A Neuro-Fuzzy Reinforcement Learning System for Autonomous Robot Dealing with Continuous Space

[1]Takashi Kuremoto, [1]Kazuhiro Tsubaki, [1]Masanao Obayashi, [1]Shingo Mabu, and [2]Kunikazu Kobayashi

[1]Yamaguchi University
2-16-1, Tokiwadai, Ube, Yamaguchi, 755-8611, Japan
Phone/FAX: +81-836-9500
E-mail: {wu, m.obayas, mabu}@yamaguchi-u.ac.jp
[2]Aichi Prefectural University
1522-3, Ibaragabasama, Nagakute, Aichi, 480-1198, Japan
E-mail: kobayashi@ist.aichi-pu.ac.jp

Abstract

In this study, to tackle the goal-navigated unknown environment exploration problem, a novel method which uses a neuro-fuzzy reinforcement learning system with continuous input-output spaces is proposed. When the position of the explorer is specified, learning process becomes to a Markov Decision Process (MDP) and the aliasing problem does not occur. So it is available to find the optimum. Specially, an action decision method using Gaussian function is proposed to output the movements in the arbitrary direction. Comparing with the conventional discrete action vectors, i.e., usually being 4 directions (up, down, left, and right), the proposed continuous actions may cause the reduction of the length of the shortest path.

Using different environments such as a U form maze, a simple exploration plane, and a complex exploration plane with obstacles, the effectiveness of the proposed system was confirmed by simulation experiments.

1. Introduction

Artificial Intelligence (AI) attracts scientists and ordinary people from last century and becomes more available to be realized for the rapid development of computer technology recently. Suppose that a machine, e.g., a robot is not equipped with pre-knowledge of the environment, AI lets it learn adaptive behaviors to deal with the unknown environment. Reinforcement Learning (RL), a typical AL method, is a kind of machine learning algorithm which results the learner (robot) to find the optimal solution via exploration and exploitation, i.e., a trial-and-error method [1].

There are 4 elements in RL:

- State: the state of the environment (observed by the learner, or the supervisor), the input to a RL system;
- Policy: how to output (act) considering the state (situation);
- Action: the output of the RL system, the behavior of the learner;
- Reward or punishment: the influence from the environment to the learner.

Suppose there are limited states in the environment, the transmission of state happens according to the current state and the action executed by the learner. The learning process, which is to find the optimal transmission of states, becomes to be a Markov Decision Process (MDP). RL algorithms are given to find the optimum solution by modifying the policy using the reward/punishment from the environment.

When the input and output spaces are observed discretely, RL problems are easy solved. However, they are usually continuous spaces in the real world. In [2], Samejima & Omori proposed an Adaptive Basis Division (ABD) algorithm to decide the states of the environment. In [3]-[6], Kuremoto et al. used a Self-Organized Fuzzy Neural Network (SOFNN) to classify the input as the states. However, few works provide method to deal with continuous actions spaces in RL. The reason can be considered that when the action space is continuous, the state-action value function, i.e., Q function becomes infinite, MDP is not impossible, and the trial cannot be finished.

In this paper, an action reconstruction method using Gaussian distribution of discrete Q function is proposed. The proposed method is based on the neuro-fuzzy reinforcement learning system proposed in our previous works [3]-[6], i.e., discriminate the observed state with SOFNN, and then output Q values with assumed action vectors. The continuous action is generated according to a Gaussian distribution with the mean of Q values and a certain standard deviation. Additionally, different from the previous works used local observation information, world

coordinate data (absolute position) is utilized. Simulation experiments of 2-D goal –navigation problem with kinds of environments were performed to verify the effectiveness of the proposed method.

2. RL system with SOFNN

A reinforcement learning system using the Self-Organized Fuzzy Neural Network (SOFNN) [3]-[6] is shown in Fig. 1 and Fig. 2.

