Reinforcement Learning with Internal-Dynamics-based Exploration Using a Chaotic Neural Network

Katsunari Shibata and Yuta Sakashita
Dept. of Electrical and Electronic Engineering, Oita University
700 Dannoharu, Oita, JAPAN 870–1192
Email: shibata@oita-u.ac.jp

***Abstract*—In this paper, a novel concept is proposed where exploration, which is essential in reinforcement learning, is considered to be one aspect of the motions generated by the learner's internal dynamics and is expected to develop through learning towards more purposeful higher dynamic functions such as "thinking". To realize such a concept, a chaotic neural network is introduced for generating motions with exploratory factors that are derived from the internal chaotic dynamics without adding external random noises. Effective exploration is expected based on the dynamics called "chaotic itinerancy", which is also expected to be the key to learning higher dynamic functions more easily that require both stable and transitive dynamics. This paper also proposes a reinforcement learning method without any external random noise, using the temporal difference (TD) error of the state value and the contribution trace of each input to the output increase in each neuron. It was confirmed in a simple learning task that by using a chaotic neural network, an agent could explore in accordance with the internal chaotic dynamics and could learn goal-directed behaviors. The proposed framework seems promising to explain the emergence of higher intelligence in real lives and also to develop human-like intelligence though there are many remaining problems to be solved.**

I. Introduction

From the viewpoint of "emergence of intelligence", reinforcement learning seems promising due to its autonomous learning ability based on the learner's own trial and error, and its function emergence property when used with a recurrent neural network [1]. In order to realize such trial and error, the use of stochastic exploration that needs a random number generator has been taken for granted. In cases with discrete action generation, one action is chosen stochastically using the learner's outputs, such as ϵ-greedy or Boltzmann selection using Q-values [2]. While, in cases with continuous motion generation, a motion is picked according to the probabilistic distribution determined by the learner's outputs such as the actor outputs in Actor-Critic [2][1] as shown in Fig. 1.

Our group has been questioning the conventional exploration from the following points. 1) In humans, exploration seems far more intelligent. When we meet a fork in a road, we do not stochastically move each of our actuators independently as shown in Fig. 2, but do so intelligently, choosing to go down one of the two roads behind the fork. That needs sophisticated integration of many actuators' motions. 2) Does each real life have an independent random number generator used only for stochastic exploration? 3) What should the basis be for adjusting the ratio of exploration and exploitation autonomously? 4) Exploration is influenced by the control interval (cycle).

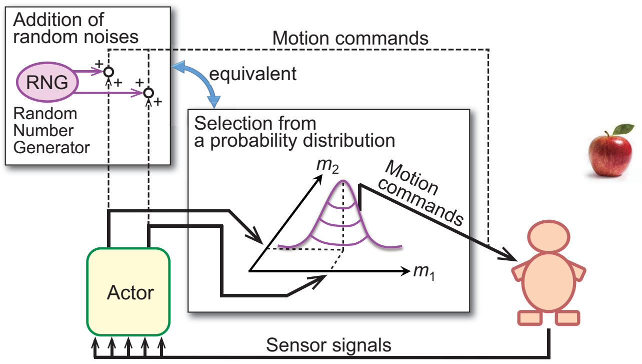


Fig. 1. The stochastic selection of continuous motion in reinforcement learning. It is equivalent to the addition of external random numbers to the actor outputs.

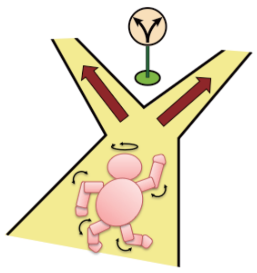


Fig. 2. Fork Problem: Adding a random number to each motion command is not an effective method to explore which way a robot should go at a fork.

Although the same level of random noises is added, if the control interval becomes 10 times longer, the exploration is completely different. From the above, the development of smart exploration in autonomous learning systems has been an enduring desire. We showed that a system using reinforcement learning with a recurrent neural network acquires intelligent exploration behaviors [3][4], but a random number generator for stochastic exploration was still required.

On the other hand, it is well known that chaotic dynamics have the ability to realize effective exploration. Many works have been conducted for a long time, but the ability has mainly been investigated in the field of associative memory. Several works have tried to use a chaotic sequence in reinforcement learning on behalf of a random number generator [5][6] as shown in Fig. 3(a). However, the chaotic sequence generator is still located outside of the action generator.

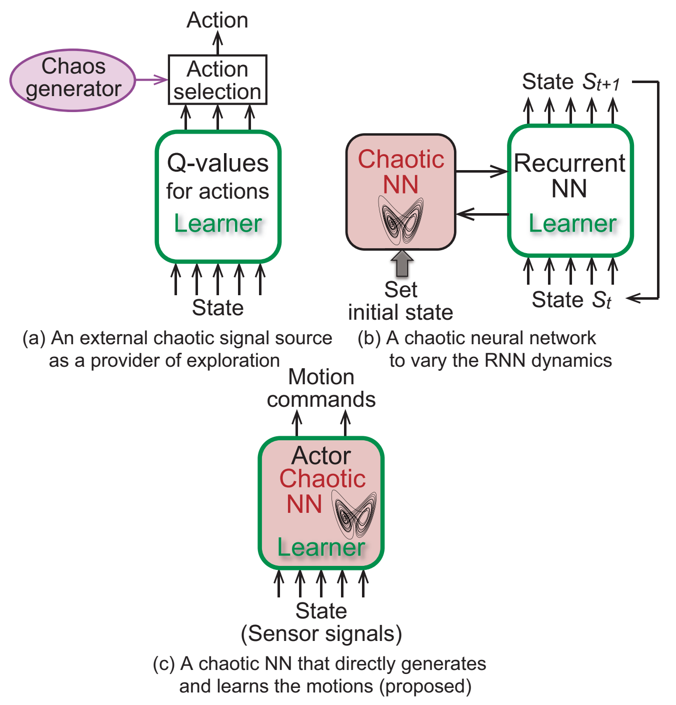


Fig. 3. Two conventional uses of chaos in learning and the proposed method.

