A Self-Organized Fuzzy-Neuro Reinforcement Learning System for Continuous State Space for Autonomous Robots

Masanao Obayashi[1], Takashi Kuremoto[1] and Kunikazu Kobayashi[1]

[1] *Division of Computer Science & Design Engineering, Yamaguchi University, Ube, Japan*

E-mail:{ m.obayas,wu,koba}@yamaguchi-u.ac.jp

Abstract

This paper proposes the system that combines self-organized fuzzy-neural networks with reinforcement learning system (Q-learning, stochastic gradient ascent : SGA) to realize the autonomous robot behavior learning for continuous state space. The self-organized fuzzy neural network works as adaptive input state space classifier to adapt the change of environment, the part of reinforcement learning has the learning ability corresponding to rule for the input state space . Simultaneously, to simulate the real environment the robot has ability to estimate own-position. Finally, it is clarified that our proposed system is effective through the autonomous robot behavior learning simulation by using the khepera robot simulator.

1. Introduction

Reinforcement learning (R.L.) is a framework for an agent to learn the choice of an optimal action to adapt unknown environment based on a reinforcement signal [1]. It has been applied to a variety of problem such as robust control [3]-[5], nonlinear prediction [6], swarm behavior learning [7] and so on.

There are Q-learning, actor-critic and stochastic gradient ascent and so on as representative reinforcement learning methods. Q-Learning is usually suitable for discrete state and action space and actor-critic and SGA are for continuous state and action space. In real environment, many of real environments are continuous one, actor-critic is much used.

On the other hand, in reinforcement learning, in order to select the appropriate action adapting to change of environment and to achieve the object, it is very important for robots to deal with corresponding to characteristic of the state taken from environment. Considering that, we propose that the self-organized fuzzy neural network (SOFNN) is taken into the reinforcement learning system. Fusing the SOFNN with actor-critic, [6] applies its fusion system to swarm behavior problem, and [8] applies to nonlinear system control problem. Fusing the SOFN with stochastic gradient ascent, [5] also applies it to nonlinear time series prediction. .

The proposed system in this paper is depicted in Fig.1. In Fig.1, SOFNN works as adaptive input state space classifier to adapt the change of environment, the present reinforcement learning part has learning ability corresponding to classified rules with Fitness Degree by SOFNN. Rules of SOFNN increase in case that the difference between the present input state and previous ones is bigger than threshold. In this study both cases, Q-learning and SGA, are considered. Simultaneously, to simulate the real environment the robot has ability to estimate own-position. Finally, it is clarified that our proposed system is effective through the autonomous robot behavior learning simulation by using the khepera robot simulator.

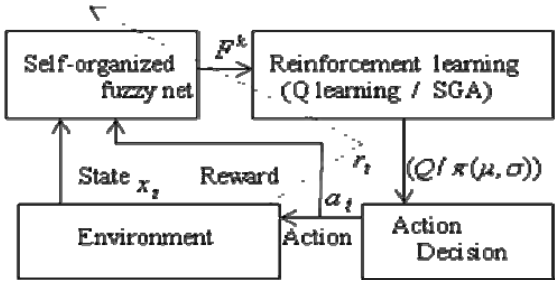

Fig.1 Structure of our proposed system

2. Self-organizing fuzzy neural network

Our self-organized fuzzy inference net is designed with a hidden layer which units are Radial Basis Function(RBF)-like fuzzy membership functions $B_i^k(x_i(t))$ to classify input states, and fuzzy rules are generated by multiplying their corresponding

membership functions as same as in [4?]. The number of membership functions and rules of fuzzy net are important for a fuzzy inference system. We proposed a self-organized fuzzy neural network (SOFNN) which constructed adaptive membership functions and rules using training data and thresholds previously [7] and [8]. Wang, Cheng, and Yi proposed a structure learning algorithm for adding and merging units using TD error distribution recently [5]. The self-multiplication algorithm to decide the size of fuzzy net in [4] is also used here.

