NN. Therefore, the agent always acts with the highest action value, without any stochastic action selection.

The agent moves to the next cell in the direction according to a_t, the agent state changes from s_t to s_{t+1} and the agent receives a reward $r_{t+1}(s_t, a_t)$. This one cycle of interaction between an agent and an environment is referred to as 1 step. When an agent moves to a wall cell or outside the grid world by taking the action a_t, the agent state does not change (i.e. $s_{t+1} = s_t$) and the agent receives $r_{t+1} = -0.1$. Meanwhile, when the agent reaches the goal cell, it gets a reward $r_{t+1} = 1.0$. When the agent reaches the goal or 200 steps are passed, one episode ends and the agent is placed on one of the start cells. Then the reservoir internal state $\boldsymbol{x}_0$ is set to zero vector and new episode starts.

In this study, to examine whether the agent can adapt to changes in the environment and automatically switch between exploration and exploitation modes, action-output correspondence is shifted on the middle of learning. More concretely, the cross arrow in Fig. 2, which shows the correspondence between action value and moving direction, rotates 90 degrees in a counter-clockwise direction.

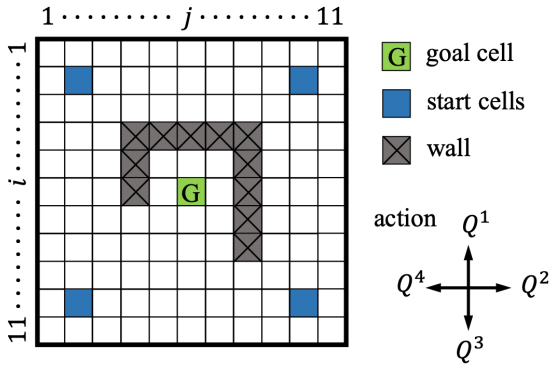


Fig. 2. The 11×11 grid world environment. Each cell corresponds to the state of the environment and can be referred by row column numbers (i, j). At the beginning of an episode, the agent is randomly placed at the "start cell". In one step, the agent can move one cell in four directions north, south, east and west. The agent cannot be on the "wall" and outside the world. The agent moves there and a punishment is given. If the agent reaches the "goal cell", the episode ends with a reward.

C. Q-learning

The agent network is trained by Q-learning. When the agent takes the action a_t at state s_t and gets a reward r_{t+1} at next state s_{t+1}, the action value of a_t is evaluated as follows:

$$\hat{Q}(\boldsymbol{s}_t, a_t) = r_{t+1} + \gamma \max_a Q(\boldsymbol{s}_{t+1}, a) \tag{8}$$

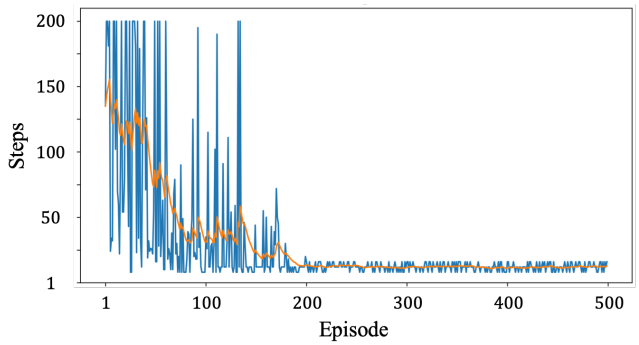
Based on this evaluated value, the action value function $Q(s_t, a_t)$ is updated as follows:

$$\begin{aligned}&Q(\boldsymbol{s}_t, a_t) \\ &\leftarrow Q(\boldsymbol{s}_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(\boldsymbol{s}_{t+1}, a) - Q(\boldsymbol{s}_t, a_t) \right] \\ &\quad = (1-\alpha) Q(\boldsymbol{s}_t, a_t) + \alpha \hat{Q}(\boldsymbol{s}_t, a_t)\end{aligned} \tag{9}$$

