2017 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2017, Cancun, Mexico, December 4-7, 2017

A Complex-Valued Reinforcement Learning Algorithm Using Complex-Valued Neural Networks

Masaki Mochida†, Hidehiro Nakano† and Arata Miyauchi†

†Faculty of Knowledge Engineering, Tokyo City University
1–28–1 Tamazutsumi, Setagaya-ku, Tokyo 158-8557, Japan
Email: g1781530@tcu.ac.jp, hnakano@tcu.ac.jp, miyauchi@ic.cs.tcu.ac.jp

Abstract—In Complex-valued Reinforcement Learning (CRL), each action-value is represented by a complex value. Then, search history can be naturally included in the argument, while dominance relationships are decided by the amplitude. CRL is effective for the environments with perceptual aliasing. In order to apply larger-problems with the large number of states, this paper introduces the function approximation for the action-value function by using complex-valued neural networks. The simulation results for a benchmark problem are shown.

1. Introduction

For autonomous machines, it is necessary to generate behavior rules. However, it is difficult for designers comprehensively to describe them because there is uncertainty in the real world. Therefore, there is Reinforcement Learning (RL) as a method to let the machines learn behavior rules autonomously.

RL is a method to learn behavior rules by trial and error with the environment as a learning object [1]. In RL, an agent that is a learner observes its own state from the environment and selects an action based on that state. Then, the agent evaluates the action based on the rewards thus obtained, and learns better behavior rules. For RL, there are some conditions to be considered when applying to real problems. One of them is perceptual aliasing.

Perceptual aliasing occurs when sensors with enough ability are not available. This makes it difficult for the agent to identify the state, causing a problem in selecting the action. To solve this problem, we can improve the discrimination ability of the current state by memorizing past observations and actions. However, this requires sufficient memory size and computational resources. As a method for avoiding this problem, there is Complex-valued Reinforcement Learning (CRL)[2].

In CRL, action values are represented by complex values. Then, information on the path length of the propagated reward is incorporated in the action values. By using this information, the ability to distinguish each state can be improved. In addition, CRL doesn't require so much memory size and computation resources.

In RL, there are cases where continuous values are handled in expression of states. It requires fine discretization for them, but action value space becomes large-scale. This leads to an increase in required memory size and a decrease in learning speed. This problem can be dealt with by expressing the action value space by a function approximator. There is a conventional method in which the action value function of CRL is represented by a complex-valued Radial Basis Function (RBF) networks[3].

In this paper, we apply a complex-valued neural networks[4] as a function approximator of the action value function in CRL. A complex-valued neural network is a neural network in which network parameters and variables are complex values. In the conventional method, plural networks were prepared corresponding to the number of actions, whereas in the proposed method, actions are taken as inputs to the network. This can share the network between actions and prevent an increase in the network size due to an increase in the number of actions. Simulation results are shown in the Mountain Car task [6] where the state space is continuous.

2. Complex-valued Reinforcement Learning using Complex-valued Neural Networks

2.1. $\dot{\mathrm{Q}}$-learning

$\dot{\mathrm{Q}}$-learning[2] is a kind of CRL algorihtms. The update equations of complex-valued action value $\dot{Q}(x_t, a_t)$ are shown below.

$$\dot{Q}(x_t, a_t) \leftarrow (1-\alpha)\dot{Q}(x_t, a_t) + \alpha\left[r_{t+1} + \gamma\dot{Q}(x_{t+1}, a'_{t+1})\right]e^{i\omega} \quad (1)$$

$$a'_{t+1} = \underset{a\in A_{(x_{t+1})}}{\operatorname{argmax}} \operatorname{Re}\left(\dot{Q}(x_{t+1}, a)\bar{\dot{I}}_t\right) \quad (2)$$

where x is an observed state, a is an action, A is a set of actions, r is the reward obtained by the action and t is time-step. α, γ, ω are parameters, which is the learning rate, the discount rate of the propagated reward, and the rotational amount of phase, respectively. $\dot{Q}(x_t, a_t)$ expresses the cumulative reward value by the absolute value and expresses the path length of the propagated reward by the phase on the complex plane. According to Eq.(1), by performing phase rotation on the propagated reward when updating, continuity is given to the neighboring states, and the path length of the propagation reward is added to the action

value. $\dot{\mathrm{Q}}$-learning uses the internal reference value $\dot{I}_t$ which represents the context of the agent for action selection. The internal reference value is the following expression.

