Reinforcement Learning of a Memory Task using an Echo State Network with Multi-Layer Readout

Toshitaka Matsuki and Katsunari Shibata

Oita University, 700 Dannoharu, Oita, Japan
{matsuki,shibata}@oita-u.ac.jp

Abstract. Training a neural network (NN) through reinforcement learning (RL) has been focused on recently, and a recurrent NN (RNN) is used in learning tasks that require memory. Meanwhile, to cover the shortcomings in learning an RNN, the reservoir network (RN) has been often employed mainly in supervised learning. The RN is a special RNN and has attracted much attention owing to its rich dynamic representations. An approach involving the use of a multi-layer readout (MLR), which comprises a multi-layer NN, was studied for acquiring complex representations using the RN. This study demonstrates that an RN with MLR can learn a "memory task" through RL with back propagation. In addition, non-linear representations required to clear the task are not observed in the RN but are constructed by learning in the MLR. The results suggest that the MLR can make up for the limited computational ability in an RN.

Keywords: Echo State Network, Reservoir Network, Reservoir Computing, Reinforcement Learning

1 Introduction

Deep learning(DL) has surpassed existing approaches in various fields. It suggests that a successfully trained large scale neural network (NN) that is used as a massive parallel processing system is more flexible and powerful than the systems that are carefully designed by engineers. In recent years, end-to-end reinforcement learning (RL), wherein the entire process from sensors to motors is composed of one NN without modularization and trained through RL, has been a subject of focus[1]. For quite some time, our group has suggested that the end-to-end RL approach is critical for developing a system with higher functions and has demonstrated the emergence of various functions[2]. Recently, deep mind succeeded in training an NN to play Atari video games using the end-to-end RL approach[3]. A feature of this approach is that the system autonomously acquires purposive and general internal representations or functions only from rewards and punishments without any prior knowledge about tasks.

When the system learns proper behaviors with time, it has to handle time series data and acquire necessary internal dynamics. A recurrent structure is

required for the system to learn such functions using an NN. We demonstrated that a recurrent NN (RNN) trained by back propagation through time (BPTT) using autonomously produced training signals based on RL can acquire a function of "memory" or "prediction"[4][5]. However, it is difficult for a regular RNN to acquire complex dynamics such as multiple transitions among states through learning.

BPTT is generally used to train an RNN, but factors such as slow convergence, instability, and computational complexity can cause problems. A reservoir network (RN) such as a liquid state machine, proposed by Jaeger[6], or an echo state network (ESN), proposed by Maass[7], is often used to overcome such issues. The RN uses an RNN called "reservoir," which comprises many neurons that are sparsely connected with each other in a randomly chosen fixed weight. The reservoir captures the history of inputs, receives its outputs as feedback, and forms dynamics including rich information. The outputs of RN are generated as the linear combinations of the activations of reservoir neurons by readout units, and the network is trained by updating only the readout weights from the reservoir neurons to generate the desired values. Therefore, it is easy for the RN to learn to process the time series data and generate complex time series patterns. We believe that the RN can be a key to solving the problems associated with acquiring complex dynamics through learning. In this study, an ESN, which is a kind of RN having rate model neurons, is used. The ESN has been used in various studies, including motor control[8] and dynamic pattern generation[9][10].

To generate outputs that cannot be expressed as linear combinations of dynamic signals in reservoir and inputs from the environment, an approach using more expressive multi-layer readout (MLR), which uses a multi-layer NN (MLNN) trained by back propagation (BP) instead of regular readout units for output generation was studied[11]. Bush and Anderson showed that ESN with MLR can approximate the Q-function in a partially observable environment through Q-learning[12]; Babinec and Pospíchal showed that the accuracy of time series forecasting was improved with this approach[13]. These studies were conducted more than a decade ago. However, we believe that such an architecture will be vital in the future as more complex internal dynamics and computation are required to follow the trend of the increasing importance of end-to-end RL.

In this study, we focus on learning to memorize necessary information from the past and utilize this information to generate appropriate behaviors using an ESN with MLR. We also demonstrate that such functions can be learned by simple BP that does not involve trace back to the past as in BPTT.