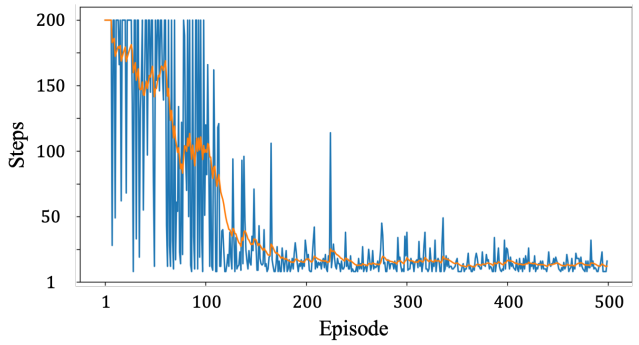
where, $0 \leq \alpha \leq 1.0$ is the step-size parameter, which influences the rate of learning. γ is a discount rate that determines the present value of discounted future rewards. In this paper, the episode ends when the agent reaches the goal, therefore, the ideal action value with which the agent reaches the goal equals to $r_{t+1} = 1.0$. Employing this algorithm to train the network in this study, the training signal for Q^{a_t} is given as follows:

$$T_t^{a_t} = r_{t+1} + \gamma \max_a Q_{t+1}^a \tag{10}$$

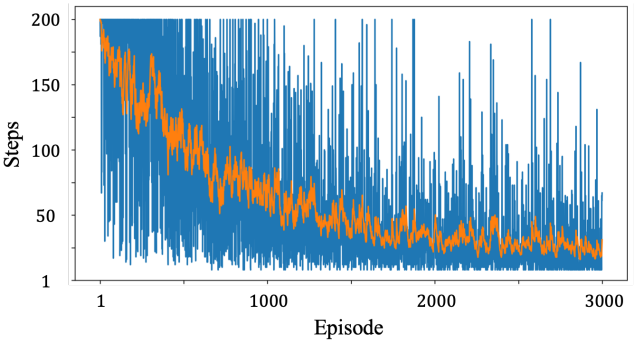
Training signals for outputs other than Q^{a_t} are generated as the errors are 0, and then, $\boldsymbol{W}^{out}$ is updated with (5). If the agent moves according to the maximum action value selected to generate the training signal of previous step in (10), the learning procedure equals to the SARSA method.



(a) Explorating with internal chaos dynamics



(b) Explorating with external random numbers

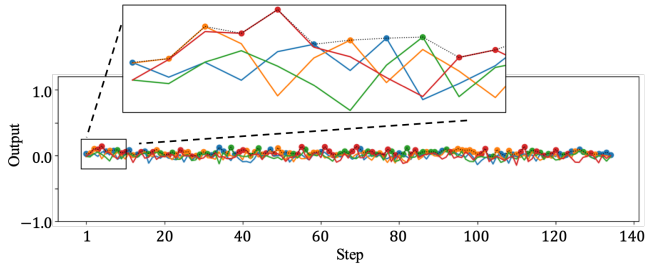


(c) The reservoir is replaced with random vector

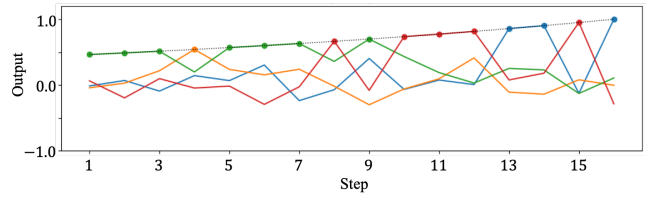
Fig. 3. Learning curve indicating the number of steps to the goal in each training episode. Orange line is a moving average of the numbers. The reduction of the number shows successful learning of the agent.

III. Result

Fig. 3 (a) shows the learning curve for 500 training episodes. The horizontal axis indicates the number of episodes and the vertical axis indicates the number of steps at each episode. As seen in the figure, the number of steps reduced as the learning episodes proceeded, and that shows the agent successfully learned. This result suggests that a chaotic NN can learn through Q-learning with exploration driven by its internal chaotic dynamics.



(a) episode 1



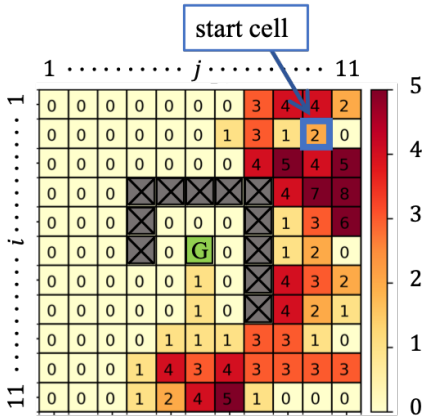
(b) episode 500

Fig. 4. Network output change in one episode for the cases of before learning (episode 1) and after learning (episode 500). Each color indicates an action as Q^1:blue, Q^2:orange, Q^3:green and Q^4:red. Plotted points and dotted line indicate the highest output at each step.

