Automatic Learning Based on SOM and Reinforcement Learning

Ryo Maeda and Masafumi Hagiwara
Department of Information and Computer Science, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama 223-8522, Japan
{maeda,hagiwara}@soft.ics.keio.ac.jp

Abstract − *In this paper, we propose a system having automatic learning ability. It can acquire skills or knowledge by repeating trials in an unknown situation. It is assumed that the proposed system does not have special knowledge or mechanism on the target: what the system tries to do is to continue the target as long as possible. A lot of studies have been made on autonomous learning, however, they are toward specific problems: learning is limited on the target problem. This research aims at obtaining universal learning mechanism which does not depend on a specific problem. The proposed system employs reinforcement learning self-organized feature map to cope with huge state. "Tetris" and "Puyo-puyo" are used as learning target problems. Experimental results show the effectiveness of the proposed system.*

1 Introduction

In the real world, the situation which cannot be predicted exists. The traditional computers are able to perform only specified operation. Therefore, it is difficult to cope with unexpected events [1].

Most of the studies on artificial intelligence employ top-down method [2]. However this approach requires huge amount of if-then rules. For example, in 1997, the super computer "Deep Blue" of IBM won the chess champion. The algorithm of this computer is a total search based on simple logic [3]. It requires a lot of cost to constitute this system. The computer should respond more flexibly by the autonomous study.

Unsupervised learning is often used for game learning. These methods adopt important knowledge and information into the system, and can achieve effective learning by adjusting parameters [4]. However, it is necessary to adjust an evaluation function appropriately for each target. Finally this kind of method is reduced to optimization of parameters. And the obtained rules are based on these parameters.

Supervised learning requires the explicit provision of input-output pairs. In game learning, this input might be a play log of users. However, in supervised learning, teacher data with targets are indispensable, and the performance depends on teacher data [5].

In contrast, reinforcement learning(RL) uses a scalar reward signal to evaluate an input-output pair, and discovers the optimal output for each input through trial and error, even if RL has neither prior knowledge nor teacher data [6]- [7].

RL is an efficient method without priori knowledge about the environment. However RL needs numerous trials to learn. Since RL must cover all of the state-action pairs of the target, if the target is complicated, learning will become difficult.

In this paper, we propose a new learning system adapted for environment by repeating trials. What the proposed system tries to do is to continue the target as long as possible.

In order to cope with huge state, we combined self-organizing feature map(SOM) and RL. SOM can do clustering without teacher signal [8]. The number of states can be reduced by applying RL only to a representation pattern of clustering.

The proposed system treats two or more sensors. And the system presumes the value of each sensor information, when each information is mapped on SOM and is learned by RL.

We select "Tetris" and "Puyo-puyo" that are popular games as targets. Our system can acquire the skill that operate the target object adequately. "Tetris" and "Puyo-puyo" are the similar environment, however, they are different considerably. We evaluate and discuss the auto leaning skill in strange environment.

2 System

2.1 Outline of the proposed system

Outline of the proposed system is shown in Fig.1. The system learns to operate target object appropriately in unknown environment. Our system acquires state of the environment by observing environment. Basically the proposed system tries to know how to operate agents