2 Method

2.1 Network

The network architectures used in this study are shown in Fig. 1. Instead of single layer readout units, as shown in Fig. 1(a), an RN comprises a multi layer

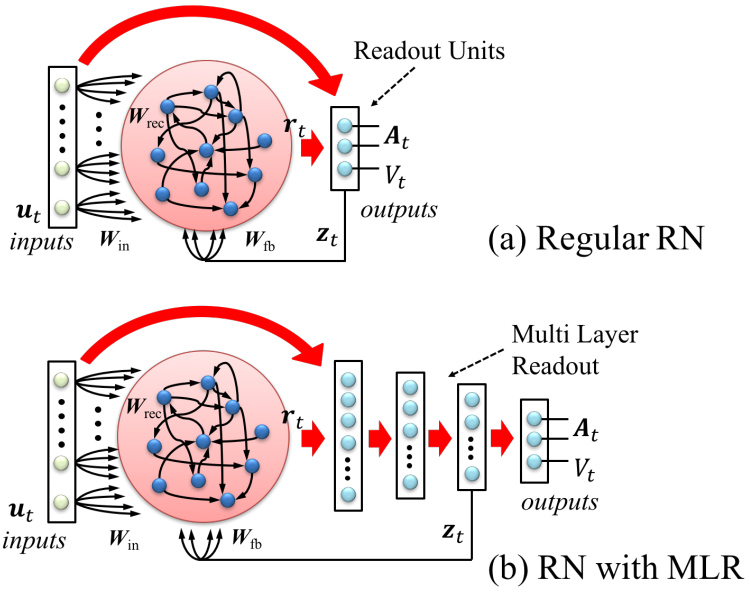


Fig. 1. Network architectures of (a) a regular reservoir network(RN) and (b) an RN with multi-layer readout(MLR).

neural network (MLNN) such as multi-layer readout (MLR), as shown in Fig. 1(b), to generate outputs. The inputs from the environment are provided to the reservoir and readout units or MLNN, and the outputs of the reservoir neurons are provided to the MLNN. The reservoir's capacity for storing large volumes of information and the MLNN's ability of flexibly extracting the necessary information from large volumes of information and generating appropriate outputs are combined. Therefore, it is expected that tracing back to the past with BPTT is no longer required, and memory functions can be acquired only with BP.

The number of reservoir neurons is $N_x = 1000$. The reservoir neurons are dynamical model neurons and are recurrently connected with the connection probability $p = 0.1$. The internal state vector of reservoir neurons at time t, $\boldsymbol{x}(t) \in \mathbb{R}^{N_x}$ is given as

$$\boldsymbol{x}_t = \Big(1 - a\Big)\boldsymbol{x}_{t-1} + a\Big(\lambda \boldsymbol{W}_{rec}\boldsymbol{r}_{t-1} + \boldsymbol{W}_{in}\boldsymbol{u}_t + \boldsymbol{W}_{fb}\boldsymbol{z}_{t-1}\Big), \tag{1}$$

where $a = 0.1$ is a constant value called leaking rate that determines the time scale of reservoir dynamics. $\boldsymbol{W}_{rec} \in \mathbb{R}^{N_x \times N_x}$ is the recurrent connection weight matrix of the reservoir, and each component is set to a value that is randomly generated from a Gaussian distribution with zero mean and variance $1/pN_x$. $\lambda = 1.2$ is a scale of recurrent weights of the reservoir. Larger λ makes the dynamics of the reservoir neurons more chaotic. $\boldsymbol{r}_t$ is the output vector of reservoir neurons. $\boldsymbol{W}_{in} \in \mathbb{R}^{N_x \times N_i}$ is the weight matrix from the input to the reservoir neurons.

$\boldsymbol{W}_{fb} \in \mathbb{R}^{N_x \times N_f}$ is the weight matrix from the MLR or readout units to the reservoir neurons, and $\boldsymbol{z}_t$ is the feedback vector. Each component of $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{fb}$ is set to a uniformly random number between -1 and 1. The activation function of every neuron in the reservoir is the *tanh* function.