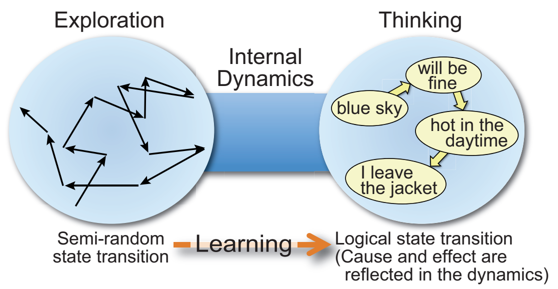

Exploration
Thinking
Internal Dynamics
will be fine
blue sky
hot in the daytime
I leave the jacket
Semi-random state transition
Learning
Logical state transition (Cause and effect are reflected in the dynamics)

Fig. 4. "Thinking" is developed from "exploration" on the line of "internal dynamics" through learning.

Arie et al. proposed a learning system with a chaotic neural network (CNN) connected with a regular recurrent neural network (RNN) as shown in Fig. 3(b). It explores a novel goal-directed sequence as a combination of the given primitive sequences by varying the initial states of the CNN, with the expectation that it is sensitive to its initial conditions [7]. It is a pioneering work from the viewpoint of an internal-dynamics-based exploration produced by the combination of both networks. However, the RNN learns primitive sequences given by a human tutor in advance, and the CNN is not expected to generate motions, but instead only varies the sequence. Furthermore, the system explores off-line and chooses the best one. Those suggest the underlying concept that exploration and motion generation are completely different.

In this paper, the authors present a novel idea that exploration is one aspect of the internal dynamics formed within the motion-generating (actor) network itself, and the dynamics develop through learning resulting in higher dynamic functions such as 'thinking' as shown in Fig. 4. In other words, 'exploration' and 'thinking' are both treated along the same line of "internal dynamics". Learning takes in useful dynamics to obtain rewards, such as the cause and effect of the world, and grows the dynamics from 'exploration' to 'thinking', but when the learner meets an uncertain or unfamiliar situation, the dynamics return to 'exploration'. The dynamics of 'thinking' need both stability in a state and transition between states. These two demands seem to be contradictory at first glance, and it is difficult to learn their coexistence from scratch [8].

Chaotic dynamics seem to meet the above requirements. Especially, "chaotic itinerancy" [9] can be considered to be the origin of effective exploration at the fork as well as thinking dynamics. Chaotic dynamics build a bridge from "exploration" to "thinking". Chaotic transitory phenomena are observed experimentally in real biological lives, and the difference between the dynamics for known and unknown stimulus was also observed [10]. Similar results can be seen in a simulation using a chaotic neural network, which can also learn unknown input patterns to be known patterns [11]. After the learning of a state sequence that changes stochastically using a recurrent network, chaotic dynamics were observed in it [12]. They are interesting for demonstrating the possibility that the purposeful control of chaotic dynamics can be acquired through learning.

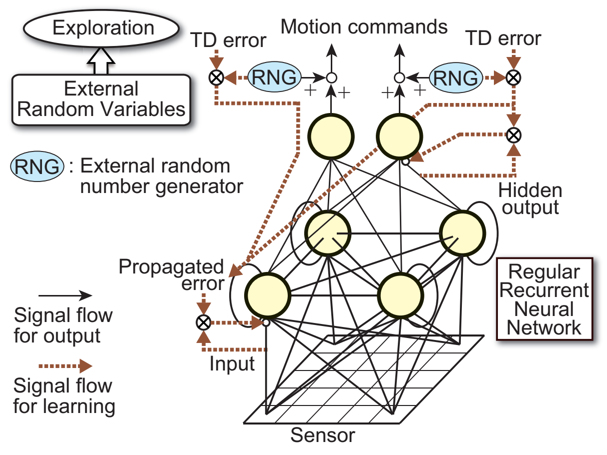
According to the above discussion, the authors propose to use a chaotic neural network (CNN) for motion generation, and to train that CNN through reinforcement learning as shown in Fig. 3(c). Thanks to its internal chaotic dynamics, the learner can generate exploratory behaviors without any external random noise, and the dynamics are expected to change purposefully through learning.

However, there is one serious problem. Conventional reinforcement learning uses the correlation between the given perturbations or noises and the resulting change in the state or action value. It can be realized because a perturbation that is added from the outside of the action generator can be explicitly isolated from the original action or motion commands. However, in the proposed system, exploration is generated as a function of the internal dynamics, and the perturbations and motion commands are originally inseparable. In this paper, by focusing on the temporal difference (TD) error of the state value and the contribution trace of each input to the output increase in each neuron, a novel and simple reinforcement learning algorithm based on the explorations generated by the internal dynamics of a CNN is proposed.

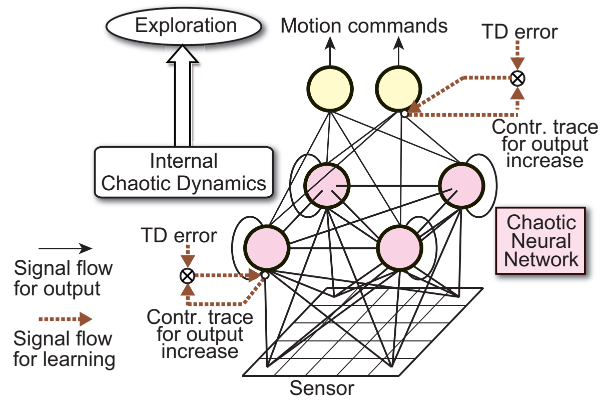
Here, in the initial step for this big paradigm shift, it is confirmed in a simple task whether reinforcement learning using a CNN works without any external random noise.