2.1 Fuzzy net – x

The part of fuzzy net is shown in Fig. 2 in detail. For an *n*-dimension input state space $\mathbf{x}(x_1(t), x_2(t),..., x_n(t))$, a fuzzy inference net is designed with a hidden layer of fuzzy membership functions $B_i^k(x_i(t))$ to categorize the states.

$$B_{i,l}^k(x_i) = \exp\left\{-\frac{1}{2}\left(\frac{x_i - u_{i,l}^k}{v_{i,l}^k}\right)^2\right\} \qquad (1)$$

,

Here $u_{i,l}^k$, $v_{i,l}^k$ denotes the mean and the deviation of *l th* membership function, a kind of Radial basis function (RBF), corresponding to *i*th input $x_i(t)$, and connecting to *k*th fuzzy rule respectively.

Let $K(t)$ be the largest number of fuzzy rules R^k ($k = 1, 2, \ldots, K(t)$), then we have (2).

For R^k:

if($x_1(t)$ is $B_{1,L_1}^k(x_1(t))$,..., $x_n(t)$ is $B_{n,L_n}^k(x_n(t))$)

$$\text{then} \quad F^k(\mathrm{x}(t)) = \prod_{i=1}^{n} B_{i,L_i}^k(x_i(t)) \ , \qquad (2)$$

where $F^k(\mathrm{x}(t))$ means the fitness of the rule R^k for an input set $\mathbf{x}(t)$. In the case of the self-organized fuzzy net -a, in the above formulation, L_i is replaced by D_i ($i = 1, , m$) .

2.2 Adding procedure of fuzzy sets and rules

The number of membership functions and rules of fuzzy net are important for a fuzzy inference system. We proposed a self-organized fuzzy neural network (SOFNN) which constructed adaptive membership fuzzy net are important for a fuzzy inference system. We proposed a self-organized fuzzy neural network

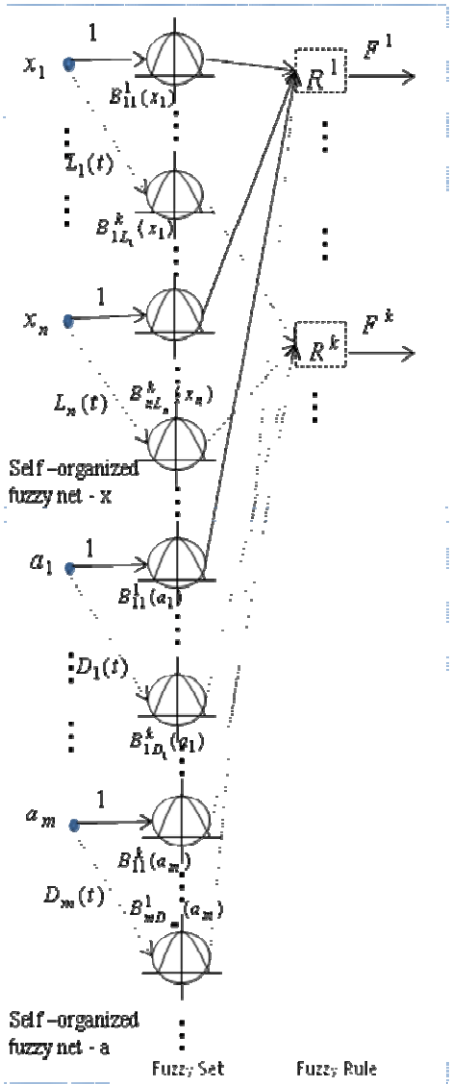

Fig.2 Structure of the self-organized fuzzy neural network (SOFNN)

(SOFNN) which constructed adaptive membership functions and rules using training data and thresholds previously [5]. Wang, Cheng, and Yi proposed a structure learning algorithm for adding and merging units using TD error distribution recently [8]. Here we use a simpler self-multiplication algorithm to decide the size of fuzzy net. Only one membership function is generated by the first input data (for example, the position of agent) of each input state vector. The value of its center equals to the value of input, 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, and the output of the rule R^1 equals to

$F^1(\mathrm{x}(1)) = \prod_{i=1}^{n} B_{i,L_i}^1(x_i(1))$ according to (2). For the next input state $\mathbf{x}(x_1(t), x_2(t),..., x_n(t))$, a new membership function is generated if (3) is satisfied.

$$\max_l B_{i,l}(x_i(t)) < \theta_{th} \tag{3}$$

Here $B_{i,l}(x_i(t))$ denotes the value of existed membership functions calculated by (1) and $l = 1,2,...,L_i(t)$ indicates the lth membership function, the maximum number is $L_i(t)$. θ_{th} 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. Thus the fuzzy net is completed to adapt to input states.