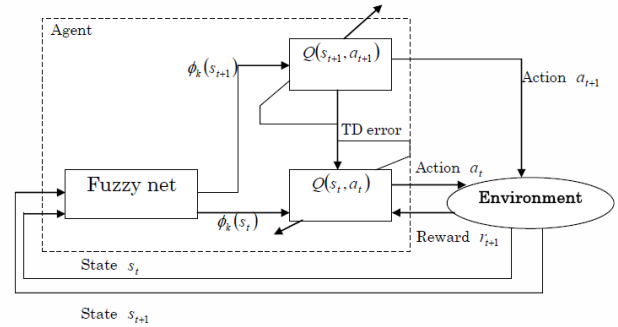


Fig1: A RL system with SOFNN

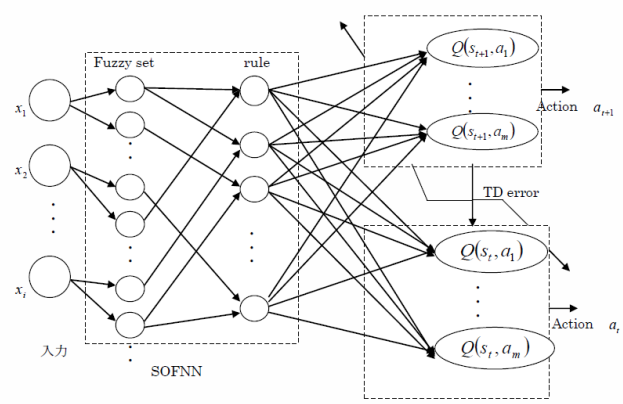


Fig2: SOFNN

For an *n*-D input state space $\mathbf{x}(x_1(t), x_2(t),..., x_n(t))$, a fuzzy inference net is designed with a hidden layer composed by units of fuzzy membership functions $B_i^k(x_i(t))$, i.e., Eq. (1), to classify input states.

$$B_i^k(x_i(t)) = \exp\left\{-\frac{(x_i(t) - c_i^k)^2}{2\sigma_i^{k\,2}}\right\} \tag{1}$$

Here c_i^k , σ_i^k denotes the mean and the deviation of *i*th membership function which corresponding to *i*th dimension of input $x_i(t)$, respectively.

Let $K(t)$ be the largest number of fuzzy rules, we have Eq. (2) :
if ($x_1(t)$ is $B_1^k(x_1(t))$, ,... , $x_n(t)$ is $B_n^k(x_n(t))$) then

$$\phi_k(\mathrm{x}(t)) = \prod_{i=1}^{n} B_i^k(x_i(t)) \tag{2}$$

where $\phi_k(\mathrm{x}(t))$ means the fitness of the rule R^k for an input set $\mathrm{x}(t)$.

To determine the number of membership functions and rules of fuzzy net, a self-organized fuzzy neural network (SOFNN) which is constructed adaptive membership functions and rules driven by training data and thresholds automatically. The self-organizing process of SOFNN is given as follows.

Only one membership function is generated by the first input data (for example, the position of agent) , the value of its membership's center equals to the value of input data, and the value of width of all Gaussian function units is fixed to an empirical value. The number of rule for membership functions is one at first, and the output of the rule R^1 equals to $\phi_1(\mathbf{x}(1)) = \prod_{i=1}^{n} B_i^1(x_i(1))$ according to Eq. (2).

For the next input state $\mathbf{x}(x_1(t), x_2(t),\ldots,x_n(t))$, a new membership function is generated if Eq. (3) is satisfied.

$$\max_s B_{i,s}(x_i(t)) < F \tag{3}$$

Here $B_{i,s}(x_i(t))$ denotes the value of existed membership functions calculated by Eq. (4) and $s = 1,2,\ldots,L_i(t)$ indicates the *s*th membership function with the maximum number $L_i(t)$. F denotes a threshold value of whether an input state is evaluated enough by existing membership functions.

A new rule is generated automatically when a new membership function is added according to Eq. (3). The membership function with the highest values in each dimension of input connects to the new rule. Iteratively, the fuzzy net is completed to adapt to input states.

State-action value functions in different observing time *t* are given by following:

$$Q(\phi(\mathbf{x}(t)), a_j, \mathbf{w}_j^{Q_i}) = \frac{\sum_k w_{kj}^{Q_i} \phi^k(\mathbf{x}(t))}{\sum_k \phi^k(\mathbf{x}(t))} \tag{4}$$

$$Q(\phi(\mathbf{x}(t+1)), a_j, \mathbf{w}_j^{Q_{t+1}}) = \frac{\sum_k w_{kj}^{Q_{t+1}} \phi^k(\mathbf{x}(t+1))}{\sum_k \phi^k(\mathbf{x}(t+1))} \quad (5)$$

where w is the weight of connection, $k = 1,2,...K^t$ is the number of fuzzy rules, $j = 1,2,...m$ is action number, a is the candidate action.

