Intelligent Agent Construction Using the Attentive Characteristic Patterns of Chaotic Neural Networks

Masanao Obayashi*, Liang-Bing Feng*, Takashi Kuremoto*, and Kunikazu Kobayashi*

*Graduate School of Science and Engineering, Yamaguchi University

2-16-2 Tokiwadai, Ube, Yamaguchi, 755-8611, Japan

(Tel : +81-836-85-9518, Fax : +81-836-85-9501)

E-mail : {m.obayas, n007we, wu, koba}@yamaguchi-u.ac.jp

Abstract - Human learns incidents by own actions and reflects them on the subsequent action as own experiences. These experiences are memorized in his brain and recollected if necessary. This research incorporates such an intelligent information processing mechanism, and applies it to an autonomous agent. In the proposed system, reinforcement Q-learning method is used for learning. Auto-associative chaotic neural network is also used as mutual associative memory system. However agent cannot retrieve some stored patterns exactly, in the case of too many stored patterns and strong correlation among them. To solve this problem, we propose to use kinds of attentive parameters and attentive characteristic patterns. The attentive characteristic pattern is a part of the stored patterns. When robots concentrate their attention on the specific part of the stored pattern, i.e., the attentive characteristic pattern, whole stored patterns are retrieved easily and completely. Finally, the effectiveness of this proposed method is verified through the simulation applied to the plural maze-searching problems.

Key Words - Intelligent, Attentive parameter, Chaotic neural network, Mutual associative memory system

1. INTRODUCTION

Recently, use of robots has been considered in various fields. Such robots which behave near to humans, namely intelligent robots have been required and have attracted attention of many researchers. As a part of things which are required for the robots, experience, learning, memory and their use etc. need to be considered.

In our previous work we proposed a method [1] to realize these functions of the intelligent robots mentioned above, that is, by reinforcement learning algorithm for learning function [2] and chaotic neural networks for memorization and association [3][4]. It is important that robots can memorize their experiences and remember them quickly and exactly and behave making use of them. However in our work there is a problem, i.e., agent could not retrieve some stored patterns exactly, because of too many stored patterns and strong correlation among them.

As a way to solve this problem, in this article we propose to use kinds of attentive parameters and attentive characteristic patterns. The attentive characteristic pattern is a part of the stored patterns. When robots concentrate their attention on the specific part of the stored pattern, i.e., the attentive characteristic pattern, whole stored patterns are retrieved easily and completely. The attentive parameter is a scalar value and corresponds to its attentive characteristic pattern and is defined for every stored pattern.

To be specific, the agent calculates the degree of similarity, i.e., the value of attentive parameter, between a specific attentive characteristic pattern and each retrieval pattern in the chaotic neural network. After that the agent changes the maximum attentive parameter value among them to 1.0, meanwhile, others to 0.0. Therefore the chaotic neural network can represent the whole pattern including the specific attentive characteristic pattern quickly and exactly.

As another issue, there is a so-called aliasing problem that agents regard the two different overall states as the same because of partially observable ability of sensors of agents. One way to solve this problem is to make use of the past history of the agent's behaviors. However it is difficult for the agent to retain its whole past history because of the memory capacity. Therefore we propose to retain the parity information of them less required capacity than whole information of history. Finally we show that our proposed methods are effective through computer simulations of the optimal path searching problem in mazes.

2. ASSOCIATIVE CHAOTIC NEURAL NETWORK

Chaotic Neural Network (CNN) is constructed with chaotic neuron models that have refractory and continuous output value. Its useful usage is as associative memory network named ACNN [3][4]. Here are the dynamics of ACNN.

$$s_i(t+1) = f(y_i(t+1) + z_i(t+1)) \qquad (1)$$

$$y_i(t+1) = k_r \cdot y_i(t) - \alpha \cdot s_i(t) + a_i \qquad (2)$$

$$z_i(t+1) = k_f \cdot z_i(t) + \sum_{j=1}^{n} \omega_{ij} s_j(t) \qquad (3)$$

where $s_i(t)$: output value of ith neuron at time t, n: numbers of input, ω_{ij}: synapse from ith neuron to jth neuron, $y_i(t)$: internal state of ith neuron at time t, $z_i(t)$: internal state of ith neuron as to reciprocal action, a_i : threshold of ith neuron, k_r, k_f: damping coefficient, α : constant.