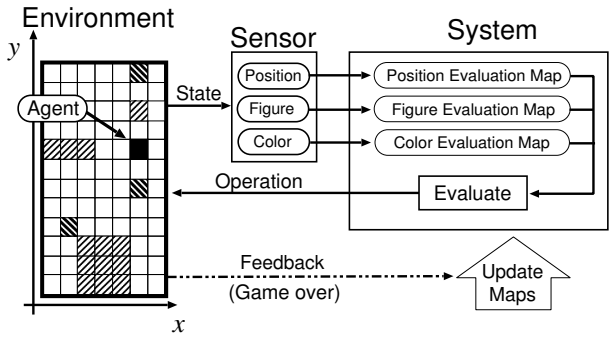

Figure 1: Outline of the system

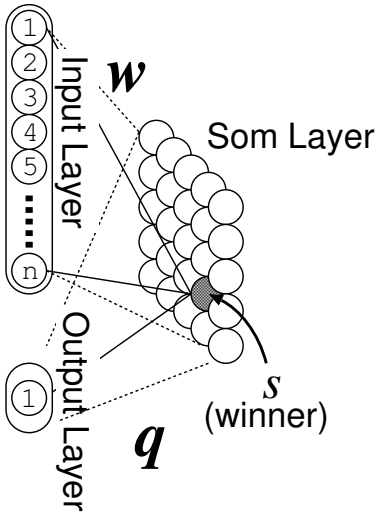

Figure 2: Outline of evaluation map

beforehand. The system consists of two or more sensors and evaluation maps corresponding to each sensor. A sensor observes an state of environment. Without sensors, the system can't get the state of environment. The same number of evaluation maps exist as a sensor. And evaluation maps are applied the structure of SOM. In this paper, we use three kinds of sensors that deal with color, position, and shape.

The system observes the state for trial and the state of agents, and operates agents appropriately based on the information on the environment. The evaluation maps acquire from the environment the information corresponding to each map, and estimate the evaluation value for operating agents. The evaluation value is calculated at every map in all cases to which agent should act next. An action having the highest comprehensive evaluation is selected as next action. Comprehensive evaluation is combined evaluation of each map.

The proposed system learns by feedback information. The evaluation is updated with reinforcement learning. However, if RL is applied to all patterns, since the number of states becomes huge, learning may not be performed appropriately. Therefore, all of the patterns are clustered using SOM. Similar inputs are arranged on a map in near position. The number of states can be reduced by applying RL to the representation pattern of each cluster.

Outline of evaluation map is shown in Fig.2. Sensor information is input to evaluation maps, and evaluation value for input pattern is output. Maps are composed of input layer, SOM layer and output layer. Input layer and SOM layer are fully connected by a weight vector $\boldsymbol{w}$. The weight vector is updated using by competitive learning. SOM layer and output layer are connected one by one by a evaluation value vector $\boldsymbol{q}$. The evaluation value vector is updated using reinforcement learning.

The number of neurons in the input layer is equal to the number of dimensions of a sensor input. Weight vector between the input layer and the SOM layer is first initialized by small uniform random numbers.

Mapping by classifying an input pattern roughly, the map can hold an input pattern in the form of a weight vector. The number of nuerons in the output layer is one. Weight between SOM layer and output layer is regarded as evaluation value of each neuron on the SOM layer.

When evaluation map receives an input, the neuron that has weight vector close to the input will be fired. The fired neuron outputs its evaluation value. And evaluation value is updated by reinforcement learning.

Weight combined between neuron i in the SOM layer and neuron j in the input layer is $w_j^{(i)}$.

Weight vector $\boldsymbol{w}^{(i)}$ is

$$\boldsymbol{w}^{(i)} = \left(w_1^{(i)}, w_2^{(i)}, \cdots, w_n^{(i)}\right) \tag{1}$$

In a similar way, weight in SOM layer and output layer is $q^{(i)}$. Vector of evaluation value is

$$\boldsymbol{q}^{(i)} = \left(q^{(i)}\right) \tag{2}$$

If an input vector $\boldsymbol{U}$ is given, the euclidian distance of an input vector and a weight vector will be computed first. The neuron having the weight vector close to the input vector is searched, and the neuron is fired as a winner neuron. And the evaluation value vector which winner neuron holds is outputted.

$$s = \arg\min_i \left\{|\boldsymbol{U} - \boldsymbol{w}^{(i)}|\right\} \tag{3}$$

$$\boldsymbol{Q} = \boldsymbol{q}^{(s)} \tag{4}$$

2.1.1 Sensor

The proposed system use position sensor, shape sensor, and color sensor in order to acquire state of environment.

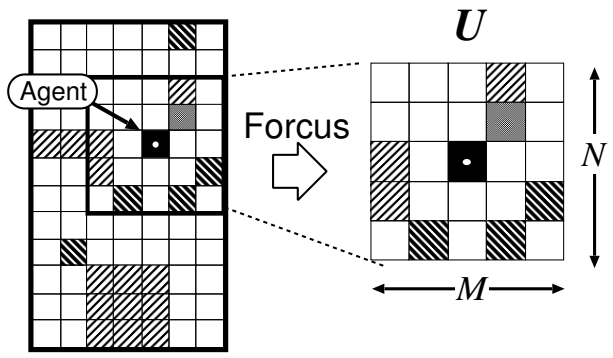

Figure 3: Vicinity pattern

Like human being observes focusing on an object, the observable range of a sensor is assumed within the $M \times N$ around the center position of agent [9]

We call the state in sensor domain as a circumference pattern.