The learning rule of Q Learning (QL) and sarsa learning [1] becomes to:

$$w_{kj}^{Q_t} \leftarrow w_{kj}^{Q_t} + \begin{cases} \alpha_t \delta \phi_k(\mathbf{x}(t)) / \sum_{k=1}^{K} \phi_k(\mathbf{x}(t)) & a_t = a_j \\ 0 & otherwise \end{cases} \quad (6)$$

$$w_{kj}^{Q_{t+1}} \leftarrow w_{kj}^{Q_t+1} + \begin{cases} \alpha_{t+1} \delta \phi_k(\mathbf{x}(t+1)) / \sum_{k=1}^{K} \phi_k(\mathbf{x}(t+1)) & a_{t+1} = a_j \\ 0 & otherwise \end{cases} \quad (7)$$

where α is the learning rate, δ is the temporal difference error (TD error) given as following.

$$\delta_{TD}^{QL} = r_t + \gamma (\max Q(\phi(\mathbf{x}(t)), a_t, \mathbf{w}_t^{Q_t}) - Q(\phi(\mathbf{x}(t)), a_t, \mathbf{w}_t^{Q_t})) \quad (8)$$

$$\delta_{TD}^{sarsa} = r_t + \gamma Q(\phi(\mathbf{x}(t+1)), a_{t+1}, \mathbf{w}_{t+1}^{Q_{t+1}}) - Q(\phi(\mathbf{x}(t)), a_t, \mathbf{w}_t^{Q_t}) \quad (9)$$

3. Continuous Action

Suppose there are 4 pseudo action directions: up, down, left, and right as shown in Fig. 3. The origin of the coordinate axes indicates the current position of the explorer in a 2-D unknown environment. The direction of the movement from the current position is given by a Gaussian distribution of Q values. The mean of the probability function (x, y) is the mean of Q values (see Fig. 3), and the standard deviation is a constant. A dot shown in the Fig. 3 indicates a random position generated according to the Gaussian distribution, and it is utilized as the direction of the continuous action. The length of the movement can be defined as 1.0 each step.

4. Simulation Experiments and Results

4. 1 Simulation I: A U Form Maze

A simple maze with U form was used in simulation I (Fig. 4(a)). The explorer starts from S position and intends to find goal G. Dark area and lines indicates obstacles, and explorer without any pre-knowledge to decide moving up, down, left, or right (1 step 1 direction, 1.0 length). This simulation is designed to compare the performance of the RL system with SOFNN when the input in different cases: local observation (conventional method), and coordinate information (proposed method). The input and output spaces are discrete.

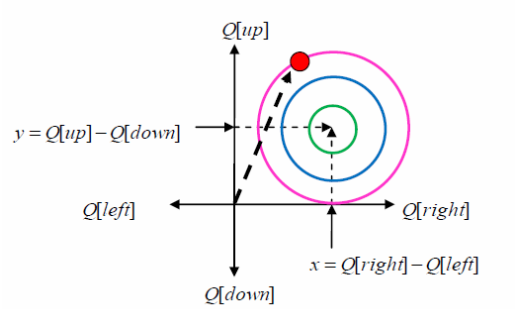
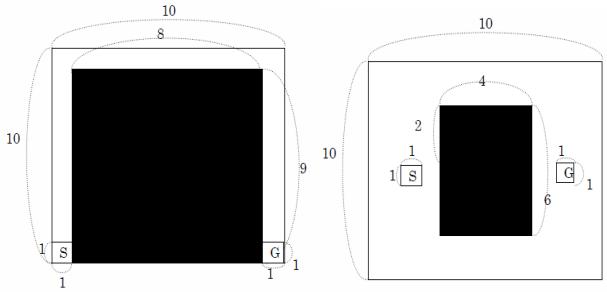


Fig3: Decide the action direction using a Gaussian distribution of Q function



(a) A U form maze (b) A simple maze

Fig4: Exploration environments used in simulations

The average exploration steps of one trial during learning process are used to compare the performance of the different input method. In Table 1, the input used coordinate information (proposed method) shows its priority to conventional local observation method in both kinds learning algorithm Q learning and Sarsa learning.

