# A Swarm Learning System using Self-Organizing Fuzzy Neural Network and Reinforcement Learning

Takashi Kuremoto, Yamaguchi University, Japan, wu@yamaguchi-u.ac.jp
Masanao Obayashi, Yamaguchi University, Japan, m.obayas@yamaguchi-u.ac.jp
Kunikazu Kobayashi, Aichi Prefectural University, Japan, kobayashi@ist.aichi-pu.ac.jp
Shingo Mabu, Yamaguchi University, Japan, mabu@yamaguchi-u.ac.jp

**Abstract:** The number of state of multi-agent systems (MAS) is tremendous, so the behavior learning of agents becomes difficult. In this paper, to acquire adaptive behaviors of agents in the unknown environment, a self-organizing fuzzy neural network (SOFNN) is utilized as a state classifier. The output of SOFNN is connected to parameters of a stochastic policy of action selection. Using a reinforcement learning (RL) algorithm "stochastic gradient ascent" (SGA) proposed by Kimura et al., a swarm learning system can be composed in the case of co-operation of agents are rewarded. In the goal-navigation exploration problem, swarm learning showed is priority of learning performance comparing to individual learning which has no any reward to the cooperative behaviors of agents.

***Key-Words:*** *Self-Organizing Fuzzy Neural Network (SOFNN), Reinforcement Learning (RL), Stochastic Gradient Ascent (SGA), multi-agent system (MAS)*

## 1. Introduction

The development of intelligent systems has been giving impact to the social for decades. Autonomous robots, for example, have been entering into our daily lives since the end of last century, and artificial intelligence shows its attraction and bright future more clear recently.

Facing to the unknown or dynamical environment, an intelligent system needs to identify the situation (state), extract the knowledge hiding in its observed information, and judge to output adaptive actions to accomplish its tasks or survive in the environment. Tasks for intelligent systems, for example, hunting a prey, finding the minimum path of a maze, predicting the future value of a time series data, etc., are usually with uncertain elements. In other words, intelligence or "know-how" needs to be obtained according to the system design and its learning algorithm.

Reinforcement learning (RL), as an active unsupervised machine learning method, has been shown its usability in the fields of adaptive control, system identification, pattern recognition, time series prediction, and so on [1] [2]. To solve the goal-directed navigation problem for autonomous mobile robot, Obayashi et al. proposed a RL system [3] [4] which uses a self-organized fuzzy neural network (SOFNN) and a stochastic gradient ascent (SGA) learning algorithm given by Kimura et al. for the RL in partially observable Markov decision process (POMDP) [5]. SOFNN can also serve a state identifier for the conventional learning algorithms such as actor-critic learning, Q-learning, or Sarsa learning [6]-[9]. Moreover, the RL with SOFNN and SGA is also applied to the time series forecasting successfully [10]-[12].

In this paper, we propose to use the RL system with SOFNN and SGA to acquire cooperative actions of multiple agents. When the multiple agents learn to acquire adaptive actions, the states of the environment observed by agents

become to be uncertain for the influence of the dynamic actions of other agents. So the problem of the multi-agent system (MAS) belongs to POMDP and the learning convergence becomes difficult. Here we adopt a positive reward in to SGA for the action which leads a suitable distance between agents and a negative reward in the opposite case. These rewards make agents to acquire the cooperative adaptive actions and accelerate the learning convergence The structure of SOFNN is shown in Fig. 1.

For an *n*-dimension input state space $\mathbf{x}(x_1(t), x_2(t),\ldots,x_n(t))$, 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.

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

Here $c_i^k$, $\sigma_i^k$ 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

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

To determine the number of membership functions and rules of fuzzy net, a self-organized fuzzy neural network (SOFNN) which is constructed by adaptive membership functions and rules growing according to training data and comparing with the case of learning independently. Simulation results of goal-directed navigation problem using two agents showed the effectiveness of the RL system.