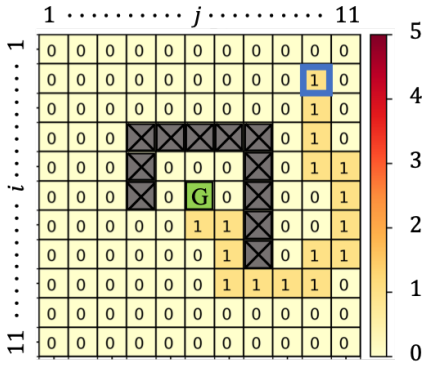
The action values that the agent network generates at the episode 1 and 500 are shown in Fig. 4 and the frequency of agent visits at the episode 1 and 500 are shown in Fig. 5. As shown in Fig. 4 (a), for early episodes, the action values chaotically fluctuate due to its chaotic internal dynamics in the network as shown in Fig. 6, and the agent seems to take actions almost randomly. Furthermore, as shown in Fig. 5 (a), the chaotic network outputs realized wide exploration of the agent in the grid world.

Meanwhile, as shown in Fig. 4 (b), the outputs became ordered and generated outputs that are almost ideal action values after learning. Fig. 5 (b) shows the route to the goal that the agent moved along. The agent successfully moved to the goal, however, we can see an unnecessary move at $(i, j) = (5, 10)$ by which the agent needs 2 extra steps than the optimal. This can be considered that the exploration was finished because of the reduction of the chaoticity before the agent had explored sufficiently in the environment.

To compare the results between random number exploration and chaos exploration, we show the training result of the same task by a feed forward NN trained with ε-greedy Q-learning (fixed at $\varepsilon = 0.2$) in Fig. 3 (b). Here, the network has a hidden layer consisting of 200 neurons, whose activation function is $\tanh$ function, and neurons in the output layer generate outputs as linear combinations of hidden layer outputs. The network is updated with back propagation algorithm (learning rate $\eta = 0.1$) to reduce the loss function as in (4). The network succeeded to learn the task. Comparing Fig. 3 (a) and (b), a difference can be seen after reducing the number of steps until around 200 episodes. That is, in (a), the number of steps



(a) episode 1



(b) episode 500

Fig. 5. Frequency of agent visits before learning (episode 1) and after learning (episode 500). In both cases, start position is $(i, j) = (2, 10)$ (indicated by blue box).

changes by randomness of start positions in (a), meanwhile, in (b), it fluctuates more hardly than (a). It can be considered that as the learning progressed, the chaotic outputs became ordered and the exploration automatically ended in case (a) in which the exploration is driven by internal chaotic dynamics in a chaotic NN. On the other hand, in case (b), the agent continued to explore with ε-greedy action selection until the end of the final episode. When we adjust the ε value appropriately as the learning progresses, the agent stops exploring. However, the exploration in chaos based RL can be characterized by this "automatic" reduction of exploration component through learning.

Here, we wonder whether the reservoir plays the role as only a random number generator. Then, we examine the task learning with an agent network whose reservoir is replaced with N_x dimensional uniform random number vector from -1 to 1. The learning curve is shown in Fig. 3 (c). Although the number of steps reduced as the learning progressed, it was reduced more slowly than the other cases (Fig. 3 (a) and (b)) and it had remained large and fluctuating. This result suggests that the dynamics in the reservoir is chaotic dynamics that reflects the influences of inputs from environment or learning

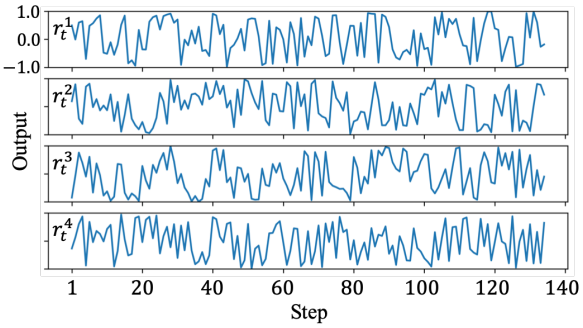
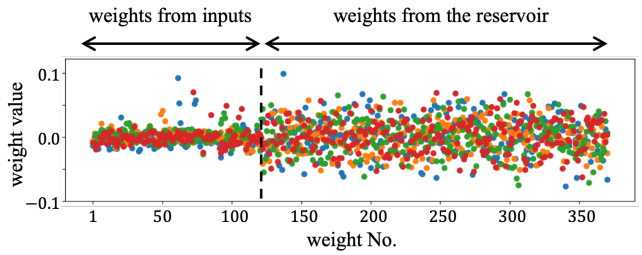