The MLR is a four-layer NN with static neurons in the order of $100, 40, 10$ and 3 from the bottom layer; the activation function of each neuron is the *tanh* function. Each neuron in the bottom layer of MLR receives outputs from all the reservoir neurons. The outputs of $N_f = 10$ neurons in the hidden layer, one lower than the output layer, are fed back to every neuron of the reservoir as feedback vector $\boldsymbol{z}_t \in \mathbb{R}^{N_f}$. $N_i = 7$ inputs are derived from the environment; all these inputs are given to each neuron in the reservoir and the bottom layer of MLR. Each initial weight of MLR is set to a randomly generated value from a Gaussian distribution with zero mean and variance $0.01/n$, where n is the number of inputs in each layer. The MLP is trained by BP and the stochastic gradient descent algorithm with a learning rate of 0.01.

The network outputs are critic V_t and actor vector $\boldsymbol{A}_t$. The sum of $\boldsymbol{A}_t$ and the exploration component vector $\boldsymbol{rnd}_t$ is used as the motion signal of the agent at time t. Each component of $\boldsymbol{rnd}_t$ is set to a uniformly random value between -1 and 1.

2.2 Learning

In this network, only the weights of the paths indicated by the red arrows in Fig. 1 are trained using the BP based actor-critic algorithm. The training signal for critic at time $t-1$ is given as

$$V_{t-1}^{train} = V(\boldsymbol{u}_{t-1}) + \hat{r}_{t-1} = r_t + \gamma V(\boldsymbol{u}_t), \tag{2}$$

where $\hat{r}_{t-1}$ is TD-error at time $t-1$, which is given as

$$\hat{r}_{t-1} = r_t + \gamma V(\boldsymbol{u}_t) - V(\boldsymbol{u}_{t-1}), \tag{3}$$

where r_t is a reward received by the agent at time t and $\gamma = 0.99$ is the discount rate. The training signal for actor at time $t-1$ is given as

$$\boldsymbol{A}_{t-1}^{train} = \boldsymbol{A}(\boldsymbol{u}_{t-1}) + \hat{r}_{t-1}\boldsymbol{rnd}_{t-1}. \tag{4}$$

The weights in reservoir $\boldsymbol{W}_{rec}, \boldsymbol{W}_{in}$ and $\boldsymbol{W}_{fb}$ are not trained.

3 Experiment

A memory task was employed to examine the capability of an RN with MLR. Comparing a regular RN and an RN with MLR, we examined the practicality of the parallel and flexible processing capability of MLNN in the memory task.

The outline of the task is shown in Fig. 2. An agent is placed on a 15.0×15.0 plane space. At every step, the agent moves according to the two actor

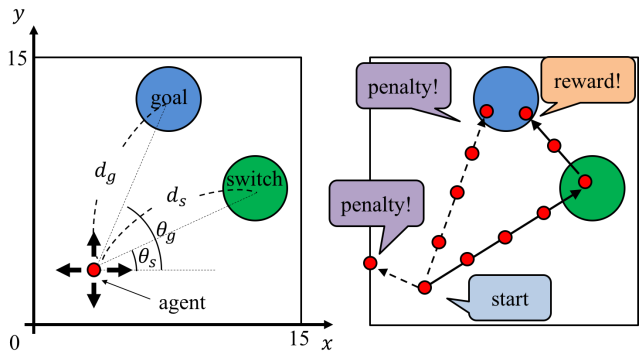


Fig. 2. Outline of the memory task. An agent must first enter the switch area, and then go to the goal area.

outputs each of which determines the moving distance in either x or y directions respectively. The purpose of this agent is to learn the actions needed to first enter the switch area, and then go to the goal. The radius of the goal or switch area is 1.5.

From the environment, an agent receives $N_i = 7$ signals as an input vector

$$\boldsymbol{u}_t = [d'_g,\ sin\theta_g,\ cos\theta_g,\ d'_s,\ sin\theta_s,\ cos\theta_s,\ signal], \tag{5}$$

where d'_g, d'_s are distances to the goal and switch areas, respectively, and are normalized into the interval $[-1, 1]$. θ_g and θ_s are the angles between the x-axis and the goal or switch direction from the agent. Only when the agent is in the switch area, a $signal$ is given as