$$\dot{I}_t = \begin{cases} \dot{Q}(x_t, a_t)e^{-i\omega} & for \ \ t \geq 0 \\ \dot{Q}\left(x_0, \underset{a \in A_{(x_0)}}{\operatorname{argmax}} \left|\dot{Q}(x_0, a)\right|\right) & for \ \ t = -1 \end{cases} \quad (3)$$

The internal reference value holds the previous action value as the context of the agent, and the agent refers to the internal reference value so that it is possible to select the action along the propagation path of the reward from the continuity of the phase. Then, $\dot{\mathrm{Q}}$-learning make it possible to distinguish states which are equated by perceptual aliasing. Such action selection is performed using the inner product of complex-valued action value and internal reference value as follows.

$$\mathrm{Re}\left(\dot{Q}(x_t, a)\bar{\dot{I}}_{t-1}\right) \quad (4)$$

By using this, it is easier to select an action whose absolute value is large and whose phase is closer to the internal reference value. Namely, an action with higher cumulative reward value and continuity with internal reference value is more likely to be selected.

2.2. $\dot{\mathrm{Q}}$-learning using Complex-valued Neural Networks

A complex-valued neural network[4] is a neural network in which network parameters and variables are complex values. Complex-valued neural networks can express any complex-valued input-output relationships by changing parameter values. This makes it possible to handle a complex-valued neural network as a complex number function approximator. In addition, complex-valued neural networks are said to be able to represent complex values well because of the feature that parameters and variables are complex values.

The complex-valued action value function approximated by a complex-valued neural network is shown below.

$$\dot{Q}(\boldsymbol{x}, \boldsymbol{a}) = \sum_{k=1}^{H} \dot{w}_k^o \dot{\xi}_k + \dot{b}^o \quad (5)$$

$$\dot{\xi}_k = f\left(\sum_{j=1}^{N} \dot{w}_{kj}^h \dot{X}_j + \dot{b}_k^h\right) \quad k \in \{1 \ldots H\} \quad (6)$$

$$\dot{\boldsymbol{X}} = (x_1, \ldots, x_n, a_1, \ldots, a_m) \quad (7)$$

$$f(\dot{u}) = \tanh(\mathrm{Re}(\dot{u})) + i \tanh(\mathrm{Im}(\dot{u})) \quad (8)$$

where $\dot{w}_{kj}^h$ and $\dot{b}_k^h$ are the parameters of the hidden layer, and $\dot{w}_k^o$ and $\dot{b}^o$ are those of the output layer. N is the number of input layer neurons. Assuming that the number of dimensions of states is n and the number of dimensions of actions is m, $N = n + m$. H is the number of hidden layer neurons. In the conventional method using the complex-valued RBF network[3], only the state is input and different

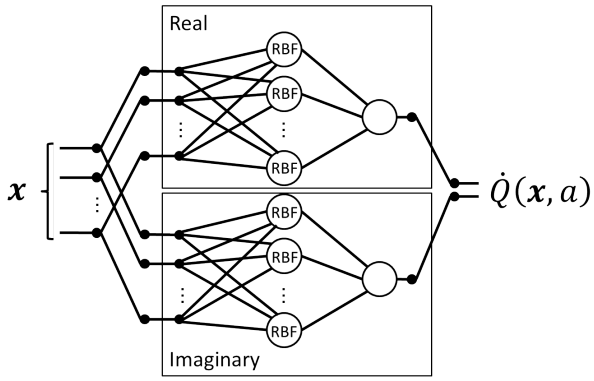

Figure 1: Complex-valued RBF network for a single action

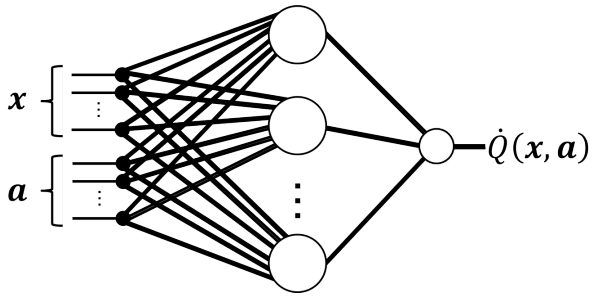

Figure 2: Complex-valued neural network

networks are prepared for each action. A conceptual diagram of a complex-valued RBF network is shown in Fig.1. This is a single network for a single action. Therefore, multiple networks are required corresponding to the number of actions. On the other hand, in our method, the action value function is expressed in a single network without preparing multiple networks by inputting both states and actions. A conceptual diagram of a complex-valued neural network is shown in Fig.2.