3. SGA with SOFNN

The objective of learning of the robot is to form a stochastic policy[2], that assigns probability distribution over actions to each observation to maximize the acquisition of total rewards. A policy $\pi(a, W, X)$ denotes probability density function of selecting action a in the observation X. The policy is represented by a parametric function approximator using the internal variable vector W. The robot can improve the policy π by modifying W. For example, W corresponds to connection weights where the action selecting probability is represented by neural networks. In this study, it is assumed that the action of the robot follows normal distribution, means and standard deviation of its distribution of optimal action are estimated by SOFNN with SGA (see Fig.3).

The mean μ_j and standard deviation σ_j of the normal distribution is estimated as follows:

$$\mu_j = \frac{\sum_k F^k \cdot \mu_j^k}{\sum_k F^k}, \quad \sigma_j = \frac{\sum_k F^k \cdot \sigma_j^k}{\sum_k F^k} \tag{4}$$

$(j = 1, \cdots, m)$

R^k if $(x_1 \text{ is } B_1^k, x_2 \text{ is } B_2^k, \quad , x_n \text{ is } B_n^k)$
then $\mu^k = (\mu_1^k, \mu_2^k, \quad , \mu_m^k)$
$\sigma^k = (\sigma_1^k, \sigma_2^k, \quad , \sigma_m^k)$.

3.1 Action selection

In this study, it is assumed that the action of the robot follows normal distribution of action a_j $(j = 1, \cdots, m)$,

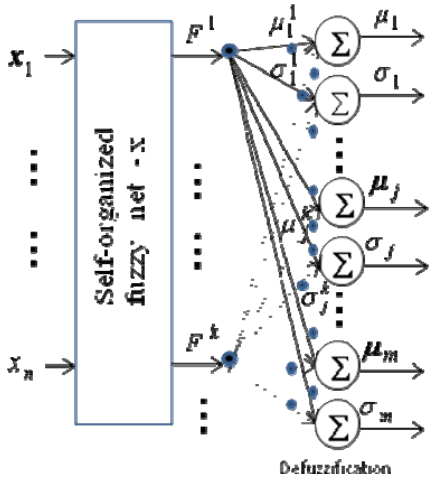

Fig.3 Structure of SGA using SOFN

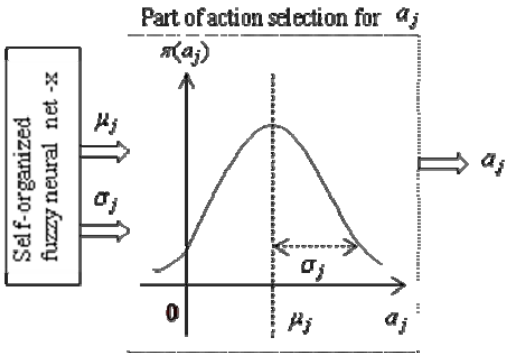

Fig.4 Action selection in SGA using SOFN

$$\pi(a_j) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left\{-\frac{(a_j - \mu_j)^2}{2\sigma_j^2}\right\} \tag{5}$$

$(j = 1, \cdots, m)$

The action of the robot can be obtained by generating a random data according this probability function (5), and is executed (see Fig.4).