An input-output function $f(\cdot)$ to be used with CNN is described as follows,

$$f(t) = \frac{1 - \exp(-t/\varepsilon)}{1 + \exp(-t/\varepsilon)}, \qquad (4)$$

where ε is a positive constant. Threshold for binary representation is set to 0.5 for all neurons.

For example, in the case of ACNN, patterns stored into the ACNN are set as follows,

$$\omega_{ij} = \frac{1}{P}\sum_{p=1}^{P}(2x_i^p - 1)(2x_j^p - 1), \qquad (5)$$

x_i^p : *ith* element of *pth* stored pattern (0 or 1), P : a number of stored patterns.

Fig. 2 shows the example of behaviors of the chaotic neural network consists of $n = 100$ neurons (10 times 10) with 4 memory patterns and the network as shown in Fig. 1. We find that the stored patterns in the CNN are appearing through patterns of the output chaotically. Table 2 shows the parameters used in this CNN.

3. MUTUAL ASSOCIATIVE CHAOTIC NEURAL NETWORK

In this section, we consider the CNN as memory system of the agent, i.e. we explain the way to use the auto-associative CNN as a mutual-associative CNN (MACNN). The structure of both CNNs is same except the meaning of the synaptic weights. In the MCNN, it consists of environmental inputs (I) and their corresponding actions (O) and other neurons with random values to weaken the correlation among each stored patterns (M) (see Fig. 3). After learning of the optimal actions and input-output patterns are stored into the MACNN, when the agent moves in the environment, agent gets the information from environments and set the input information (I) to

MACNN as the initial state, the action corresponding to input information (O) are retrieved. The agent executes the action. However, as the stored patterns increase, it becomes difficult to retrieve the desired patterns because of their strong correlations. Therefore, to solve this problem, we propose the method with using the attentive parameters.

$$W\{=\omega_{ij}\} = \begin{bmatrix} I^1 \\ M^1 \\ O^1 \end{bmatrix} \begin{bmatrix} I^{1T} & M^{1T} & O^{1T} \end{bmatrix} + \cdots + \begin{bmatrix} I^P \\ M^P \\ O^P \end{bmatrix} \begin{bmatrix} I^{PT} & M^{PT} & O^{PT} \end{bmatrix} \qquad (6)$$

4. ATTENTIVE PARAMETERS

4.1 Attentive parameter

The synaptic weights are set as follows,

$$\omega_{ij} = \frac{1}{P} \sum_{p=1}^{P} \lambda_p(t) \cdot (2x_i^p - 1)(2x_j^p - 1), \qquad (7)$$

where $\lambda_p(t)$: attentive parameter of $p\,th$ stored pattern $x_i^p (= \begin{bmatrix} I^{pT} \; M^{pT} \; O^{pT} \end{bmatrix}^T)$ at time t. The $\lambda_p(t)$ is restricted as follows,

$$\begin{cases} 0 \le \lambda_p(t) \le 1 \\ \sum_{p=1}^{P} \lambda_p(t) = 1 \end{cases} \qquad (8)$$

4.2 Calculation of values of attentive parameters

In MACNN, the part of the stored pattern is set as the characteristic. An example of two stored patterns and characteristic 20 bits during operation at time t are shown in Fig. 4(a)(b), respectively. The values of attentive parameters of the stored pattern $\lambda_p(t)\ (p=1,2)$ are as follows,

$$\begin{aligned}\lambda_1(t) &= \frac{9/20}{9/20+17/20} = 9/26 \\ \lambda_2(t) &= \frac{17/20}{9/20+17/20} = 17/26\end{aligned} \qquad (9)$$

Thus, attentive parameter gives the similarity of the output pattern for the $p\,th$ stored patterns.