## 2. RL system with SOFNN and SGA

### 2.1 SOFNN

A self-organizing fuzzy neural network (SOFNN) is proposed by Obayashi et al. [3] [4]

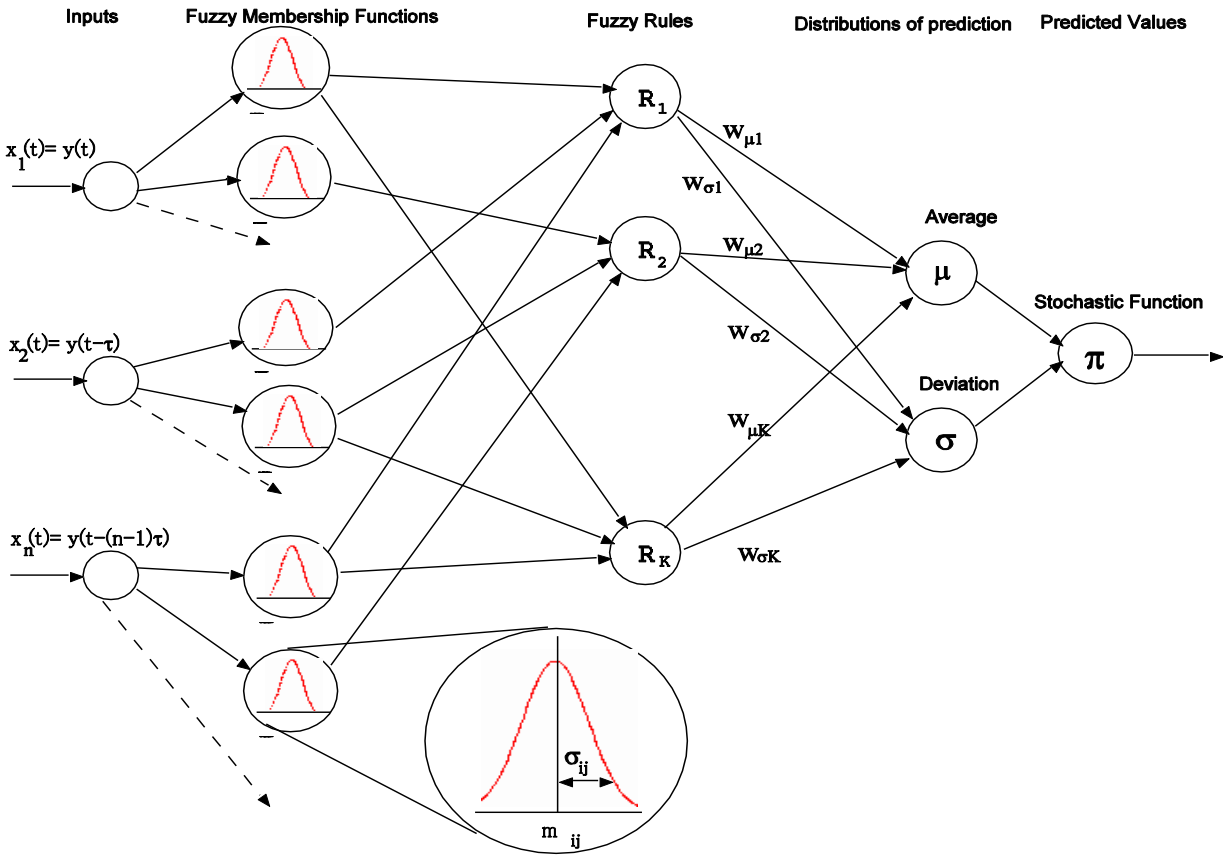



Figure 1 A self-organizing fuzzy neural network (SOFNN)

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 $c_i^k$ equals to the value of input data, and the value of width of all Gaussian function units $\sigma_i^k$ 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 $\mathbf{x}(x_1(t), x_2(t), ..., x_n(t))$, 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. (1) and $s = 1,2,...,L_i(t)$ 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 maximum value in other input dimension is chosen to be connected to the new rule. Iteratively, the fuzzy net is completed to adapt to the input data.

Eq. (2) plays a role of state classification. The output of Fuzzy rule nodes are summarized with modifiable weights as the output of the system.

### 2.2 SGA

Kimura et al.'s stochastic gradient ascent (SGA) algorithm [5] can be summarized as below:

Step 1. Observe an input $\mathbf{x}(t)$ from training data of time series.

Step 2. Predict a future data or an adaptive action $\hat{y}(t+1)$ according to a probability $\pi(\hat{y}(t+1), \mathbf{w}, \mathbf{x}(t))$.

Step 3. Receive the immediate reward $r_t$ from the environment or by calculating the prediction error.

Step 4. Improve the policy $\pi(\hat{y}(t+1), \mathbf{w}, \mathbf{x}(t))$ by renewing its internal variable $\mathbf{w}$ according to Eq. (4).