The complex-valued neural network learns using complex back propagation[5]. The following error function is used in network learning.

$$E = \frac{1}{2}\left|\left[r_{t+1} + \gamma \dot{Q}(x_{t+1}, a'_{t+1})\right] e^{i\omega} - \dot{Q}(x_t, a_t)\right|^2 \quad (9)$$

The update equations for each network parameter are the followings.

$$\dot{\delta}_t = \left[r_{t+1} + \gamma \dot{Q}(x_{t+1}, a'_{t+1})\right] e^{i\omega} - \dot{Q}(x_t, a_t) \quad (10)$$

$$\begin{aligned} \Delta \dot{b}^o &= -\alpha^o \frac{\partial E}{\partial \, \mathrm{Re}(\dot{b}^o)} - i\alpha^o \frac{\partial E}{\partial \, \mathrm{Im}(\dot{b}^o)} \\ &= \alpha^o \dot{\delta}_t \end{aligned} \quad (11)$$

$$\begin{aligned} \Delta \dot{w}_k^o &= -\alpha^o \frac{\partial E}{\partial \, \mathrm{Re}(\dot{w}_k^o)} - i\alpha^o \frac{\partial E}{\partial \, \mathrm{Im}(\dot{w}_k^o)} \\ &= \Delta \dot{b}^o \bar{\dot{\xi}}_k \end{aligned} \quad (12)$$

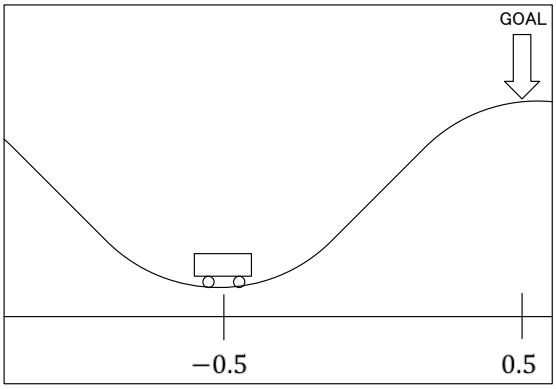

Figure 3: Mountain Car Task

$$\begin{aligned}\Delta\dot{b}_k^h &= -\alpha^h\frac{\partial E}{\partial\,\mathrm{Re}(\dot{b}_k^h)} - i\alpha^h\frac{\partial E}{\partial\,\mathrm{Im}(\dot{b}_k^h)} \\ &= \alpha^h\Big\{(1-\mathrm{Re}(\dot{\xi}_k)^2)(\mathrm{Re}(\dot{\delta}_t)\,\mathrm{Re}(\dot{w}_k^o)+\mathrm{Im}(\dot{\delta}_t)\,\mathrm{Im}(\dot{w}_k^o)) \\ &\quad + i(1-\mathrm{Im}(\dot{\xi}_k)^2)(\mathrm{Im}(\dot{\delta}_t)\,\mathrm{Re}(\dot{w}_k^o)-\mathrm{Re}(\dot{\delta}_t)\,\mathrm{Im}(\dot{w}_k^o))\Big\} \qquad (13)\end{aligned}$$

$$\begin{aligned}\Delta\dot{w}_{kj}^h &= -\alpha^h\frac{\partial E}{\partial\,\mathrm{Re}(\dot{w}_{kj}^h)} - i\alpha^h\frac{\partial E}{\partial\,\mathrm{Im}(\dot{w}_{kj}^h)} \\ &= \Delta\dot{b}_k^h\bar{\dot{X}}_j \qquad (14)\end{aligned}$$

3. Experiments

3.1. Environment

In order to verify the performance of our method in continuous space, we experiment with the Mountain Car task[6] whose states are continuous. The shape of the environment is shown in Fig.3. An agent has a position and a velocity as a state, and determines an acceleration direction by an action. The velocity is updated by the position and action, and the position is updated by the velocity. The velocity and position updating equations are shown below.

$$v \leftarrow v + 0.001a - 0.0025\cos(3x) \qquad (15)$$

$$x \leftarrow x + v \qquad (16)$$

where x is the position and v is the velocity. The range of each value is $-0.07 \le v \le 0.07$, and $-1.2 \le x \le 0.5$. a is the action which is either of left acceleration (-1), no acceleration (0), right acceleration (1). The initial state is $x = -0.5$, $v = 0$, and reward is given when reaching the target state $x \ge 0.5$. The target state is on the top of the mountain, and in order to climb the mountain, it is necessary to make momentum from the opposite mountain. Here, we restrict the perception of the agent to only the position, causing perceptual aliasing. That is, we experiment with the number of dimensions of states, $n = 1$, the number of dimensions of actions, $m = 1$, and the number of inputs, $N = 2$.