4.3 Operation of the MACNN

The attentive parameter is used as follows.

Step1: The values of each attentive parameter of stored patterns are calculated like Eq. (9).

Step2: The number of stored pattern which has max value of attentive parameter is decided by

$$k = \arg\max_i \lambda_i(t).$$

Step3: If $\lambda_k(t) > \theta_{th}$, then

$$\lambda_k(t) = 1.0, \lambda_i(t) = 0.0\ (i \neq k)$$

else nothing is done.

Step4: Go to the next MACNN operation.

5. STORED PATTERNS WITH PARITY BIT

The agent is often faced with aliasing problem, i.e. there is often that agent is mistaken certain different environments as same environment because of the lack of its sensor ability. In this article, we solve this problem by making use of past information, i.e. adding parity bits to the stored patterns of state-action pairs. This is the less memory required method than storing of all information used before.

6. REINFORCEMENT LEARNING

In this article, agent uses the Q-learning method to learn the optimal action, as follows,

$$Q(s,a) = Q(s,a) + \alpha\{r + \gamma \max_{a'} Q(s',a') - Q(s,a)\}, \quad (10)$$

where s : state, a : action, s' : next state, a' : next action, r : reward. α, γ : positive constant.

The agent action is selected using the next probability,

$$B(a \mid s) = \frac{\exp(Q(s,a)/T)}{\sum_{b \in A} \exp(Q(s,b)/T)}, \quad (11)$$

where $B(a \mid s)$ is probability of action a under the state s , T is positive constant called temperature parameter.

7. ALGORITYM

Our proposed method is organized as follows,

1) Learning of the optimal action using the Q-leaning with the parity bits in section 5 and 6.
2) Constructing the synaptic weights with attentive parameters (Eq. (6)-(7)) in section 3 and 4 using the results of learning.
3) Agent moves step by step in the maze making use of MACNN with Eq. (1)-(4) in section 2.

8. COMPUTER SIMULATION

In this section, our proposed method, MACNN with attentive parameters and parity bits, are simulated to confirm their effectiveness through the optimal path searching of maze problems in Fig. 6. Fig. 5 shows the sensor range of the agent. Table 1 shows the action and its code taken by the agent. The agent always moves keeping the attitude that its front of head is always upward of the article and take one action shown in Table 1.

At first 5 mazes in Fig.6 are solved using Q-Learning and their optimal paths are also drawn in each maze by lines with arrows. The coordinate, environment and its corresponding to action of each maze are also shown in Fig.6. Next these pair of environments and action of five mazes is stored in one CNN using form of Eq.(7). Showing in Fig.6, there are 44 patterns to store in the MCNN, and 6 of them are aliasing cases, i.e., some environments are same but each optimal action is different, as following cases, $(5,C)_1$ and $(4,D)_4$, $(2,E)_2$ and $(4,D)_4$, $(6,E)_3$ and $(5,F)_4$, $(3,D)_5$ and $(5,F)_4$, $(3,A)_3$ and $(6,A)_4$, $(2,A)_3$ and $(6,A)_{4,}$ where $(a,b)_,$ c in $(a,b)_c$ means coordinate and the number of mazes, respectively.

8.1 Simulation results only with attentive characteristic without parity bits

In this case, in maze 1 and 2 agent could get the goal within 10 and 12 steps making use of the attentive parameters though there are aliasing (5,C) and (2,C), respectively. In maze 3 and 5, agent could get the goal when the agent select the action as learning chaotically, but it could not get the goal within the predefined trial number when the agent select the other actions. In maze 4, The agent could not get the goal because of aliasing at $(3,B)_3$ and $(6,B)_4$,$(4,D)_4$ and $(5,F)_4$.