II. Reinforcement Learning Using a Chaotic Neural Network

Here, actor-critic [2] is employed as a reinforcement learning architecture considering that the actor outputs can be sent directly to the motors as continuous motion commands. In conventional reinforcement learning using a neural network, one motion is decided on stochastically by adding an external random noise to each actor output as shown in Fig. 5(a). Each connection weight is updated by back propagation [13]. The error at each output neuron is the product of TD error and the random noise. In the proposed learning system, a



(a) Conventional exploration and learning



(b) Proposed exploration and learning

Fig. 5. Comparison of conventional reinforcement learning with external perturbations and proposed reinforcement learning with internal-dynamics-based explorations using a chaotic neural network.

chaotic neural network (CNN) generates the motions of the learner working as the actor as shown in Fig. 5(b). The learner explores according to the internal dynamics of the CNN without any random noise from the outside as mentioned.

A primary concern will be what type of CNN is used and what parameters are chosen for good exploration and/or for good learning of dynamics. However, here, this research focuses on confirming whether reinforcement learning works or not. Therefore, the type of CNN and the parameters used are not investigated systematically. It has two layers except for the input layer, and 50 neurons are connected with each other in the hidden layer. In a CNN, a refractory period is often introduced in each neuron [14], but here, for simplicity, regular static neurons are used and produce chaotic dynamics. The gain of the sigmoid function that is used as the activation function in each hidden or output neuron is 7.0, and the value ranges from -0.5 to 0.5. Each of the hidden neurons computes the internal state from the inputs and the previous outputs of the hidden neurons as

$$u_{j,t}^{(2)} = \sum_i w_{j,i}^{(2)} o_{i,t}^{(1)} + \sum_i w_{\mathrm{fb}j,i}^{(2)} o_{i,t-1}^{(2)} \tag{1}$$

where superscript (1) and (2) indicate the input and hidden layers respectively. $u_{j,t}^{(2)}$ is the internal state of the j-th neuron in the hidden layer of the CNN at time t, and $w_{j,i}^{(2)}$ is the connection weight of the j-th hidden neuron for the connection from the i-th input, $o_{i,t}^{(1)}$ is the i-th input of the CNN at time t. $w_{\mathrm{fb},j,i}^{(2)}$ is the weight for the feedback (mutual) connection from the i-th hidden neuron, and $o_{i,t-1}^{(2)}$ is the output of the i-th hidden neuron at time $t-1$. The output neuron computes the internal state as

$$u_{j,t}^{(3)} = \sum_i w_{j,i}^{(3)} o_{i,t}^{(2)} \tag{2}$$

where superscript (3) indicates the output layer. The output of each neuron is computed from the internal state as

$$o_{j,t}^{(l)} = \frac{1}{1+exp(-g \cdot u_{j,t}^{(l)})} - 0.5 \tag{3}$$

where l is the layer number and g is the gain of the sigmoid function, set to 7.0. The network is a simple recurrent neural network, but the large feedback connection weights (± 1.5) and the large gain in sigmoid function enable the generation of chaotic dynamics.

We consider that the critic output should be generated in the same network at last, but here, at first, there is another neural network for critic. It is a regular layered neural network with the same inputs as the CNN and generates only critic output. It has no connections between the hidden neurons, and the sigmoid function, whose gain is 1.0 and value ranges from -0.5 to 0.5, is used as the activation function of each of the neurons in the network including the output neuron.

The critic network is trained in the same way as the conventional one. The training signal tr_{critic} is generated based on temporal difference (TD) learning as

$$tr_{critic,t} = \gamma o_{critic,t+1} + r_{t+1} = \hat{r}_t + o_{critic,t} \tag{4}$$

where γ is a discount factor, $o_{critic,t+1}$ is the output of the critic network at time $t+1$, r_{t+1} is the given reward at time $t+1$, and $\hat{r}_t$ is the TD error as

$$\hat{r}_t = \gamma o_{critic,t+1} + r_{t+1} - o_{critic,t}. \tag{5}$$

Here, since an episodic task is assumed, when the learner reaches the goal, the training signal is set as

$$tr_{critic,t} = r_t, \tag{6}$$

and the episode is terminated. All the connection weights in the critic network are updated based on the ordinary back propagation [13] using the above training signal.

The learning method of the actor CNN is completely different from the conventional one, and is proposed here for the internal-dynamics-based exploration. All the connection weights in the actor CNN, except for the mutual feedback connections between the hidden neurons, are updated using the TD error $\hat{r}$ as

$$\Delta w_{j,i,t}^{(l)} = \eta \hat{r}_t c_{j,i,t}^{(l)} \tag{7}$$

where $\Delta w_{j,i,t}^{(l)}$ is the weight update for the connection from the i-th neuron in the $(l-1)$-th layer to the j-th neuron in the l-th layer. $c_{j,i,t}^{(l)}$ is the trace of correlation between the input from the i-th neuron in the $(l-1)$-th layer and the output change in the j-th neuron in the l-th layer. Written in the continuous time domain, the trace $c^{(l)}_{j,i,t}$ is updated simply according to the infinitesimal change in the output $do^{(l)}_j$ as

$$dc^{(l)}_{j,i,t} = -c^{(l)}_{j,i,t}|do^{(l)}_{j,t}| + o^{(l-1)}_{i,t} do^{(l)}_{j,t}, \tag{8}$$

and written in the discrete time domain, it is updated at each time step as

$$c^{(l)}_{j,i,t} = (1 - |\Delta o^{(l)}_{j,t}|)c^{(l)}_{j,i,t-1} + \Delta o^{(l)}_{j,t} o^{(l-1)}_{i,t} \tag{9}$$

where $\Delta o^{(l)}_{j,t} = o^{(l)}_{j,t} - o^{(l)}_{j,t-1}$. The trace is designed to represent and hold the contribution of the corresponding input to the increase in the neuron output. When the TD error is positive, the weights are updated on the basis of promoting the output change that has contributed to reach the present output value in each neuron, while, when the TD error is negative, they are updated on the basis of diminishing the output change.