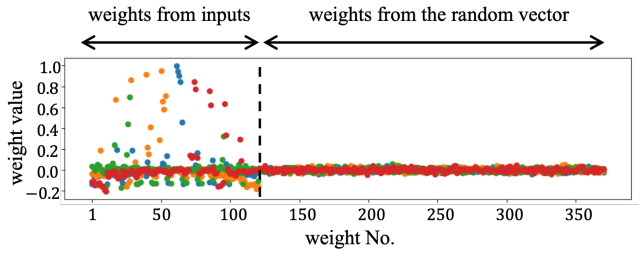
Fig. 6. Outputs of the reservoir. These lines show the values of $r_t^1, \cdots, r_t^4$ at episode 1, respectively.

of the network rather than only random dynamics. To clarify the difference of learning between the cases (a) and (c), the training result of readout weights is shown in Fig. 7. As can be seen in Fig. 7 (a), the weights from reservoir are larger than the weights from inputs in the case of exploration driven by internal chaotic dynamics in the reservoir. This result suggests that the readout units learned to generate outputs extracting the outputs from the reservoir. Meanwhile, in the case of random vector, as shown in Fig. 7 (b), the weights for the inputs vector $\boldsymbol{u}_t$ are much larger than the weights from random vector. The reason can be considered that there is no correlation between the random numbers and the ideal outputs. These results supports the hypothesis that the reservoir is not only a generator of exploration component but also it generates the chaotic dynamics reflecting the inputs and output feedback, and we can speculate that such dynamics is useful for switching from exploration mode to exploitation mode.

We tested whether the agent can resume exploration when the rule is changed after the agent has learned under the previous rule. Here, we changed the rule every 500 episode and the result is shown in Fig. 8. The steps are rapidly increased after each rule change point because the agent could not reach the goal with the behavior that had been learned in previous episodes. However, after that, the number of steps decreased gradually. That shows that the agent successfully learned the new rules after the rule changes. Fig. 9 (a) and Fig. 10 (a) shows the network outputs and frequency of agent visits, respectively, at the episode 501, which is the first rule change point. When the agent received unknown feedback from the environment, the outputs became irregular based on the chaotic dynamics, and then the agent returns to exploration mode. Fig. 9 (b) and Fig. 10 (b) show the results at the final episode. The agent successfully learned after each rule change. This result suggests that the established connection weights, which were constructed to extract the input influence in the reservoir, were modified to learn the new rule. On the process of such reconstruction, the network dynamics became chaotic and the agent can resume exploration. These results suggest that the agent can adapt to changes in the environment and automatically switch between exploration and exploitation modes.



(a) exploration with chaos dynamics



(b) random numbers

Fig. 7. The weight values to the output layer. The horizontal axis indicates the connection weight number and the vertical axis indicates the weight values. Wight No.1 – 121 are connected from input vector $\boldsymbol{u}_t$ and the others are connected from the reservoir or the random number vector. The black dashed line is drawn at No. 121. Each plot color indicates the output neuron which the weights connect to. (a) The exploration is driven by internal dynamics in the reservoir. (b) The reservoir is replaced with a random number vector. Note that the scale of vertical axis in each figure is different.

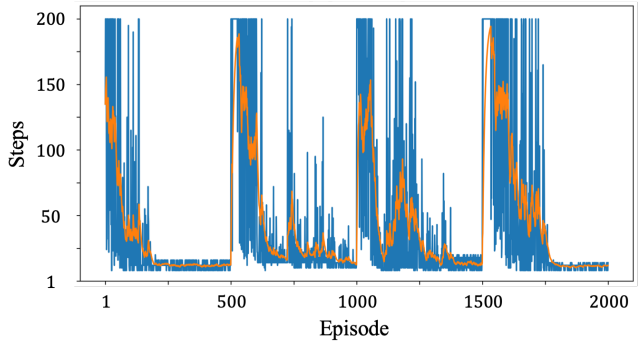
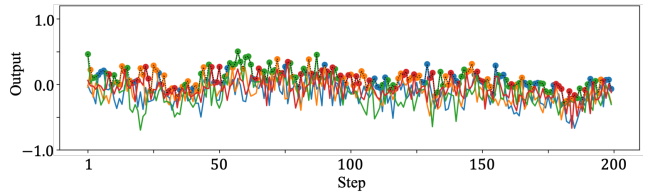