Table 1: Parameters

	(Proposed)	(Conventional)	(Table)
N	2	1	—
H	30		
α^o	0.001	0.01	0.1
α^h	0.0001	0.001	
γ	0.7		0.99
ω	$\pi/180$		
r	100		

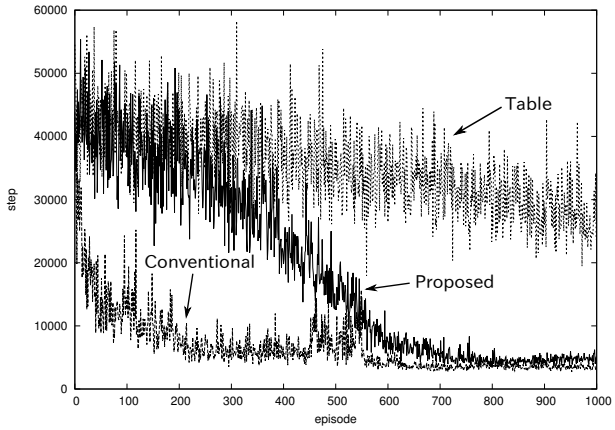

Figure 4: Results

3.2. Simulation settings

We compare the proposed method with the usual $\dot{Q}$-learning using a table function (Table) and the conventional method expressing the action value function with a complex-valued RBF network (conventional). The parameters of each method are shown in Table 1. In each method, an action selection method is Boltzmann selection. Boltzmann temperature is set to $\tau = 0.5$ in each method. In a usual $\dot{Q}$-learning using a table function, since the state requires fine discretization, the value is discretized up to the third decimal place. In the complex-valued neural network, the value range of the input is changed so as to be all positive values. That is, the value set of the input action was changed from $\{-1, 0, 1\}$ to $\{0, 5, 10\}$. One action selection is one step, and the seaquence from the initial state to reaching the target state is one episode. 1000 episodes executions is one trial. The number of trials is 50.

3.3. Results

The experimental results are shown in Fig.4. Although the number of steps is gradually decreasing in the $\dot{Q}$-learning using a table function, the learning speed is slow

and the learning does not converge even at 1000 episode. This is because the number of the discretized states in the task is large, and the large number of episodes are necessary for the learning. On the other hand, in the methods using the function approximators, since the action values of the unlearned states can also be generalized from the experienced action values, the learning speed is fast. In the $\dot{Q}$-learning using complex-valued RBF network, independent networks are used for each action, whereas the proposed method shares a single network. Still at 1000 episodes, both methods have converged to almost the same number of steps.

Since the proposed method expresses the action value function in a single network, the required number of network parameters is smaller than that of the conventional method using independent networks. That is, the proposed method requires less memory size.

In this experiment, the number of network parameters in $\dot{Q}$-learning using complex-valued RBF network is 540, and that in the proposed method is 242. When the number of actions is larger, it seems that approximation by the proposed method becomes more difficult, but the difference in the number of network parameters with the conventional method becomes wider; the proposed method can be applied to larger-scale environments easily.

4. Conclusion

We proposed a method to approximate the action value function by a complex-valued neural network for complex-valued reinforcement learning, which is a method coping with perceptual aliasing. Simulation experiments were conducted in Mountain Car tasks, and it is confirmed that the proposed method can learn even in an environment with the continuous states and perceptual aliasing. We also compared the proposed method with complex-valued reinforcement learning using a complex-valued RBF network which is an existing method, and finally obtained almost the same learning results. The proposed methods can suppress an increase of the number of network parameters for environments with the large number of actions.

The future tasks are to verify the performance in other environments and to examine the influence on the performance against the increase in the number of actions.

References

[1] Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning:An Introduction,” MIT press, 1998.

[2] T. Hamagami, T. Shibuya, and S. Shimada, “Complex-Valued Reinforcement Learning,” in Proceedings of IEEE international Conference on the Systems. Man and Cybernetics 2006, vol. 5, pp .4175–4179, 2006.

[3] Takeshi Shibuya, Hideaki Arita and Tomoki Hamagami, ”Reinforcement Learning in Continuous State Space with Perceptual Aliasing by using Complex-valued RBF Network,” Systems Man and Cybernetics (SMC), 2010 IEEE International Conference, pp1799–1803, Istanbul, Turkey, Oct.2010.

[4] A. Hirose, “Advantages and their origins of complex-valued neural networks (in Japanese),” IEICE Technical Report Vol. 109(53), pp7–12, May, 2009.

[5] T. Nitta, “An extension of the back-propagation algorithm to complex numbers,” Neural Networks 10(8), pp1391–1415, 1997.

[6] Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning:An Introduction,” MIT press, pp234–235, 1998.