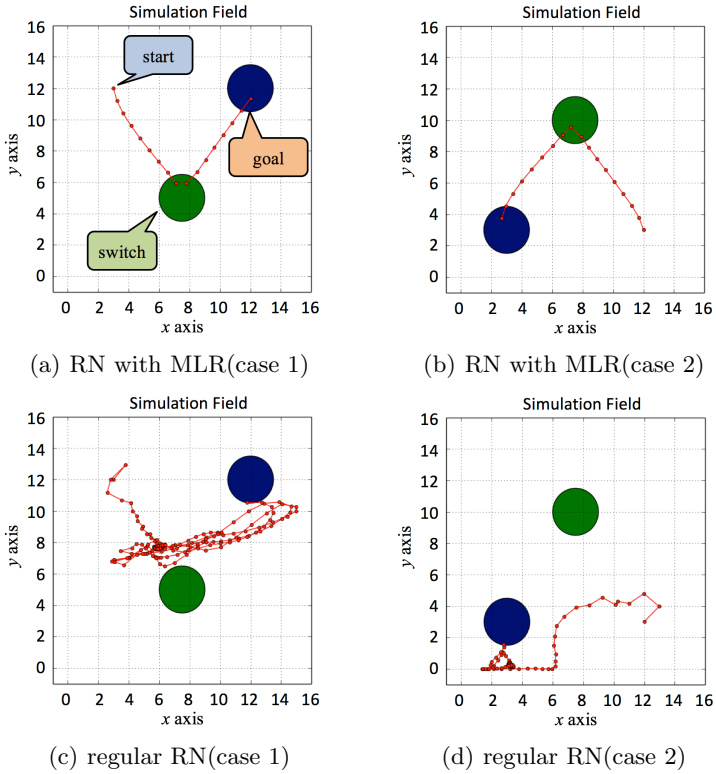
$$signal = \begin{cases} 0 & d_s > R_s \\ 10 & d_s \leq R_s, \end{cases} \tag{6}$$

where R_s is the radius of the switch area. In each trial, the agent, goal and switch are randomly located in the field, and their areas do not overlap with each other. A punishment $r_t = -0.1$ is set for an agent when it contacts the wall, and a punishment $r_t = -0.5$ is set when the agent enters the goal area before the switch area. A reward $r_t = 0.8$ is set when it enters the goal area after entering the switch area. One trial is terminated after the agent acts 200 steps or enters the goal area. After 50,000 trials, the system stops learning.

4 Result

After 50,000 learning trials, the agent behaviors were observed in two cases wherein the sign of each actor output should be changed before and after entering the switch area. The trajectories of the agent in the test trial are shown in Fig.3.

For comparison, the results in the case of RN with MLR are shown in Fig. 3(a) and 3(b) and those in the case of regular RN are shown in Fig. 3(c) and



(a) RN with MLR(case 1) (b) RN with MLR(case 2)

(c) regular RN(case 1) (d) regular RN(case 2)

Fig. 3. Comparison of agent trajectory for two cases between RN with MLR and regular RN.

3(d). As shown in Fig. 3(a) and 3(b), the agent with MLR first entered the switch area after which it entered the goal area. The result shows that this network succeeded in learning a memory task through RL without tracing back to the past as in BPTT. In addition, the network acquired functions to memorize necessary information and generate the desired action signal only with BP. In contrast, as shown in Fig. 3(c) and 3(d), the agent with a regular RN failed to learn the desired behavior. Without entering the switch area, the agent entered the goal area either after wandering in the field (case 1) or remained continually struck to the wall (case 2).

To observe the activation of the reservoir neurons, the switch was fixed at the center of the field, and the goal or the agent was located at one of the four points: $(3, 3), (3, 12), (12, 3)$ and $(12, 12)$; the test trial was then implemented. Various activations were found among the reservoir neurons, but in most cases, it was difficult to find a clear regularity in the activation. In Fig. 4, the network outputs and two characteristic activations of reservoir neurons, in certain cases, are shown with the agent trajectory. In all the cases, the activation of the neuron (1) in Fig. 4 decreases after switching in a similar way. Such neurons remember