3.2 General form of the SGA algorithm

A general form of the SGA algorithm is as follows:

1. Observe X_t in the environment.
2. Execute action a_t with probability $\pi(a_t, W, X_t)$, here, $W = \{u_{i,l}, v_{i,l}, \mu_j^k, \sigma_j^k\}$
   $(i = 1, \cdots, n,\ l = 1, \cdots, L_i,\ j = 1, \cdots, m, k = 1, \cdots, K(t))$.
3. Receive the immediate reward r_t
4. Calculate $e_i(t)$ and $E_i(t)$ as

$$e_i(t) = \frac{\partial}{\partial w_i} \ln\{\pi(\mathbf{a}_t, W, \mathbf{x}_t)\},$$

$$E_i(t) = e_i(t) + \gamma E_i(t-1), \quad \gamma\ (0 \le \gamma < 1),$$

where w_i denotes the i^{th} component of W.

5. Calculate $\Delta w_i(t)$ as

$$\Delta w_i(t) = (r_t - b) E_i(t) \tag{7}$$

where b denotes thereinforcement baseline.

6. Policy Improvement : update W as

$$\Delta W = (\Delta w_1(t), \Delta w_2(t), \cdots, \Delta w_i(t), \cdots),$$

$$W \leftarrow W + \alpha(1-\gamma)\Delta W, \tag{8}$$

where α .is a nonnegative learning rate factor.

7. Move to the time step t+1, and go to step 1.

4. Q-learning with SOFNN

An object of the basic Q-Leaning is discrete state and action space. However its extension to continuous state and action space has been executed so far, for example, [7]. In this study, we introduce SOFNN into Q-Leaning [7]. The proposed system is shown in Fig.5.

Outline of our Q-learning algorithm is as follows:

1. Calculate $f_k(\mathbf{x}, \mathbf{a})$ using $\mathbf{x}_t$.

$$f_k(\mathbf{x}_t, \mathbf{a}_t) = \mathbf{c}_x^k \cdot \mathbf{x}_t + \mathbf{c}_a^k \cdot \mathbf{a}_t + c_k \tag{9}$$

where,

$$\mathbf{c}_\mathbf{x}^k = [c_{x1}^k, \cdots, c_{xn}^k]^T, \mathbf{x}_t = [x_1, \cdots, x_n]^T,$$

$$\mathbf{c}_\mathbf{a}^k = [c_{a1}^k, \cdots, c_{am}^k]^T, \mathbf{a}_t = [a_1, \cdots, a_m]^T,$$

c_k for k^{th} rule,

these parameters are to be adjusted.

2. Calculate Q function :

$$Q(\mathbf{x}_t, \mathbf{a}_t) = \frac{\sum F^k \cdot f_k(\mathbf{x}_t, \mathbf{a}_t)}{\sum F^k} \tag{10}$$

3. Calculate $P(\mathbf{a} \mid \mathbf{x})$ using Eq.(11).

$$P(\mathbf{a} \mid \mathbf{x}) = \frac{\exp(Q(\mathbf{x}, \mathbf{a})/T)}{\sum_{\mathbf{b} \in action} \exp(Q(\mathbf{x}, \mathbf{b})/T)} \tag{11}$$

4. Decide action $\mathbf{a}$ as follows:

4.1 Make predefined number of segments J divided equally.

4.2 Select segment j by roulette according to Eq.(11).

4.3 Select continuous action $\mathbf{a}$ by generating a random data.

5. Receive the immediate reward r_t

6. Observe $\mathbf{x}_{t+1}$ in the environment.

7. Update $Q(\mathbf{x}_t, \mathbf{a}_t)$ using Eq.(12),

$$Q(\mathbf{x}_t, \mathbf{a}_t) \leftarrow (1-\alpha) Q(\mathbf{x}_t, \mathbf{a}_t) + \alpha \left\{ r_t + \gamma \max_i Q(\mathbf{x}_{t+1}, \mathbf{a}_i) \right\} \tag{12}$$

8. Move to the time step t+1, and go to step 1.

4.1 Learning procedure of Q-learning with SOFNN

Learning procedure is executed using commonly used steepest descent method. The criterion function E is like this,

$$E = \frac{1}{2}(\Delta Q)^2 = \frac{1}{2}\left\{ r_t + \gamma \max_b Q(x_{t+1}, b) - Q(x_t, a_t) \right\}^2 \tag{13}$$