2.1.2 Position evaluation map

The position of an agent when the agent is operating appropriately is inputted to the position evaluation map. Position evaluation map outputs the evaluation value that is an indicator of position. In this reason, as human pays attention to an object, the proposed system captures rough position of agent to environment.

The input vector is expressed as

$$\boldsymbol{V} = (x, y) \tag{5}$$

.

2.1.3 Shape evaluation map

Input of shape evaluation map checks whether a cell $u_{i,j}$ exists in the area $M \times N$ of circumference pattern. Circumference pattern is the existence of the cell in the Board. If there is a cell, circumference pattern is regarded as 1, and without cell, pattern is regarded as 0. Average of nine neighboring cells is added to each cell.

Therefore, input vector is expressed as

$$\boldsymbol{V} = (v_{0,0}, v_{1,0}, \cdots, v_{M,0}, v_{0,1}, \cdots, v_{M-1,N}, v_{M,N}) \tag{6}$$

$$v_{i,j} = u_{i,j} + \sum_{n=j-1}^{j+1} \sum_{m=i-1}^{i+1} (u_{m,n})/9 \tag{7}$$

2.1.4 Color evaluation map

Color evaluation map receives the number of each color in circumference pattern. Since the system used only the information on the level of a color like human.

When there are C_c kinds of color already observed , the dimension of input layer is C_c. neuron i of input layer is inputted number v_i of the color in circumference pattern. For the neuron i in the input layer, v_i is the number of color i pixels which exists in the circumference pattern.

$$\boldsymbol{V} = (v_1, v_2, \cdots v_{C_c}) \tag{8}$$

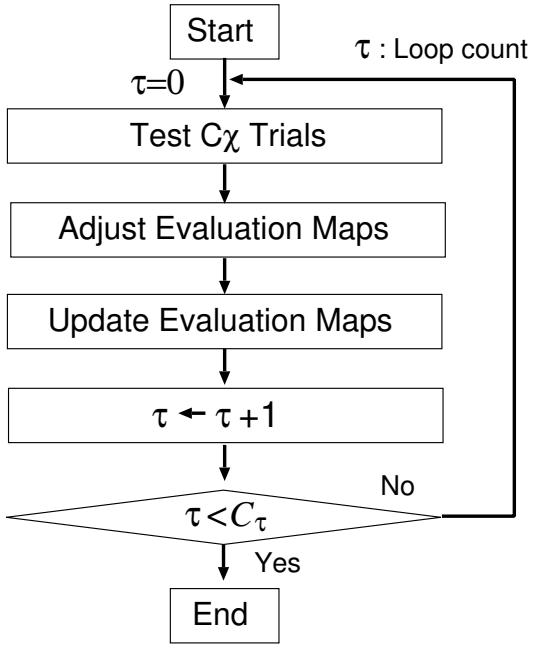

Figure 4: Outline of learning

2.2 Algorithm to operate Agent

1. Evaluate all actions $\boldsymbol{a}$.
2. Enter sensor information to maps, and output evaluation value $\boldsymbol{Q}_{map}(a)$ for each action $\boldsymbol{a}$.
3. Adopt act $\boldsymbol{a}$ that maximize equation $P(a)$.

$$P(a) = \prod_{map} (\pi(map) Q_{map}(a)) \times \xi(t) \quad \text{for all } maps \tag{9}$$

Where, $\xi(t)$ is a uniform random number. C_τ is the number of times of learning. This is because RL learns by mistake and error.

2.3 Learning

The outline of learning is shown in Fig.4.

Whenever the system operates agents, the system renews evaluation maps once. Until updating map, for all actions during trying target, the learning continues.

1. Based on trial, the proposed system inputs sensor information to each evaluation maps.
2. Fired neuron for the input is selected.
3. The weight vector held each neuron is adjusted.