One major difference from conventional learning is that there are no error signals propagated backward. For each hidden neuron, the input signals are the same as any other, but the traces $c_{j,i}$ are different among the hidden neurons, and that enables learning without error propagation. However, further investigation is necessary for this point hereafter.

The authors believe that acquisition of useful dynamics through learning is very important, and one of the advantages to introducing a CNN into reinforcement learning is to reflect the learned dynamics into the chaotic dynamics. However, for simplicity, the weights of the mutual connections between the hidden neurons in the CNN are not learned here.

III. Simulation

In this section, it is examined that the proposed learning works in a simple task. As shown in Fig. 6, there is a 22×22 field, and an agent is placed at random with its center located within the 20 × 20 area in the field at the beginning of each episode. The agent moves, and when its center reaches the circle within a diameter of 2.0 at the center of the field, 0.4 is given as a reward. Otherwise, no reward or penalty is given. The episode is terminated when it either reaches the goal, or fails to do so in 30,000 steps. There is a visual sensor that has 11×11 visual cells, each of which has a 2×2 receptive field, and covers the whole the field without any overlap between cells. The size of the agent is also 2×2, and each visual cell outputs the area occupied by the agent in the cell. If the agent image fits just one of the cells, the corresponding signal is 4.0, and all the other signals are 0.0. The 121 signals from the cells are sent to both networks as inputs.

Two outputs are generated in the actor CNN, and the agent moves accordingly. Each of the two outputs is responsible for the x or y direction. After every 5,000 episodes, the environment changes and the responsible direction for each output is swapped. Each output works as acceleration as

$$v_{x,t} = 0.9v_{x,t-1} + o^{(3)}_{1or2,t} \tag{10}$$

$$v_{y,t} = 0.9v_{y,t-1} + o^{(3)}_{2or1,t} \tag{11}$$

where $v_{x,t}$ and $v_{y,t}$ are the speed in the direction of x and y respectively, but their range is limited; -1.0 to 1.0. However, to make the maximum speed to be a constant value that is not

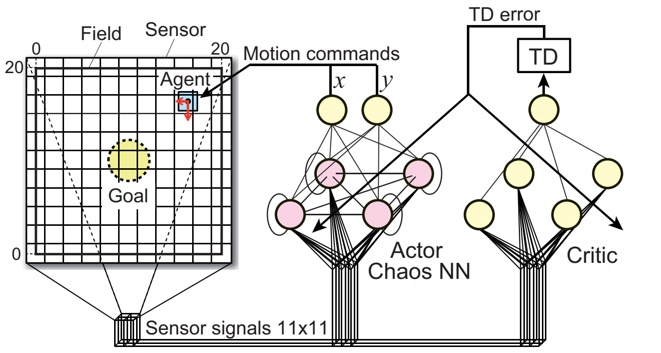


Fig. 6. Task, learner's architecture and signal flow.

TABLE I. Parameters used in the simulation.

		Actor CNN	Critic NN
Number of Layers		3	3
Number of Inputs		121	121
Number of Hidden Neurons		50	30
Number of Outputs		2	1
Gain of Sigmoid Function		7.0	1.0
Value Range of Sigmoid Function		-0.5 - 0.5	-0.5 - 0.5
Learning Rate	Output <- Hidden	4.0	4.0
	Hidden <- Hidden	0.0	-
	Hidden <- Input	0.4	4.0
Initial Weights (range of random numbers)	Output <- Hidden	±0.1	0.0
	Hidden <- Hidden	±1.5	-
	Hidden <- Input	±0.1	±0.1
Discount Factor γ		0.98	
Reward at Goal r		0.4	

dependent on the direction of the agent's motion, the velocity is normalized by the maximum speed for the direction as

$$loc_{x,t} = loc_{x,t-1} + v_{x,t}/max\_v_t \tag{12}$$

$$loc_{y,t} = loc_{y,t-1} + v_{y,t}/max\_v_t \tag{13}$$

where max_v_t is the maximum possible speed for the direction at time t. For example, when one of v_x and v_y is 0.0, $max\_v = \sqrt{1.0^2 + 0.0^2} = 1.0$, and when v_x is equal to v_y, max_v is $\sqrt{1.0^2 + 1.0^2} = \sqrt{2}$. When it collides with the wall at the boundary of the field, it stops at the location, and v_x and v_y are reset to 0.0. No penalty is imposed. The agent motion is dynamic, but there are no inputs for the agent to know the velocity, thus causing perceptual aliasing. Table I shows the parameters used. When the number of steps in each episode becomes so large that $0.125^{\frac{1}{step}}$ is larger than the discount factor γ, the factor is set to $0.125^{\frac{1}{step}}$ temporally so as that the critic value does not become too small through learning. As previously mentioned, these parameters were not finely tuned.

Learning was conducted for 20,000 episodes in each simulation run, and 20 simulation runs were performed with varying initial connection weights. The following result is the best out of the 20 runs. There were no episodes in which more than 2,000 steps were taken for the agent to reach the goal even though the environment changed after every 5,000th episode.

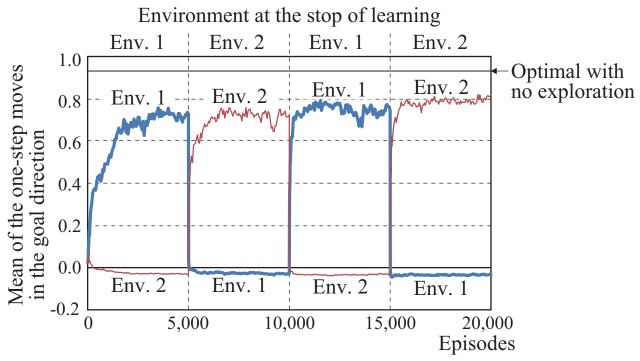


Fig. 7. As a learning curve, the mean of the one-step moves in the goal direction over 200 test episodes is plotted. The used actor CNN was saved at the episode shown in the lateral axis. The environment where the agent moved changed after every 5,000th episode, and the test was performed for both environments: Env. 1 and Env. 2.