8.2 Simulation results with attentive characteristic and parity bits

In this case, the agent could get the goal in all mazes because that the agent could discriminate the appropriate action through the past 4 step parity bits and attentive parameters to recollect the exact patterns.

9. CONCLUSIONS

We proposed a new and simple method to use kinds of attentive parameters and attentive characteristic patterns to recollect collect patterns quickly and correctly, and also proposed the method that uses less memory required parity bits to solve the aliasing problem. It is verified that our proposed methods are useful through the computer simulation of the optimal path search in the maze problem.

REFERENCES

[1] Obayashi, M. and Narita, K. and Kuremoto, T. and Kobayashi, K., A Reinforcement Learning System with Chaotic Neural Networks-Based Adaptive Hierarchical Memory Structure for Autonomous Robots, *Proceeding of International Conference on Control, Automation, and Systems*, pp69-74 (2008)

[2] R.S. Sutton, A.G. Barto, Reinforcement Learning, The MIT Press *(*1998)

[3] Aihara, K. and Takabe, T. and Toyoda, M., Chaotic Neural Networks, *Physics Letters A*, pp.333-340 (1990)

[4] Adachi, K., Tanabe, T., Associative Dynamics in Chaotic Neural Networks, *Neural Networks,* Vol.10, pp.83-98 (1997)