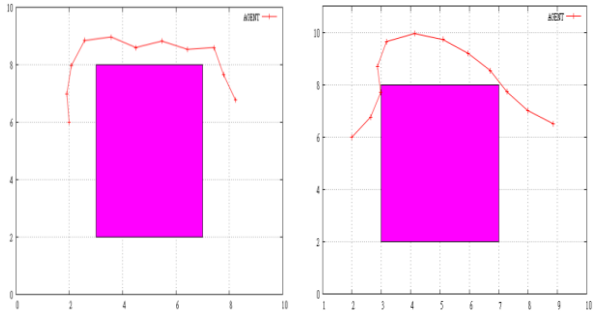
Table 1: Result of Simulation I (average exploration steps of one trial during learning process)

	Local observation (conventional)	Coordinate data (proposed)
Q Learning	1201.98	139.69
Sarsa Learning	1471.56	55.24

4.2 Simulation II: A Simple Maze

A simple maze exploration problem as shown in Fig. 4(b) was designed in simulation II to investigate the effectiveness of the proposed method, i.e., using coordinate data in discrete space or continuous space. The final routes found by different methods are shown in Fig. 5. It can be confirmed that both of Q learning (QL) and

sarsa learning (SL) made the explorer achieved at the goal, however, the continuous output resulted longer distance compare with the discrete output method (Table 2).



(a) QL (continuous output) (b) SL (continuous output)
Fig5: Final routes found by different methods

Table 2: Result of Simulation II (average exploration steps of one trial during learning process)

	Discrete (proposed)	Continuous (proposed)
Q Learning	13.0	26.06
Sarsa Learning	13.0	13.91

4.3 Simulation III: Autonomous Robot

The proposed system was used to control an autonomous robot e-puck in the simulator Webots [7]. Fig. 6 shows a snap shot of the exploration simulation. The average route length for one trial of the robot during learning process is shown in Table 3. Discrete output with Q Learning showed better performance than continuous one, meanwhile, Sarsa Learning opposite.

Table 3: Result of Simulation III (average exploration steps of one trial during learning process)

	Discrete (proposed)	Continuous (proposed)
Q Learning	60.22	85.16
Sarsa Learning	127.37	106.02

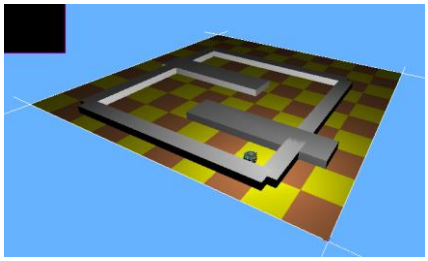


Fig6: A robot e-puck explores an unknown environment with obstacles

(URL of video: http://www.nn.csse.yamaguchi-u.ac.jp/home/wu/image/webots2014tsubaki.avi)

5. Conclusions

A neuro-fuzzy reinforcement learning system was improved by using coordinate information as input. To deal with the real world exploration problem for autonomous robots, continuous action was generated by a Gaussian distribution function of the state-action value function (Q value). Simulation results showed the effectiveness of the proposed method.

Acknowledgment

This work was supported by JSPS KAKENHI Grant No. 26330254 and No.25330287.

References

[1] R. S. Sutton, and A. G. Barto, Reinforcement learning: an introduction, *The MIT Press*, Cambridge, 1998

[2] K. Samejima and T. Omori, "Adaptive Internal State Space Construction Method for Reinforcement Learning of a Real-World Agent", *Neural Networks*, Vol.12, pp.1143—1155, 1999

[3] T. Kuremoto , M. Obayashi , and K. Kobayashi, "Adaptive Swarm Behavior Acquisition by a Neuro-Fuzzy System and Reinforcement Learning Algorithm", *International Journal of Intelligent Computing and Cybernetics*, Vol.2, No.4, pp.724-744, 2009

[4] T. Kuremoto, Y. Yamano, M. Obayashi, and K. Kobayashi, "An Improved Internal Model for Swarm Formation and Adaptive Swarm Behavior Acquisition", *Journal of Circuits, Systems and Computers*, Vol.18, No.8, pp.1517-1531, 2009

[5] T. Kuremoto, Y. Yamano, L-B. Feng, K. Kobayashi, and M. Obayashi, "A Neuro-Fuzzy Network with Reinforcement Learning Algorithm for Swarm Learning", *Lecture Notes in Electronic Engineering*, Vol.304, pp.815-823, 2011

[6] T. Kuremoto, Y. Yamano, L-B. Feng, K. Kobayashi, and M. Obayashi: "Adaptive Swarm Behavior Acquisition Using a Neuro-Fuzzy Reinforcement Learning System", *IEEJ Transactions on EIS* (in Japanese), Vol.133-C, No.5, pp.1076-1085, 2013

[7] Webots: https://www.cyberbotics.com/