Figure 7 shows the learning curve. The actor CNN was saved after the number of episodes of learning indicated by the lateral axis, and the mean of the one-step moves in the goal direction over 200 test episodes using the saved network is plotted. At each test episode, the agent was located on one of the 200 points that uniformly divide the circle, whose radius is 9 from the center. If the agent did not reach the goal within 100 steps during the test episode, the episode was terminated. Each of the two lines in the figure shows a case in the environment 1 or 2. The moves did not reach the optimal level with no exploration, but it can be seen that the agent learned to reach the goal in the learning environment. After the environment changed, the agent learned its motion even though no random exploration was done. In the environment where the agent was not learning at the time, the move was negative, or in other words, the agent was at a position further from the goal after 100 steps.

Each figure in Fig. 8 shows the trajectories in eight test episodes from different starting locations using the actor CNN after some episodes of learning. When the agent could not reach the goal in 100 steps, the trajectory for the first 100 steps is shown. It can be seen that before learning: (a), the agent moved around according to the chaotic dynamics, and in three of the eight episodes, it reached the goal. Even in the other five episodes, it reached the goal in 2,000 steps. After 200 episodes of learning:(b), a tendency to go to the goal can be seen, and in all the eight episodes, the agent reached the goal in 100 steps. After 5,000 episodes of learning: (c), the agent reached the goal in smaller steps though it sometimes took a roundabout route. Using the same CNN, the agent is put in the environment 2 where the responsible direction of the actor outputs is swapped from the environment 1: (d), the agent went around the bottom right or top left corner of the field. It moved around to some extent, but could not reach the goal even after 30,000 steps. However, when the agent learned during test episodes, it reached the goal within 3,000 steps from any of the eight starting locations. After 5040 episodes: (e), the agent had learned 40 episodes in environment 2, and it reached the goal from any of the eight starting locations within 100 steps. Finally, after 10,000 episodes: (f), the agent could reach the goal in 10 to 12 steps though the optimal step is nine.

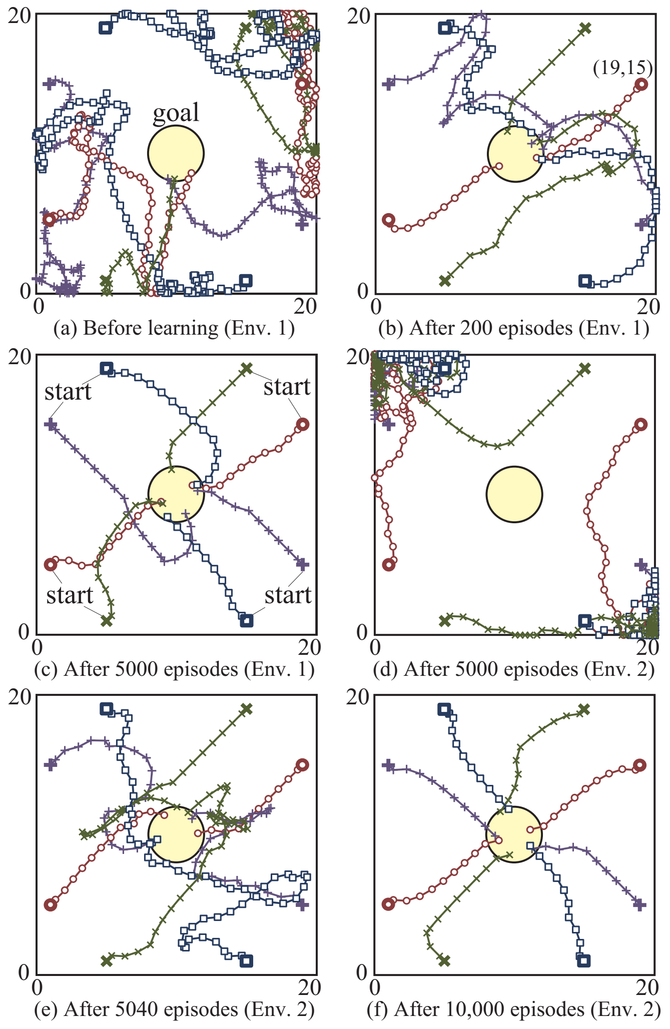


Fig. 8. Changes in the agent trajectories from the 8 test locations using the saved CNN. The environment in which the agent learned changed every 5,000 episodes. In the cases of (c) and (d), the used CNN was saved just before the environment changed from 1 to 2 during learning. In the case of (f), the used CNN was saved just before the environment changed from 2 to 1. The larger and thicker marks indicate the agent's starting locations.

In Fig. 9, the output changes in the two output neurons and six hidden neurons in one episode from a start location (19.0, 15.0) for three of the six cases in Fig. 8 are shown. Note that Fig. 9 (a) and (c) show the output change for the first 10 steps, while Fig. 9 (d) shows those from the 91st through 100th steps. Before learning: (a), the two actor outputs changed their value often within the non-saturation region where the value is neither very large nor very small. The outputs of the hidden neurons went back and forth between around 0.5 and around -0.5. That is because the initial weights of the mutual connections between the hidden neurons are large. After 5,000 episodes of learning, (c), as shown in the top graph, the critic output got to represent the state value even though it is slightly smaller than the ideal curve computed from the discount factor. It can also be seen that the actor outputs took the value of 0.5 or -0.5, and rarely changed. The hidden neurons changed their values, but not so often as in the case in (a). However, in the environment 2, where the responsible directions of the two actor outputs were swapped: (d), using the same CNN as (c),