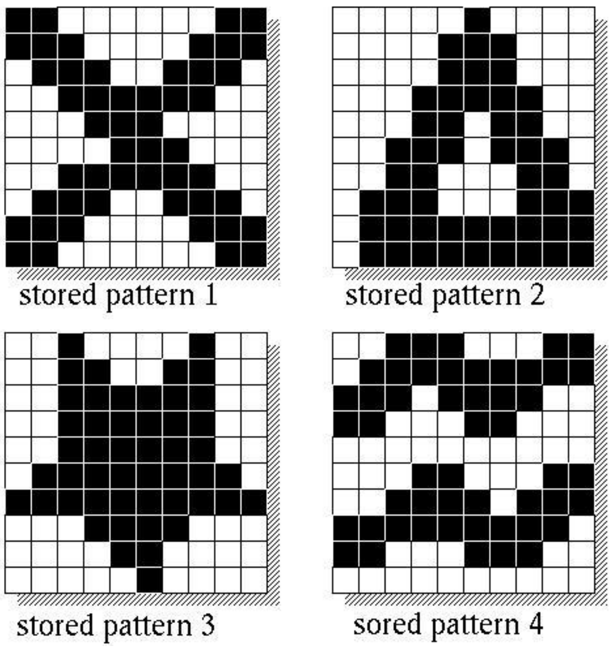


Fig. 1　Stored patterns in CNN

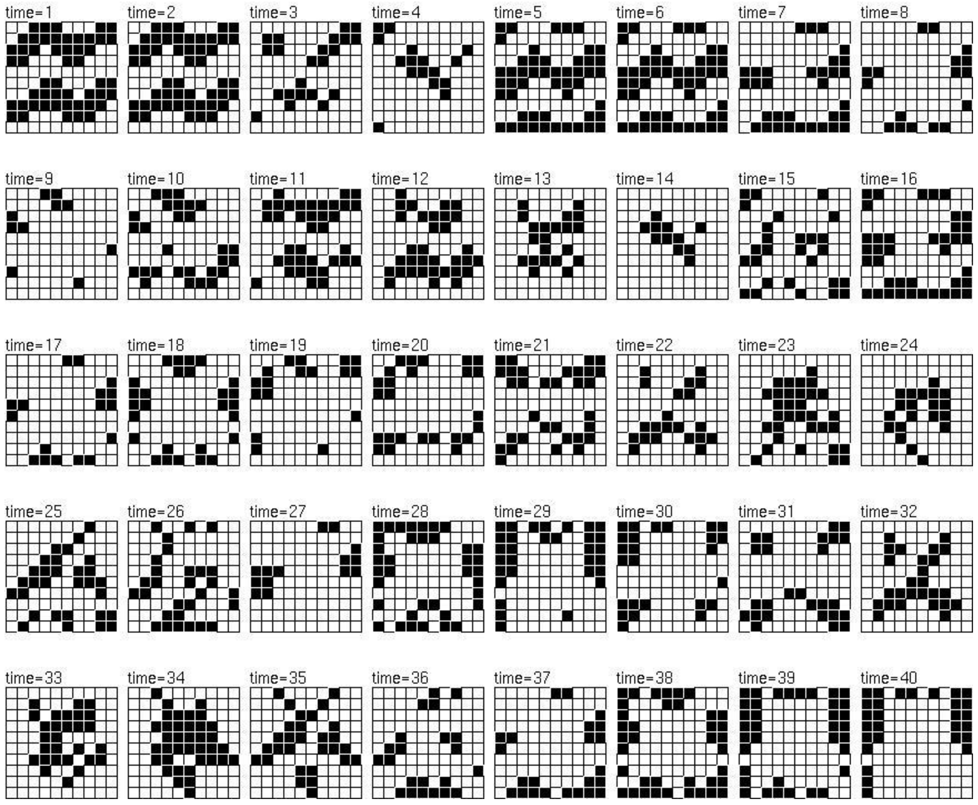


Fig. 2　Behavior of the CNN

Table 1 Parameters used in the computer simulation

k_f	k_f	a_i	α	ε
0.20	0.90	2.0	10.0	0.015

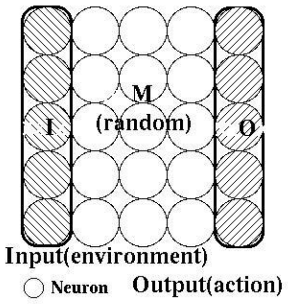
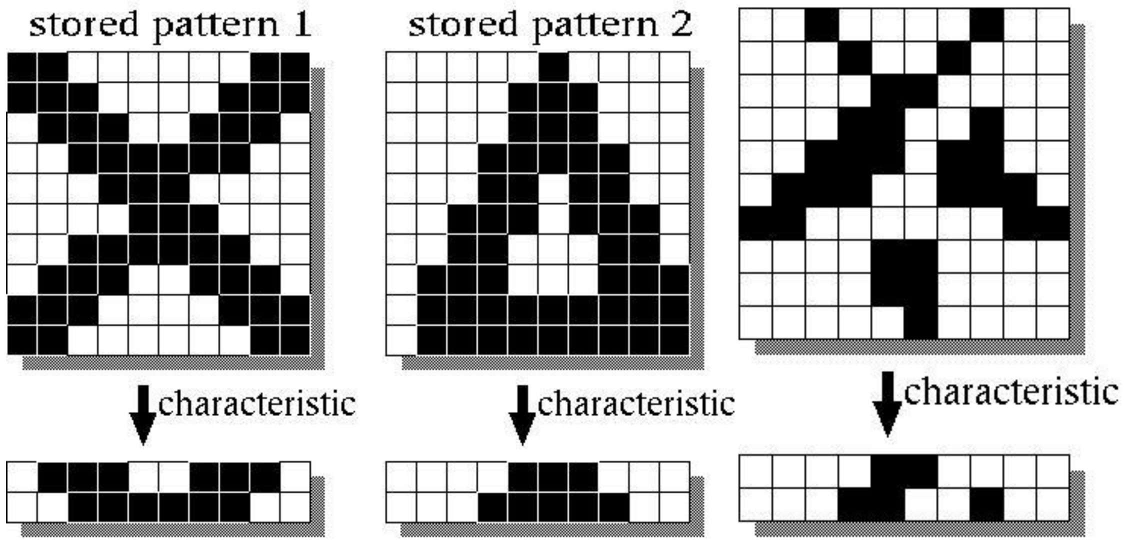


Fig.3 Image of Mutual CNN



(a) (b)

Fig.4 (a) Example of two stored patterns, (b) the characteristic bits at time t during operation