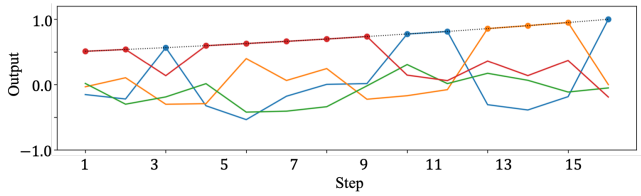
Fig. 8. Learning curve. This indicates the number of steps in each training episode. Orange line is a moving average of the numbers. The reduction of the step shows the successfully learning of the agent. The task rule is changed at episode 501, 1001 and 1501.

IV. Conclusion

In previous studies, we have adopted Actor-Critic method in chaos-based RL. In this paper, we demonstrated that a chaotic NN learned a goal task in a grid world environment without external random exploration noise and showed that Q-learning algorithm can be adopted in chaos-based RL. It is confirmed that irregular outputs originated from chaotic reservoir became ordered as learning progressed, and the dynamics automatically transited from exploration mode to exploitation mode. Additionally, by comparing with the result



(a) episode 501



(b) episode 2000

Fig. 9. Network output change in one episode for the cases of immediately after the first rule change (episode 501) and after learning fourth rule (episode 2000). Each color indicates an action as Q^1:blue, Q^2:orange, Q^3:green and Q^4:red. Plotted points and dotted line indicate the highest output at each step.

of the network whose reservoir is replaced with random number vector, we showed the dynamics in the reservoir reflects the inputs and output feedback, rather than only a random-like dynamics. Furthermore, it is shown that, when the task rule is changed after learning, the outputs became irregular based on the chaotic internal dynamics, and then the agent return to exploration mode and succeeded to learn the new rules.

The grid world goal task employed in this study does not require memory function for the learner agent. We predict that this frame also can be adopted to tasks that require memory function because the ESN, which is used as a network having chaotic dynamics in this study, actually has the ability to process series data. Moreover, if the network is layered more deeply, it is expected the agent can learn more complex tasks [26] [27] [28].

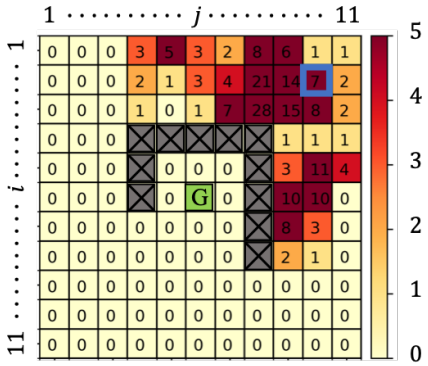
We consider that the phenomenon of the exploration driven by chaotic dynamics and the stabilization by learning agrees with the knowledge found by Freeman in his research about olfactory system of a rabbit. In addition, we predict that neural network dynamics around the edge of chaos is appropriate to realize transiting between chaotic and ordered states. We will examine how the network chaoticity changes as learning progresses and will investigate the relation between network chaoticity and the balance of exploration and exploitation.

References

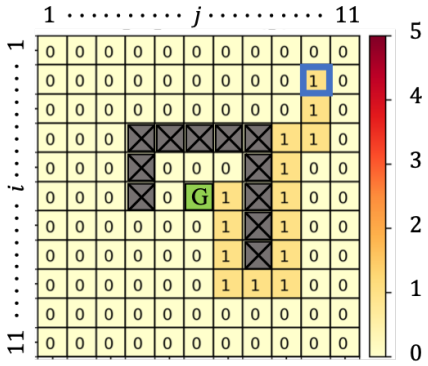
[1] K. Alex, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks,” Advances in neural information processing systems, vol. 25, pp. 1097–1105, 2012.

[2] K. He, X. Zhang, S. Ren, J.Sun, “Deep residual learning for image recognition,” Proc. of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[3] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You only look once: unified, real-time object detection,” Proc. of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016.



(a) episode 501



(b) episode 2000

Fig. 10. Frequency of agent visits at a rule change point (episode 501) and after learning (episode 2000). In both cases, start position is $(i, j) = (2, 10)$ (indicated by blue box).

[4] A. Graves, N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," International conference on machine learning, pp. 1764–1772, 2014.