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_s \Delta \mathbf{w}(t) \tag{4}$$

Here $\Delta \mathbf{w}(t) = (\Delta w_1(t), \Delta w_2(t), \cdots, \Delta w_i(t), \cdots)$ denotes synaptic weights, and other internal variables of SOFNN, $\alpha_s$ is a positive learning rate.

$$\Delta w_i(t) = (r_t - b)\overline{D}_i(t) \tag{5}$$

$$\overline{D}_i(t) = \frac{\partial}{\partial w_i} \ln\{\pi(\hat{y}(t+1), \mathbf{w}, \mathbf{x}(t))\} + \gamma \overline{D}_i(t-1) \tag{6}$$

$\gamma (0 \le \gamma < 1)$ is a discount factor, $w_i$ denotes *i*th internal variable vector, *b* denotes the reinforcement baseline.

Step 5. For next time step *t*+1, return to step 1.

The finish condition of training iteration is usually given by the convergence of Eq. (5).

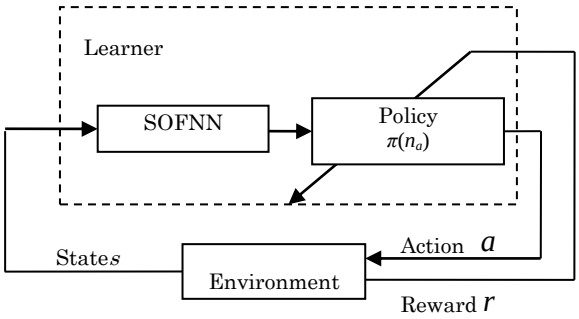



Figure 2 A RL system with SOFNN

## 3. Swarm Learning

Here, we show how the system with SOFNN and SGA is efficient to solve the problem of unknown environment exploration of multiple agents.

### 3.1 Policy of the SGA

The relationship of the RL system and its environment is shown in Figure 2. The policy of action selection is given by an asymmetric probability distribution function APDF [3] [4].

$$\pi(n_a)=\begin{cases} p\beta e^{\beta(x-\mu)} & (n_a \le \mu) \\ (1-p)\beta e^{-\beta(x-\mu)} & (n_a > \mu) \end{cases} \quad (7)$$

$$p=\frac{\sum_k w_p^k \phi_t^k(\mathbf{x}(t))}{\sum_k \phi_t^k(\mathbf{x}(t))} \quad (8)$$

$$\mu=\frac{\sum_k w_\mu^k \phi_t^k(\mathbf{x}(t))}{\sum_k \phi_t^k(\mathbf{x}(t))} \quad (9)$$

$$\beta=\frac{\sum_k w_\beta^k \phi_t^k(\mathbf{x}(t))}{\sum_k \phi_t^k(\mathbf{x}(t))} \quad (10)$$

where $n_a$ is a random number provided by probability $\pi(n_a)$, $\phi_t^k(\mathbf{x}(t))$ is given by Eq. (1) and Eq. (2), $w_p^k, w_\mu^k, w_\beta^k$ are connection weights between Fuzzy rules and parameters, Fuzzy rule $k=1,2,...K_t$.

Figure 3 shows a sample of APDF which decides the selection probability of candidate actions. When a random number z in (0.0, 1.0) is generated according to the uniform distribution, the $n_a$ is given as follows.

$$n_a=\begin{cases} \frac{1}{\beta}\ln(\frac{z}{p})+\mu & (0 \le z \le p) \\ -\frac{1}{\beta}\ln(\frac{1-z}{1-p})+\mu & (p < z \le 1.0) \end{cases} \quad (11)$$

Then, the probabilities of actions, for example, 4 actions as shown in Figure 3, can be calculated by Eq. (7)-Eq. (10). Notice the actions are defined in the different intervals in the dimension of $n_a$ ($x$ in Figure 3).

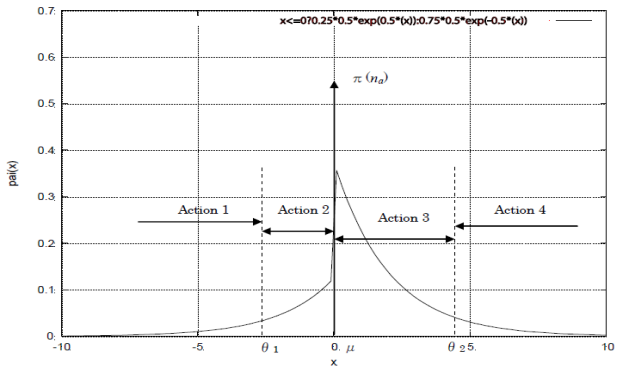



Figure 3 An asymmetric probability distribution function used as stochastic policy of action selection.