2.4 Update code vector

By using the input data, system learns weight vector of all neurons in accordance with SOM rule. Weight vector that is connected to neighborhood neuron of the fired neuron is updated. The updating formula of the weight vector $\boldsymbol{w}^{(i)}$ of the neuron i is as follows.

$$\boldsymbol{w}^{(i)} \leftarrow \boldsymbol{w}^{(i)} + \lambda(t)\phi(d_i,t)(\boldsymbol{V}-\boldsymbol{w}^{(i)}) \quad (10)$$

$$\phi(d_i,t) = \exp(-\frac{d_i}{2(\sigma(t))^2}) \quad (11)$$

$$\lambda(t) = \lambda_I \times (\frac{\lambda_F}{\lambda_I})^{\frac{t}{T}} \quad (12)$$

$$\sigma(t) = \frac{R}{2} \times (\frac{2}{R})^{\frac{t}{T}} \quad (13)$$

$\lambda(t)$ is learning factor. $\phi(d_i,t)$ is neighborhood function. d_i is distance between the neuron i and fired neuron. And R is initial size of neighborhood. λ_I is initial value of λ. λ_F is final value of λ, and T is the number of trials in other words times of learning.

Until learning times reaches a certain number, the proposed system continues the trials. Then each neuron on the SOM layer acquires the state of environment as the form of weight vector.

2.5 Reinforcement learning

Evaluation value vector is initialized by small uniform random number. When one trial is finished, the proposed system receives punishment ρ. An evaluation value vector of the neuron near a winner neuron is updated largely. The updating formula of an evaluation value vector is similar to the study rule of RL.

$$\boldsymbol{q^{(i)}} \leftarrow \boldsymbol{q^{(i)}} + \phi(d_i,t)\left[\rho + \gamma(\boldsymbol{q^{(s)}} - \boldsymbol{q^{(i)}})\right] \quad (14)$$

Where,γ is discounted rate, and ρ is the feedback from environment. In this paper, only when a trial is finished, punishment is returned from environment.

By SOM, similar input data is mapped near by on SOM layer. The neighborhood neuron of the fired neuron is updated as soon as the fired neuron is updated. It might be contribute to acceleration of convergence.

3 Experiments

We demonstrates the performance of the proposed system on two environments. These environments are "Tetris" and "Puyo-puyo".

Evaluation of experiment uses the score of the trial and length of play time of the trial. We think that the achievement of learning relates to play time closely on "Tetris" and on "Puyo-puyo". For example, play time is constant on "Othello". But trial play time depend on player's skill on "Tetris" and "Puyo-puyo". Consequently, we evaluate not only the score but also play time. Play time defines the time interval at which one agent operates once as unit time.

Table 1: Evaluation experiment("Tetris")

	score(deleted lines)		play time(time)	
	maximum	average	maximum	average
proposed method	37	9.421	128	61.3
random method	2	0.013	34	20.2

Table 2: Evaluation experiment("Puyo-puyo")

	score(deleted times)		play time(time)	
	maximum	average	maximum	average
proposed method	68	14.7	227	108.3
random method	33	6.3	136	65.7

We compared the proposed method with the random method. After performing renewal of a map C_τ times, 10,000 times performance evaluation was carried out. And maximum value and average value were calculated.

3.1 Evaluation experiment("Tetris")

First, we evaluated the proposed system on "Tetris". In what follows, we explain "Tetris". The falling brick is used and it can move on the playing field. While it falls, it can rotate there. This trial is finished when block is heaped up the highest. If you form a horizontal row of square blocks, the row is deleted and the blocks which remained move down. Score of the trial is the number of deleted horizontal row.

The result of simulation is shown in Table.1.

In the first learning, only one line was not able to be eliminated for the first several 100 trials. The system began to be able to delete lines by degrees through learning as shown in Fig.5. More blocks could be deleted gradually as learning progressed.

3.2 Evaluation experiment("Puyo-puyo")

Second, we evaluated the proposed system on "Puyo-puyo". In what follows, we explain "Puyo-puyo". If a match is any chain of 4 or more blocks of the same color, that are adjacent to each other (up, down, right, or left), blocks disappear and all blocks above move down. Then, Score of the trial is the number of deleted times.