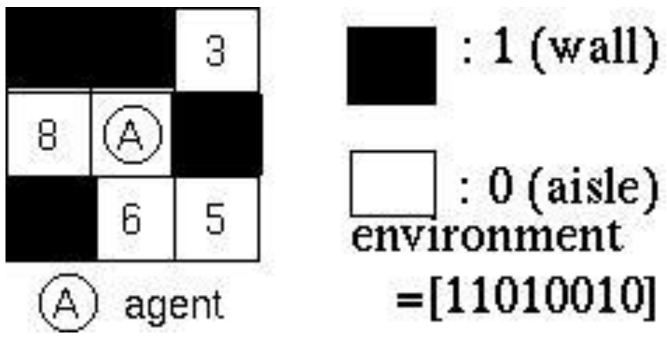
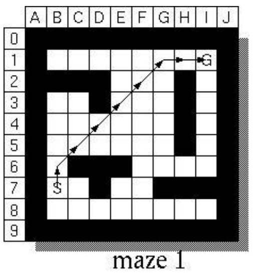


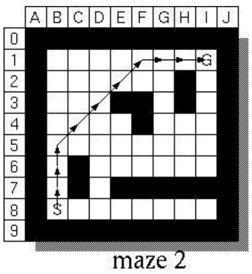
Fig.5 The sensor range of the agent

Table 2 The action of the agent and its code

action	up-left	up	up-right	right	down-right	down	down-left	left
code	0100	0000	0101	0011	0110	0001	0111	0010



maze 1



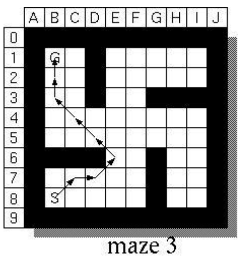
maze 2

maze 1

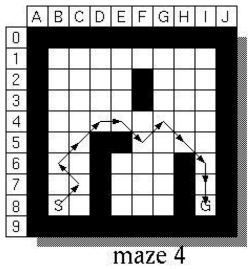
coord.	env.	act.
(7,B)	10100011	up
(6,B)	10010011	up-right
(5,C)	00001100	up-right
(4,D)	01000000	up-right
(3,E)	10000001	right
(3,F)	00000000	up-right
(2,G)	10100011	up-right
(1,H)	11100100	right

maze 2

coord.	env.	act.
(8,B)	10101111	up
(7,B)	10110011	up
(6,B)	10011011	up
(5,B)	10001011	up-right
(4,C)	00000000	up-right
(3,D)	00010000	up-right
(2,E)	00001100	up-right
(1,F)	11100000	right
(1,G)	11101000	right
(1,H)	11100100	right



maze 3



maze 4

maze 4

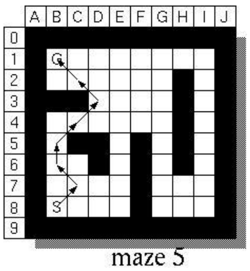
coord.	env.	act.
(8,B)	10001111	up-right
(7,C)	001110000	up-left
(6,B)	10000011	up-right
(5,C)	00011000	up-right
(4,D)	00001100	right
(4,E)	00100110	down-right
(5,F)	00000001	up-right
(4,G)	10000000	down-right
(5,H)	00000100	down-right
(6,I)	00111011	down
(7,I)	10111011	down

maze 3

coord.	env.	act.
(8,B)	10001111	up-right
(7,C)	11100000	up
(7,D)	11000000	up-right
(6,E)	00000001	up-left
(5,D)	00000110	up-left
(4,C)	00100000	up-left
(3,B)	10000011	up
(2,B)	10000011	up

maze 5

coord.	env.	act.
(8,B)	10001111	up-right
(7,C)	00100000	up-left
(6,B)	10100011	up
(5,B)	10010011	up-right
(4,C)	11001100	up-right
(3,D)	00000001	up-left
(2,C)	00000110	up-left



maze 5

Fig. 6 Mazes and its optimal path used in the simu.