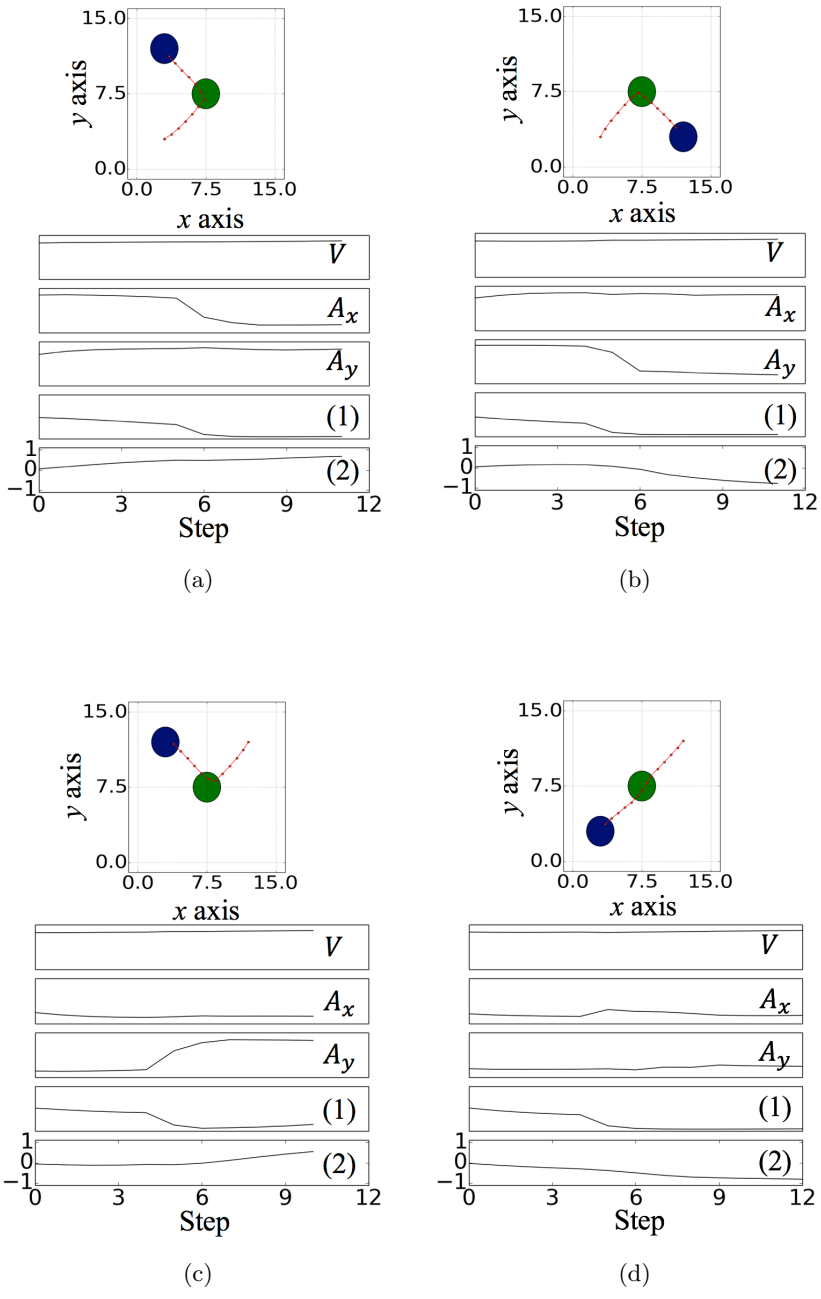


Fig. 4. Outputs of the network and two characteristic reservoir neurons during one trial for 4 cases of goal and initial agent locations. V is critic, and A_x and A_y are the actor outputs for x- and y-axis, respectively.

that the agent has already entered the switch area and contribute to reflect the memory to the outputs of actor. Some other neurons that seems to contribute to the memory function were found.