All parameters are adjusted to minimize E. By taking gradient of E respect to each parameter, updating forms are acquired as follows:

$$\begin{cases} \mu_i^k \leftarrow \mu_i^k + \varepsilon_a \dfrac{1}{\sigma_i^k} \dfrac{x_i - \mu_i^k}{\sigma_i^k} S_k (f_k - Q) \Delta Q \\ \sigma_i^k \leftarrow \sigma_i^k + \varepsilon_\sigma \dfrac{1}{\sigma_i^k} \left(\dfrac{x_i - \mu_i^k}{\sigma_i^k} \right)^2 S_k (f_k - Q) \Delta Q \end{cases} \tag{14}$$

$$\begin{cases} C_{*i}^k \leftarrow C_{*i}^k + \varepsilon_c S_k x_i \Delta Q, \quad (*: x, a) \\ C_k \leftarrow C_k + \varepsilon_c S_k \Delta Q \end{cases}, \tag{15}$$

where $S_k = F^k / \sum_k F^k$, $\varepsilon_a, \varepsilon_c$: nonnegative learning rate factor .

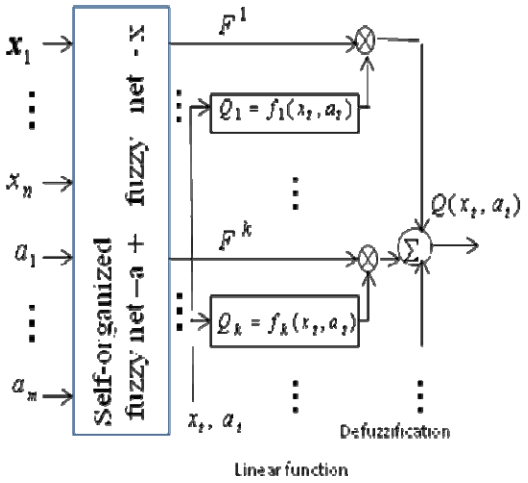

Fig.5 Structure of Q-learning using SOFN

5. Computer simulation

In this simulation, we consider the problem that the robot moves from start to goal in the 1000mm×1000mm Maze field in the Khepera

simulator shown in Fig.6. The Q-learning and SGA with SOFNN(called the proposed method) are compared with the Q-learning and SGA with fixed structure of fuzzy neural network(: FSFNN called the conventional method here).

5.1 Estimation of the position of the robot

The robot moves and estimates own position in the field. In the case that the robot takes the same action, all the position of the robot in the Khepera simulator are not always at same position, because the simulator add the ± 10 % noise to the action of the robot.

$\hat{x}, \hat{y}, \hat{\theta}$ are estimated values of x (axis), y (axis), θ (angle) from front , respectively. Here the robot has three actions and estimates own position as follows:

1. Go straight :
   Output of the right and left motor =10.0 [mm/s].
2. Right revolution :
   Output of the right and left = -4.0, 4.0 [mm/s] respectively.
3. Left revolution : reverse of the right revolution.

$\Delta\theta$ and Δdis denote the mean values of change of direction and movement distance, respectively.

1. Go straight : $E(\Delta\theta) = 0.0$[rad], $E(\Delta\text{dis}) = 5.0$[mm]
2. Right revolution :
   $E(\Delta\theta) = -0.08$[rad], $E(\Delta\text{dis}) = 0.0$[mm]
3. Left revolution : reverse of the right revolution.

The renewal form of $\hat{x}, \hat{y}, \hat{\theta}$ are as follows:

1. $\hat{x}(t+1) = \hat{x}(t) + E(\Delta\text{dis}) \times \cos(\hat{\theta})$
2. $\hat{y}(t+1) = \hat{y}(t) - E(\Delta\text{dis}) \times \cos(\hat{\theta})$
3. $\hat{\theta}(t+1) = \hat{\theta}(t) + E(\Delta\theta)$

The robot can know own position and acquire reward 0.5 by finding sub-goal, he can reset the estimation error of the position of robot there.