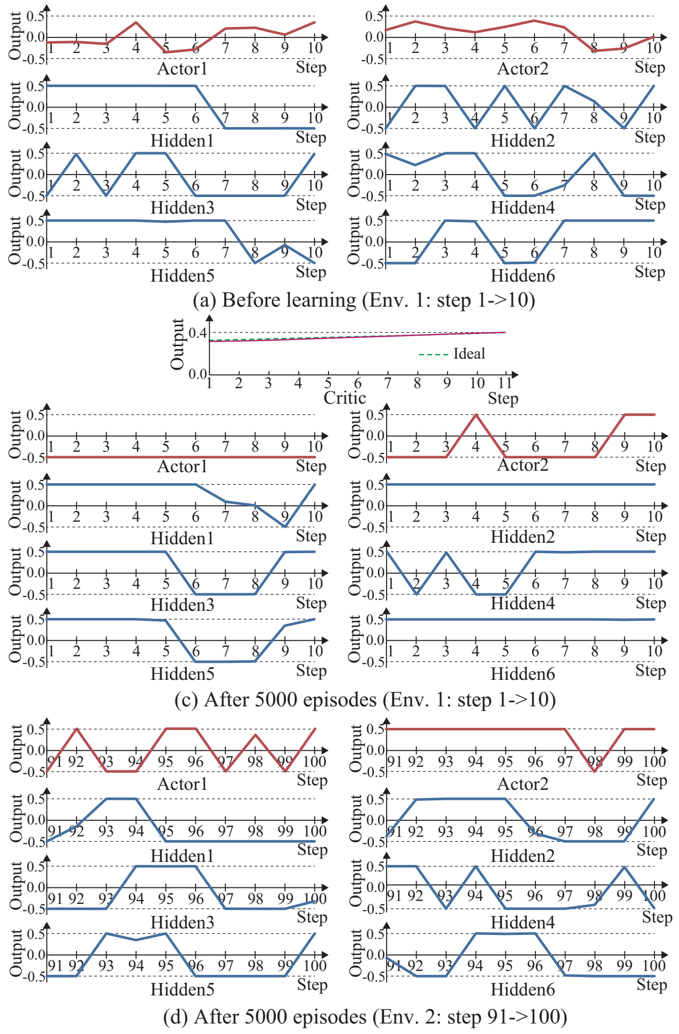


Fig. 9. Changes in the two actor outputs and first six hidden outputs for 10 steps in one episode in three of the six cases in Fig. 8. In the case of (c), the critic is also plotted to show the critic learned. In either case, the agent starts at (19.0, 15.0). In the cases of (a) and (c), the outputs in the first 10 steps are plotted, but in the case of (d), the outputs from the 91st to 100th are plotted to see the agent's state after some time from start without reaching the goal.

the agent went to the bottom right corner of the field as seen in Fig. 8 (d). It can be seen that the hidden outputs changed more often from the 91st to 100th step than in the case of (c). That must help to generate more exploratory behaviors.

As an index of exploratory behaviors, the Lyapunov exponent, which shows the sensitivity to small perturbations, was observed in the actor CNN. Here, 200 test episodes are performed as before. At each step in the 200 test episodes, a random vector whose size is 10^{-3} is added to the internal states of the 50 hidden neurons in the actor CNN, and after one step, the distance of the hidden states from the case when no perturbation is added was observed for the both environments. The expanding ratio of the distance is averaged as

$$\lambda = \frac{1}{\sum_{e=1}^{Episode} Step(e)} \sum_{e=1}^{Episode} \sum_{t=1}^{Step(e)} log_2 \frac{d(e,t)}{10^{-3}} \tag{14}$$

where $d(e,t)$ is the Euclidean distance of the states in the hidden neurons at time t when the distance 10^{-3} is given at $t-1$ in the e-th episode, $Episode$ is the number of episodes in the test, which is 200 here, and $Step(e)$ is the number of steps to reach the goal in the e-th episode. In this paper, this value is called Lyapunov exponent. Note that the dynamics are not only produced by the feedback connections in the actor CNN, but also by the loop of motion and perception of the agent.

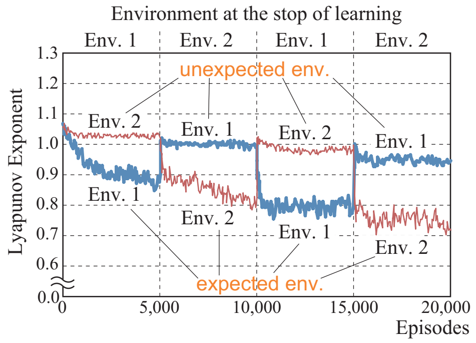


Fig. 10. Change in the Lyapunov exponent during learning for both environments, one is where the agent learned and the other is the agent did not learn at that episode.

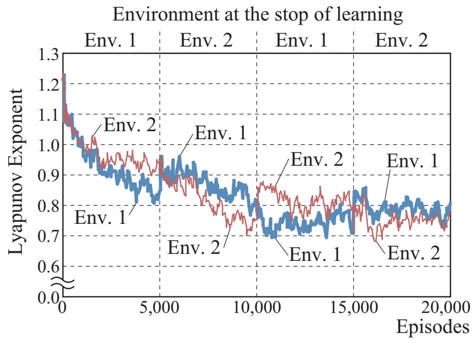


Fig. 11. Change in the Lyapunov exponent for only the first 4 steps after each start.

Figure 10 shows the change in the Lyapunov exponent during learning in each environment; the environment where the agent was either learning or not learning during the episode when the CNN was saved. It is interesting that in the environment where the agent was learning, the exponent became small according to the progress of learning, while it did not decrease so much when the test was done in the environment where the agent was not learning. Those match the output fluctuations in hidden neurons as shown in Fig. 9. It is also interesting that, after the environment changed, the exponent for the previous environment became large soon. When the exponent is observed in only the first four steps after the start, the exponents change as shown in Fig. 11. It must take some steps for the agent to perceive that the environment is unexpected. The exponent for "not-learning" environment decreased more than that in the case of Fig. 10 though the exponent is different slightly between "learning" and "not-learning" environment even in 4 steps. The cause behind these results should be investigated further.