### 3.2 Experiments and Results

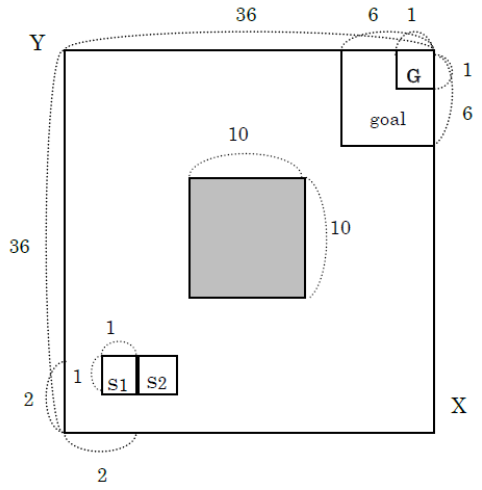



Figure 4 An environment with obstacles and goals

Table 1 Parameters of SGA with asymmetric stochastic policy used in the experiment

| Definition | Symbol | Value |
|---|---|---|
| Threshold of Action Selection | $\theta_1.\theta_2$ (Figure 3) | -1.5, 1.5 |
| Initial Value of Baseline | $b$ (Eq. (5)) | 0.6 |
| Initial Value of Traced Eligibility | $D_i$ (Eq. (6)) | 0.0 |
| Discount | $\gamma$ (Eq. (6) | 0.997 |
| Initial Value of Asymmetric Function | $\mu, \beta, p$ (Eqs. (8)-(10)) | Random number between 0 and 1, 1.0, 0.5 |
| Learning Rate | $\alpha_s$ (Eq. (4)) | 0.1, 0.001, 0.1 |

The experiment was designed as a computer simulation that two autonomous agents (robots) explored the minimum paths from their fixed start position to a goal area (Figure 4). Four direction actions, up, down, left, right can be observed by the agent. So the input to SOFNN had 4 dimensions, each with 0 (aisle) or 1 (occupied) values.

Goal area with a high reward 100.0, and crash to obstacle, other agents, wall with a negative reward -1.0. When a suitable Euclidean distance between 2 agents was considered

(from 1.0 to 3.5), a positive reward +1, otherwise -1 were added into $r_t$ in Eq. (5).

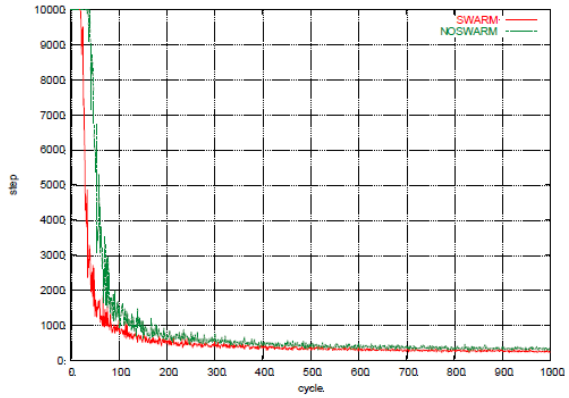
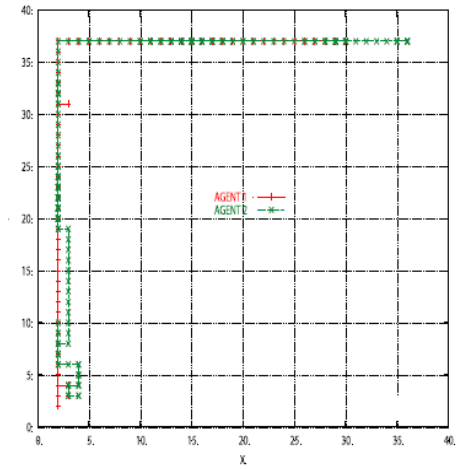
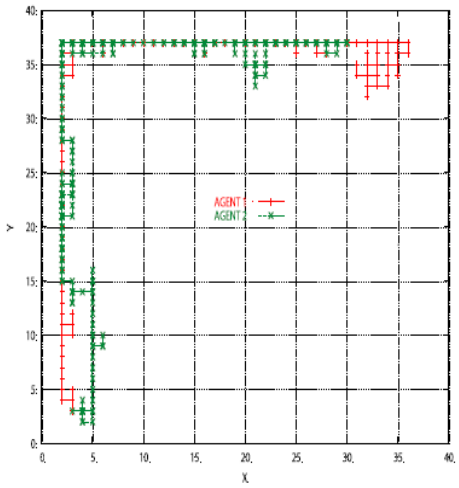



Figure 5 Comparison of learning performances between swarm learning and individual learning