[5] D. Amodei et al. "Deep speech 2: End-to-end speech recognition in english and mandarin," International conference on machine learning, pp. 173–182, 2016.

[6] O. Vinyals, A. Toshev, S. Bengio, D. Erhan, "Show and tell: a neural image caption generator," Proc. of the IEEE conference on computer vision and pattern recognition, pp. 3156–3164, 2015.

[7] R. S. Sutton, A. G. Barto, "Introduction to reinforcement learning," vol. 135, Cambridge: MIT press, 1998.

[8] G. Tesauro, "Temporal difference learning and TD-Gammon," Communications of the ACM 38.3, pp. 58–68, 1995.

[9] K. Shibata, Y. Okabe, "Reinforcement learning when visual sensory signals are directly given as inputs," Proc. of Int'l Conf. on Neural Networks, pp. 1716–1720, 1997.

[10] K. Shibata, "Functions that Emerge through end-to-end reinforcement learning -the direction for artificial general intelligence-," Proc. of The 3rd Multidisciplinary Conf. on Reinforcement Learning and Decision Making (RLDM) vol. 17, May 2017

[11] V. Mnih et al., "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[12] 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," Proc. of Int'l Conf. on Neural Information Processing, vol. 5507, pp. 970–978, 2009.

[13] K. Shibata and K. Goto, "Emergence of flexible prediction-based discrete decision making and continuous motion generation through Actor-Q-learning," Proc. of Int'l Conf. on Development and Learning and on Epigenetic Robotics, 2013.

[14] M. Hausknecht, P. Stone, "Deep Recurrent Q-Learning for partially observable MDPs," 2015 AAAI Fall Symposium Series, 2015.

[15] C. A. Skarda, W. J. Freeman, "How brains make chaos in order to make sense of the world," Behavioral and brain sciences 10.2 pp. 161–173, 1987.

[16] Y. Osana, M. Hagiwara, "Successive learning in chaotic neural network," 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227). vol. 2. IEEE, 1998.

[17] W. J. Freeman, "The physiology of perception," Scientific American 264.2 pp. 78–87, 1991.

[18] K. Shibata and Y. Sakashita, "Reinforcement learning with internal dynamics based exploration using a chaotic neural network," 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, 2015.

[19] K. Shibata and Y. Goto, "New reinforcement learning using a chaotic neural network for emergence of "Thinking"-"Exploration" grows into "Thinking" through learning," Proc. of the 3rd Multidisciplinary Conf. on Reinforcement Learning and Decision Making, vol. 17, May 2017.

[20] Y. Goto and K. Shibata, "Influence of the chaotic property on reinforcement learning using a chaotic neural network," Proc. of Int'l Conf. on Neural Information Processing. vol. 10634, pp. 759–767, November 2017

[21] K. Sato and K. Shibata, "Chaos-based reinforcement learning when introducing refractoriness in each neuron," Proc. of Int'l Conf. on Robot Intelligence Technology and Applications. Communications in Computer and Information Science, vol. 1015. Springer, pp. 76–84, December 2018.

[22] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," Bonn, Germany: German National Research Center for Information Technology GMD Technical Report 148.34, 13, 2001.

[23] W. Maass, T. Natschläger, H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," Neural computation 14.11 pp. 2531–2560, 2002.

[24] G. M. Hoerzer, R. Legenstein, W. Maass, "Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning," Cerebral cortex 24.3, pp. 677–690, 2014.

[25] T. Matsuki and K. Shibtata. "Reward-based learning of a memory-required task based on the internal dynamics of a chaotic neural network," Proc. of Int'l Conf. on Neural Information Processing. Lecture Notes in Computer Science, Springer, vol. 9947, pp. 376–383, October 2016.

[26] T. Matsuki and K. Shibtata. "Reinforcement learning of a memory task using an echo state network with multi-layer readout," Proc. of Int'l Conf. on Robot Intelligence Technology and Applications 5. RiTA 2017. Advances in Intelligent Systems and Computing, vol. 751, Springer, pp. 17–26, December 2017. (published in 2018)

[27] K. Bush, C. Anderson, "Modeling reward functions for incomplete state representations via echo state networks," 2005 IEEE International Joint Conference on Neural Networks, vol. 5, 2005.

[28] Š. Babinec, J. Pospichal, "Merging echo state and feedforward neural networks for time series forecasting," International Conference on Artificial Neural Networks. Springer, Berlin, Heidelberg, pp. 367–375, 2006.