Next, it is shown how the connection weights in the actor

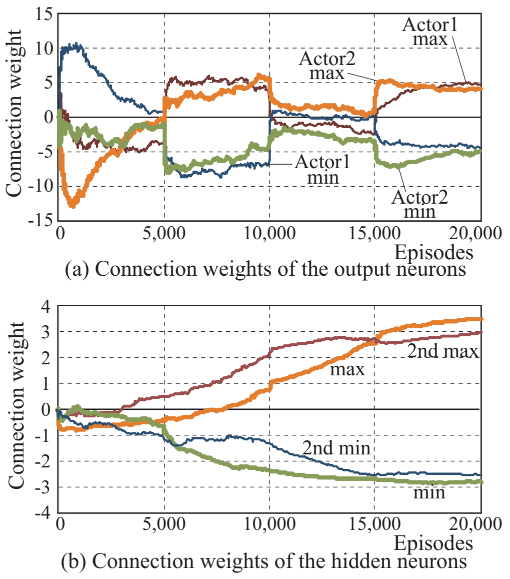


Fig. 12. Changes in several connection weights in the actor CNN.

CNN changed during learning. Fig. 12(a) shows the changes to the maximum or minimum (greater negative) weights after learning for each output neuron. Fig. 12(b) shows that of the 1st and 2nd maximum and minimum weights from inputs in the hidden layer after learning. Probably because the learning rate is greater in the output layer than in the hidden layer, the absolute value of the weights are larger in the output layer, and change a lot when the environment where the agent was learning changed after every 5,000 episodes. On the other hand, the weights in the hidden layer are apt to increase or decrease gradually. It can be thought that the changes in the connection weights from the inputs during learning resulted in more stable hidden outputs shown in Fig. 9(c) and the decrease in the Lyapunov exponent shown in Fig. 10.

IV. Comparison with the Conventional Method

The proposed method was compared with the the conventional reinforcement learning, where external random noises are added and a feedforward-type network with no feedback connections was used as the actor network. The gain of the sigmoid function in every neuron is 1.0. As shown in Table II, the other parameters were attempted to match those in the CNN case, but the learning rate and initial weights of the actor network are different. Note that the parameters were not optimized systematically, and the performance can be improved by setting more appropriate ones. As the external perturbation, a uniform random number is added to each of the actor outputs with a value range of ±0.8, ±1.0 or ±1.2, that remained unchanged during learning. For the case of the CNN, the value range of the initial weights for the mutual feedback connections between hidden neurons were varied from ±0.5, ±1.0, ±1.5 or ±2.0. A total of seven cases were investigated.

Fig. 13 shows the comparison of the learning performances. Fig. 13(a) shows the mean moves in the goal direction for one-step during learning. The mean value was computed after each 20 episodes at first, and then that value was averaged over 20,000 episodes. Note that the agent's motion was influenced by the random perturbations or chaotic dynamics. The graph shows the means and standard deviations from the 20 simulation runs with different random number sequences. In the case of CNN, they were used to decide the initial connection weights and the agent's start location for each episode. The maximum number of steps needed to reach the goal during 20,000 episodes is plotted in Fig. 13(b) as an index of the exploration ability for the environment change. One episode was terminated at the 30,000th step when the agent could not

TABLE II. Parameters used in the simulation for examining the conventional method.

		Actor CNN	Critic NN
Number of Layers		3	3
Number of Inputs		121	121
Number of Hidden Neurons		50	30
Number of Outputs		2	1
Gain of Sigmoid Function		1.0	1.0
Value Range of Sigmoid Function		-0.5 - 0.5	-0.5 - 0.5
Learning Rate	Output <- Hidden	40.0	4.0
	Hidden <- Input	40.0	4.0
Initial Weights (range of random numbers)	Output <- Hidden	0.0	0.0
	Hidden <- Input	± 0.1	± 0.1
Range of Random Numbers		±0.8 - ±1.2	–
Discount Factor γ		0.98	
Reward at Goal r		0.4	

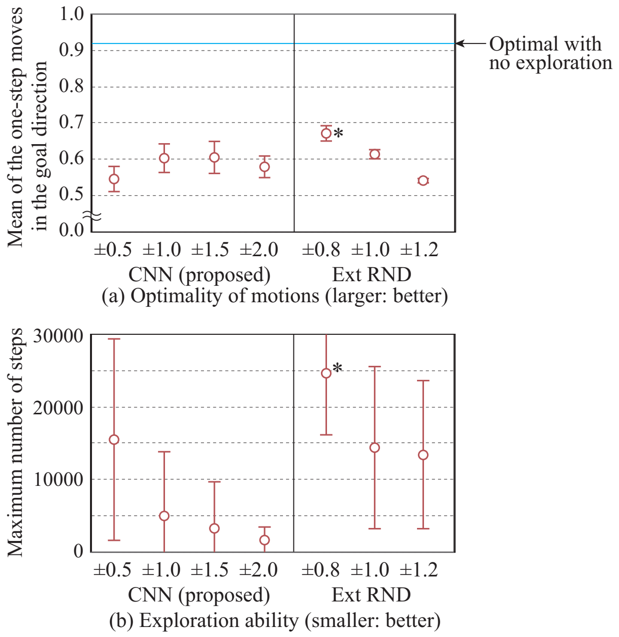


Fig. 13. Comparison of learning performance among seven cases from the two viewpoints: (a) mean of the one-step moves in the goal direction to show the optimality of motions and (b) maximum number of steps in 20,000 episodes to show the exploration ability. The means and standard deviations are plotted for each case over 20 simulation runs. [CNN: exploration by the internal dynamics of a CNN (proposed method), the number indicates the value range of the initial feedback connection weights.] [Ext RND: exploration by external random noises, and the number indicates the range of the random noises.] "*" in the graphs indicates that the agent was stuck at a corner of the field after the environment changed in two runs and the data are not included.

reach the goal. In the case of the conventional method with the level of random number ±0.8, the agent was stuck at a corner after the environment changed in two of the 20 simulation runs, and the data are not included in the plot in Fig. 13(b).