(a) Swarm learning result




(b) Individual learning result

Figure 6 Trajectories of 2 agents after learning

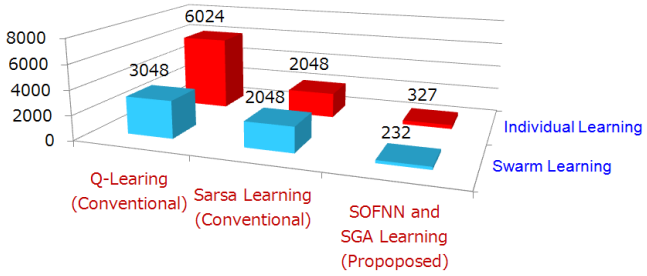



Figure 7 The comparison of learning costs (average steps of one trial)

The learning method with distance constraint is called "swarm learning", and "individual learning" means without this constraint.

Other parameters used in the experiment are shown in Table 1.

The learning performances of different learning methods are shown in Figure 5. Swarm learning showed its priority to the individual learning with faster convergence. The trajectories of learning results are shown in Figure 6. All agents found the goal, avoided the obstacle in the center of the environment, however, swarm learning with shorter steps (232) than individual learning results (327).

Comparison experiments were also performed using the conventional Q-learning and Sarsa learning [1]. In the conventional method, SOFNN was not used, and the policy of action selection used the soft-max method [1]. The learning performance, i.e., the average length of paths during the training processes, is compared in Figure 7. The proposed method, i.e., SOFNN and SGA with Swarm Learning, showed the lowest learning computational cost among the methods.

## 4. Conclusion

In this paper, a RL system with SOFNN and SGA was introduced and applied to swarm learning of multiple autonomous agents firstly. Simulations of unknown environment exploration using the proposed method were performed and results showed the effectiveness of the method. Though the simulation used a simple input vector, it suggests

that complicated information including images, high dimensional input, etc. is also available to be dealt by the proposed system.

## Acknowledgement

A part of this work was supported by Grant-in-Aid for Scientific Research of JSPS (No. 25330287 and No. 26330254)

## References:

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

[2] L. P. Kaelbling, M. L.Littman: Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, Vol. 4, 1996, pp.237-285

[3] M. Obayashi, A. Iseki, and K. Umesako: Self-organized reinforcement learning using fuzzy inference for stochastic gradient ascent method, Proceedings of the International Conference on Control, Automation and Systems (ICCAS2011), pp.735-738, 2001

[4] M. Obayashi, T. Kuremoto, and K. Kobayashi: A self-organized fuzzy-neuro reinforcement leaning system for continuous state space for autonomous robots, Proceedings of International Conference on Computational Intelligence for Modeling, Control, and Automation (CIMCA'08), pp. 552-559, 2008

[5] H. Kimura, M. Yamamura, S. Kobayashi: Reinforcement learning in partially observable Markov decision process: a stochastic gradient method. Japanese Society for Artificial Intelligence, Vol.11, No. 5, 1996, pp. 85-92 (in Japanese)

[6] T. Kuremot, 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, 2009, pp.724-744

[7] 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, 2009, pp. 1517-1531

[8] T. Kuremoto, Y. Yamano, L.-B. Feng, K. Kobayashi, and M. Obayashi: A fuzzy neural network with reinforcement learning algorithm for swarm learning, Lecture Notes in Electronic Engineering (LNEE), Vol.144, 2011, pp.101-108

[9] T. Kuremoto, M. Obayashi, K. Kobayashi: Neuro-Fuzzy Systems for Autonomous Mobile Robots. In Horizons in Computer Science Research, Vol. 8, (ed. Thomas S. Clary), pp. 67-90, Nova Scientific Publishers , 2013

[10] T. Kuremoto, M. Obayashi, A. Yamamoto, and K. Kobayashi: Neural Prediction of Chaotic Time Series Using Stochastic Gradient Ascent Algorithm. In Proceedings of the 35th ISCIE International Symposium on Stochastic Systems Theory and Its Applications (SSS'03 ), pp. 17-22, 2003

[11] T. Kuremoto, M. Obayashi, and K. Kobayashi: Forecasting Time Series by SOFNN with Reinforcement Learning. in Proceedings of the 27th Annual International Symposium on Forecasting (ISF 2007), pp.99, 2007

[12] T. Kuremoto, M. Obayashi, and K. Kobayashi: Neural Forecasting Systems. In Reinforcement Learning, Theory and Applications (Eds.: C. Weber, M. Elshaw, and N. M. Mayer), pp.1-20, In-Tech , (Online Open Access) 2008