In the simulation, 1 action means 1 step and 10 steps continues the same action without collision with wall.1 trial means 20000 steps. The 1 step movement distance is 5[mm]. The reward used in SGA with SOFNN and FSFNN learning are 1.0 for arrival to the goal, and -0.0001 for collision with wall. On the other hands, the rewards in Q-learning with those are 100.0 for arrival to the goal, and for -0.1 for collision with wall.

The learning rate factors of mean value of post condition of the fuzzy rule are 0.05 and 0.1 for SGA and Q-learning, respectively. Other learning rate factors are 0.001.The discount factor $\gamma = 0.99$ for both learning. The temperature constant T is 0.05. The number of rules for the conventional are 1152, 9216 for Q—learning ,SGA respectively to arrange and divide a the environment equally.

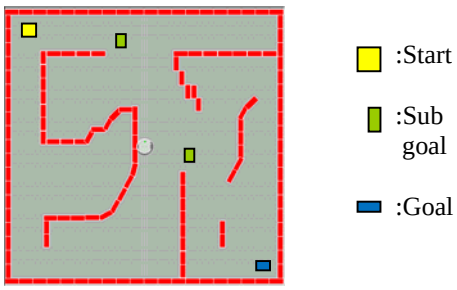

Fig.6 Maze used in the simulation

5.2 Simulation results

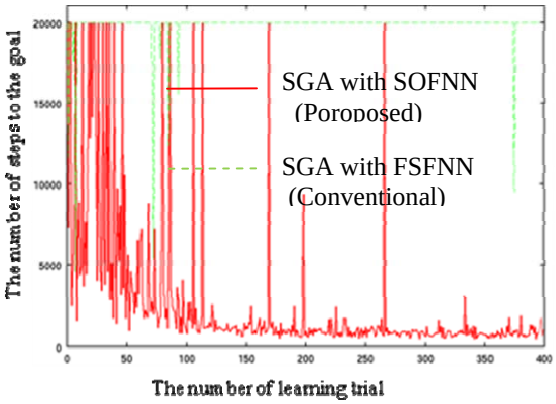

Fig.7 The number of steps to the goal for both SGA. The conventional could not arrive at the goal, however, the proposed could arrive at it.

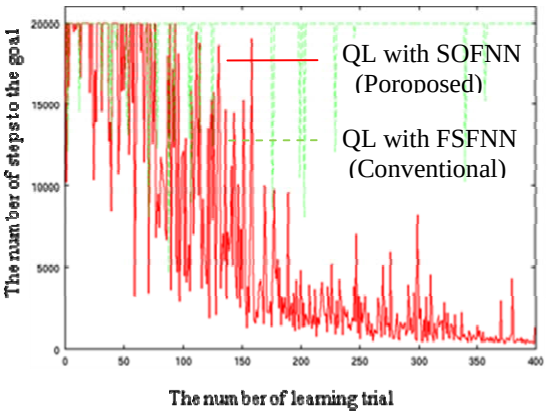

Fig.8 The number of steps to the goal for both QL. The conventional could not also arrive at the goal, however, the proposed could arrive at it.

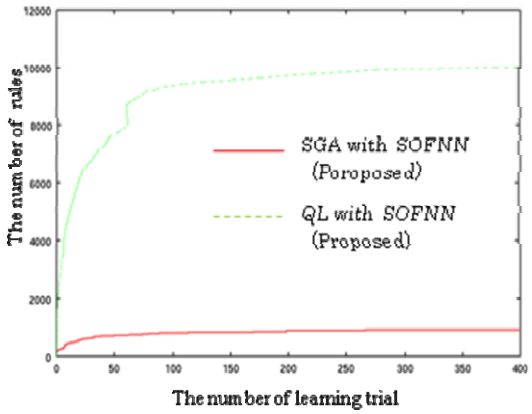

Fig.9 The number of rules are 912 and 10024 for SGA and QL, respectively after learning. That of QL is much greater than SGA.

In Fig.7-9, the position of the robot given into the system is estimated values.