When the feedback connection weights in the proposed method or the random numbers in the conventional method were large, the exploration factor became large. Therefore, when they were too large, the optimality became worse in Fig. 13(a), while, when they were too small, the exploration ability became worse in Fig. 13(b). They have an appropriate size. The standard deviation in Fig. 13(a) is large in the proposed method. That is because learning is often unstable depending on the simulation run. The exploration ability seems better in the proposed method, but it should be investigated further.

V. Expectation Towards the Emergence of Thinking and Remaining Problems

The proposed idea is completely novel and is highly anticipated to learn more complicated and useful internal dynamics. Among the higher functions, "thinking" seems to be the most typical one and the most difficult to be achieved. "Reinforcement learning using a chaotic neural network" opens up a new way to achieve it through autonomous learning. However, there are remaining problems as follows.

a. Learning method

The proposed learning method does not require the back propagation of error signals. That is a very fascinating property for developing hardwares with parallel processing. However, since the task in this paper is so simple and actually mutual connections in the actor CNN were not learned, it is difficult to say that the method is sufficient to learn complicated dynamics. The "deep learning" that presently collects attentions for its ability for abstraction seems to suggest the importance of the error propagation, especially for the division of roles among neurons. It is important to examine whether the proposed method works in even more complicated tasks, especially in the tasks in which the learning of dynamics is required.

b. Intelligent adjustment between exploration and exploitation

It is expected that, by what has already been learned being reflected, the dynamics called "chaotic itinerancy" [9] emerges and intelligent exploration is realized. It should be examined in such tasks as "the fork problem" mentioned in the Introduction. It is also expected that when exposed to an unknown or unexpected situation, the dynamics transition to a more unstable state and behavior becomes more exploratory. From these points of view, the change in the Lyapunov exponent during learning observed in this paper, as well as the past works [11][12] suggest the possibility.

c. Critic learning

In this paper, the critic is computed in another neural network, but the authors believe that it should also be learned in the same network to achieve a higher abstract space in the CNN. However, it should be examined how the fluctuations due to the chaotic dynamics influence the learning.

d. CNN model and parameters

The behaviors of a CNN change drastically according to the model and parameters. What should be done is to know more about the influence of the type and parameters in a CNN to exploration and learning, and to choose appropriate ones.

VI. Conclusion

It was confirmed that, by using a chaotic neural network, an agent could explore with internal chaotic dynamics without adding external random numbers to the actor outputs, and could learn goal-directed behaviors using the proposed learning method. In the method, the contribution trace of each input signal to the output change is computed in each neuron and the corresponding weights are updated by the product of the trace and TD error. It is also examined that the sensitivities to small perturbations decreased for the learning environment according to the progress of learning. The authors think that it shows a possibility to control the dynamics rationally through learning. This is the introduction of a completely novel idea in reinforcement learning, and is expected to learn and acquire complicated dynamics that can be called "thinking" as an extension of the exploration in chaotic dynamics, but there are many remaining problems to be solved.

Acknowledgments

The authors wish to thank to Professor Kanakubo from the Shizuoka Inst. of Sci. & Tech. for his web page that helped them to remove their bias that chaos is very difficult to use. This work was supported by JSPS KAKENHI Grant Number 15K00360.

References

[1] K. Shibata: Emergence of Intelligence through Reinforcement Learning with a Neural Network, *Advances in Reinforcement Learning*, Abdelhamid Mellouk (Ed.), InTech, 99-120, 2011

[2] R. S. Sutton & A. G. Barto: *Reinforcement Learning: An Introduction*, A Bradford Book, The MIT Press,1998

[3] K. Goto & K. Shibata: Acquisition of Deterministic Exploration and Purposive Memory through Reinforcement Learning with a Recurrent Neural Network, *Proc. of SICE Annual Conf. 2010*, FB03-1.pdf, 2010

[4] K. Shibata: Learning of Deterministic Exploration and Temporal Abstraction in Reinforcement Learning, *Proc. of SICE-ICCAS (SICE-ICASE Int'l Joint Conf.)*, 4569-4574, 2006

[5] A. B. Potapov & M. K. Ali: Nonlinear Dynamics and Chaos in Information Processing Neural Networks, *Differential Equations and Dynamical System*, **9**(3&4):259-319, 2001

[6] K. Morihiro, T. Isokawa, N. Matsui & H. Nishimura: Effects of Chaotic Exploration on Reinforcement Learning in Target Capturing Task, *Int. J. Knowledge-based and Intelligent Engineering Sys.*, **12**:369-377, 2008

[7] H. Arie, T. Endo, T. Arakaki, S. Sugano & J. Tani: Creating Novel Goal-Directed Actions at Criticality: A Neuro-Robotic Experiment, *New Mathmatics and Computation*, **5**(1): 307-334, 2009

[8] Y. Taguchi & K. Shibata: Effect of Initial Weight Value when a Problem Requiring Internal State Transition is Learned by a Recurrent NN, *Annual Meeting of the SICE Kyushu-branch*, 87-90, 2011 (in Japanese)

[9] K. Kaneko & I. Tsuda: Chaotic Itinerancy, *Chaos*, **13**: 926-936, 2003

[10] C. A. Skarda & W. J. Freeman: How brains make chaos in order to make sense of the world, *Behavioral and Brain Sci.*, **10**,161-195, 1987

[11] Y. Osana & M. Hagiwara: Successive Learning in hetero-associative memory using chaotic neural networks. *Int. J. Neural Systems*, **9**(4):285-299, 1999

[12] J. Namikawa, R. Nishimoto & J. Tani: A Neurodynamic Account of Spontaneous Behaviour, *PLoS Computational Biology*, **7**(10): e1002221, 2011

[13] D.E. Rumelhart, G.E. Hinton, R.J. Williams: Learning Internal Representation by Error Propagation. In: Parallel Distributed Processing, MIT Press, Cambridge, **1**:318?364, 1986

[14] K. Aihahara: Chaotic Neural Networks, *Bifurcation Phenomena in Nonlinear Systems and Theory of Dynamical Systems*, H. Kawakami ed., World Scientific, 143-161, 1990