The result of simulation is shown in Table.2. And the result also indicates that performance is enhanced. As

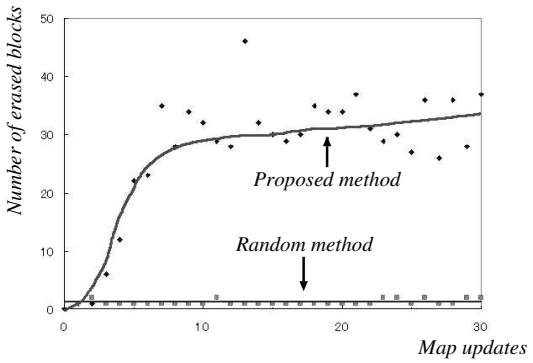

Figure 5: Maximum deleted lines on "Tetris"

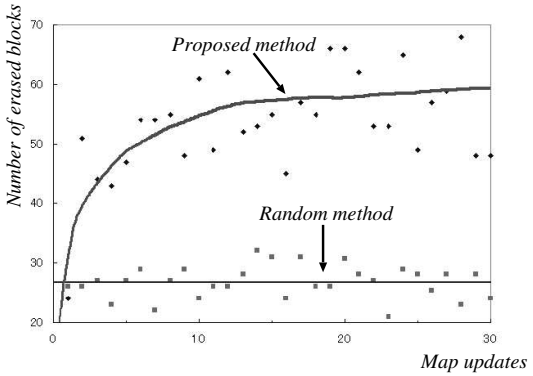

Figure 6: Maximum deleted lines on "Puyo-puyo"

shown in Fig.6.

4 Conclusion

This paper is concerned with automatic learning in unknown environment. The system learns through trial and error, and doesn't have valuable knowledge and mechanism about target. It tries to continue the target as long as possible. This research aims at obtaining universal learning mechanism which does not depend on a specific problem. We combined self organizing map and reinforcement learning to cope with huge state, and used "Tetris" and "Puyo-puyo" as learning target problems. We confirmed that the proposed system got better performance by degrees, and could learn without information specialized by the target. Experimental results show that the proposed system has useful learning mechanism for unknown environment.

Acknowledgment

This research is partly supported by Keio University Special Grant-in-Aid for Innovative Collaborative Research Projects.

References

[1] Tomohiro Yamaguti: "Propagating Learning Behaviors from a Virtual Agent to a Physical Robot in Exploitation-Oriented Reinforcement Learning" Journal of Japanese Society for Artificial Intelligence, Vol.12, No.4, pp.570-581, 1997. (in japanese)

[2] Hitoshi Matsubara: "Recent Progresses on Game Programming Reserches," Journal of Japanese Society for Artificial Intelligence, Vol.10, No.6, pp.835-845, Nov. 1995. (in japanese)

[3] Hitoshi Matsubara: "What Can AI Resercheres Learn from Deep Blue's Victory?" Journal of Japanese Society for Artificial Intelligence, Vol.12, No.5, pp.698-703, 1997. (in japanese)

[4] Singer J. A.: "Co-evolving a Neural-Net Evaluation Function for Othello by Combining Genetic Algorithms and Reinforment Learning," Lecture Notes in Computer Science, Vol.2074, pp.377-389, 2001.

[5] Mirai Tabuse, Masafumi Hagiwara: "Automatic Learning for Acquisition of Tetris's Skills using Fuzzy Inference Neural Networks," Journal of Japan Society for Fuzzy Theory and Systems, Vol.11, No6, pp.1089-1097, Dec. 1999. (in japanese)

[6] Sadayoshi Mikami "Reinforcement Learning," Morikita publisher, 2000. (in japanese)

[7] Andrew James Smith: "Application of the self-organizing map to reinforcement learning," Neural Networks 15, pp.1107-1124, 2002.

[8] Teuvo Kohonen: "The Self-Organizing Map," Proceeding of IEEE, Vol.78, No.9, pp.1464-1480, Sep. 1990.

[9] Yuuichiro Anzai "Recongition and Learning," Iwanami bookstore, 1989. (in japanese)