A non-linear function of present sensor signals and the memorized information that represents whether the agent has already entered the switch area are required to generate appropriate outputs. In the case of Fig. 4(b), by entering the switch area, the y-motion should be changed from "go up" to "go down," whereas in the case of Fig.4(c), it should be changed from "go down" to "go up". Then, to determine whether such outputs are generated in the reservoir, we attempted to find the reservoir neurons with the same sign as the output as that of the actor output for y-axis motion before and after the agent was inside the switch area. The neuron (2) in Fig. 4 is the only one found among the total of 1,000 reservoir neurons. However, the activation pattern of neuron (2), shown in Fig. 4, seems to lag behind the actor output for the y-axis. Then, the feedback connection weight matrix $\boldsymbol{W}_{fb}$ is set to zero and the same test was performed to eliminate the influences from the MLR to the reservoir. In that case, the activation which has a similar feature to the neuron(2) was not observed in the reservoir but the agent could clear the task. This suggests that the activation in Fig. 4(2) appeared under the influences of the MLR through the feedback connections. Considering that the regular RN could not learn the memory task, the non-linear function of memorized information in the reservoir and present sensor signals required to clear the task are not generated in the reservoir but are constructed through learning in the MLR. In other words, the learning of MLNN in MLR is necessary to non-linearly integrate the outputs of reservoir and sensor signals, which enables the switching of actor outputs based on memory.

5 Conclusion

This study demonstrated that an RN with MLR, which generates outputs with an MLNN instead of readout units, can learn a memory task using RL with simple BP without using BPTT. Various activations were observed among the neurons in the reservoir. There were neurons whose activation decreases after switching in a similar way in all cases, and it is considered that these neurons greatly contribute to the function of memory. It was confirmed that non-linear representations, such as changes in actor outputs before and after reaching the switch area, were not observed in the reservoir dynamics; however, owing to the expressive MLNN, the MLR learns to construct the non-linear representations and successfully generate appropriate actor outputs. Our future study includes applying this frame to more complex tasks; analyzing the length of time the network can continue to store necessary information; analyzing how the feedback from a hidden layer of MLR to the reservoir influences reservoir dynamics; developing a method to train input weights of the reservoir to extract necessary information from inputs; and verifying whether our already proposed RL

method [14][15], in which chaotic internal dynamics in a NN are used as exploration components, is applicable.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 15K00360.

References

1. Y.LeCun, Y.Bengio and G.Hinton: Deep learning. Nature 521, 436−444 (2015)
2. K.Shibata: Functions that Emerge through End-to-end Reinforcement Learning.: arXiv preprint arXiv:1703.02239 (2017)
3. V.Mnih, K.Kavukcuoglu, D.Silver, A.Graves, I.Antonoglou, D.Wierstra, and M.Riedmiller: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.(2013)
4. K.Shibata and H.Utsunomiya: Discovery of pattern meaning from delayed rewards by reinforcement learning with a recurrent neural network, Proc. of IJCNN., pp. 1445−1452(2011)
5. K.Shibata and K.Goto: Emergence of flexible prediction-based discrete decision making and continuous motion generation through Actor-Q-Learning, Proc. of ICDL-Epirob. 2013, ID 15 (2013)
6. 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 (2001): 13.
7. W.Maass, T.Natschlger, and H.Markram: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural computation 14.11, 2531−2560 (2002)
8. M.Salmen, and P.G.Ploger: Echo state networks used for motor control. Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE (2005)
9. D.Sussillo, L.F.Abbott: Generating coherent patterns of activity from chaotic neural networks. Neuron Article, Vol.63, No.4, pp.544-557(2009)
10. G.M.Hoerzer, R.Legenstein and W.Maass: Emergence of complex computational structures from chaotic neural networks through Reward-Modulated Hebbian Learning. Cerebral Cortex, Vol.24 No.3, pp.677−690 (2014)
11. M.Lukusevičius and H.Jaeger: Reservoir computing approaches to recurrent neural network training.: Computer Science Review 3.3, pp.127−149. (2009)
12. K.Bush, and C.Anderson: Modeling reward functions for incomplete state representations via echo state networks.: Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on. Vol. 5. IEEE (2005)
13. Š.Babinec, and J.Pospchal: Merging echo state and feedforward neural networks for time series forecasting.: Artificial Neural NetworksICANN 2006, pp.367−375. (2006)
14. Y.Goto and K.Shibata: Emergence of Higher Exploration in Reinforcement Learning Using a Chaotic Neural Network, Proc. of Int'l Conf. on Neural Information Processing (ICONIP)2016, LNCS 9947, pp. 40−48 (2016)
15. T.Matsuki and K.Shibata: 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 (ICONIP)2016, LNCS 9947, pp. 376-383 (2016)