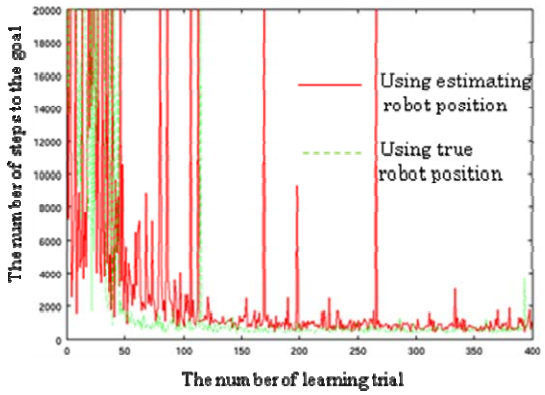

Fig.10 The number of steps to the goal for estimated and true values of the position of the robot.

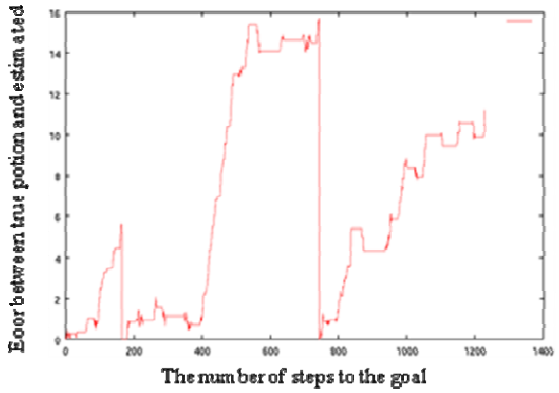

Fig.11 The change of error between true robot position and estimated one on the way to the goal at 150 trials. On the way, error is reset to 0 two times because the robot arrived at the sub-goal.

The number of average step to the goal per 10 times after learning are 515.9 and 528.9 for SGA with SOFNN and Q-learning with SOFNN, respectively. It is said that both proposed methods may have same performance.

6. CONCLUSIONS

We proposed the system that combines self-organized fuzzy-neural networks (SOFNN) with reinforcement learning system (Q-learning, stochastic gradient ascent : SGA) to realize the autonomous robot behavior learning for continuous state space, and showed its effectiveness through goal searching problem in Khepera simulator.

In our future work, we would like to try to expand the proposed system to that with memory and function of image processing in order to deal with real and plural navigator problems.

Acknowledgements

We would like to thank Mr. K. Hirosawa for his work in experiments, and a part of this study was supported by JSPS-KAKENHI (No.18500230, No.20500277 and No.20500207).

REFERENCES

[1] R.S. Sutton, A.G. Barto "Reinforcement Learning", The MIT Press,1998

[2] H. Kimura, S. Kobayashi: Reinforcement Learning for Continuous Action using Stochastic Gradient Ascent, Intelligent Autonomous Systems (IAS-5) pp.288--295 , 1998

[3] Jun Morimoto, Kenji Doya "Robust Reinforcement Learning ", *Neural Computation* ,Vol. 17, pp.335-359,2005

[4] M.Obayashi,N.Nakahara,T.Kuremoto,K.Kobayashi : " A Robust Reinforcement Learning Using Concept of Sliding Mode Control", Proc. of The 13th International Symposiumu on Artificial Life and Robotics,pp547-550,2008

[5] T. Kuremoto, M. Obayashi, K. Kobayashi: Nonlinear prediction by reinforcement learning, LNCS, Vol. 3644, pp.1085--1094 ,2005

[6] T. Kuremoto, M. Obayashi, K. Kobayashi, H. Adachi, K.Yoneda: A Reinforcement Learning System for Swarm Behaviors,Proc.2008 IEEE World Congress on Computational Intelligence (WCCI /IJCNN 2008), pp.3710—3715,2008

[7] Tadashi Horiuchi, Akinori Fujino, Osamu Katai,Tetsuo Sawaragi "Fuzzy Interpolation-Based Q-Learning with Continupus Inouts and Outputs, Trans. of the Society of Instrument and Control Engineers, Vo.35, No.2, pp.271-279,1999

[8] X. S. Wang, Y. H. Cheng and J. Q. Yi: A fuzzy Actor–Critic reinforcement learning network, Information Sci ences, 177, pp.3764--3781 ,2007