Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Realization of Reinforcement Learning using Multi-Winners KFM Associative Memory

Takahiro Ikeya
Graduate School of Bionics, Computer and Media Sciences
Tokyo University of Technology
Hachioji, Tokyo

Yuko Osana
School of Computer Science
Tokyo University of Technology
Hachioji, Tokyo
osana@cs.teu.ac.jp

***Abstract*—In this paper, we propose a multi-winners Kohonen Feature Map (KFM) associative memory, and apply it to reinforcement learning. In the proposed model, the patterns are trained by the successive learning algorithm of the conventional KFM associative memory. The proposed model has two kinds of recall methods, and one of them is selected based on whether or not the input pattern is the trained pattern. In one of the recall method, the output of the input/output layer is calculated as the weighted sum of the connection weights of the fired neuron in the map layer according to their internal states. In the other one method, one of the weight-fixed neurons are selected in the map layer, and the output of the input/output layer is determined based on the connection weights of the neuron. In the reinforcement learning, the proposed model can select the trained corresponding action if the known environment is given. Moreover, it can select appropriate action based on the trained similar situation even if the unknown environment is given.**

***Index Terms*—Kohonen Feature Map(KFM) Associative Memory, Successive Learning, Reinforcement Learning**

I. Introduction

The reinforcement learning is a sub-area of machine learning concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward[1]. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states.

Temporal Difference (TD) learning is one of the reinforcement learning algorithm. The TD learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas. TD resembles a Monte Carlo method because it learns by sampling the environment according to some policy. TD is related to dynamic programming techniques because it approximates its current estimate based on previously learned estimates. The actor-critic method[2] is the method based on the TD learning, and consists of two parts; (1) actor which selects the action and (2) critic which evaluate the action and the state.

On the other hand, neural networks are drawing much attention as a method to realize flexible information processing. Neural networks consider neuron groups of the brain in the creature, and imitate these neurons technologically. Neural networks have some features, especially one of the important features is that the networks can learn to acquire the ability of information processing. The flexible information processing ability of the neural network and the adaptive learning ability of the reinforcement learning are combined, some reinforcement learning method using neural networks are proposed[3][4].

In this paper, we propose a multi-winners Kohonen Feature Map (KFM) associative memory, and apply it to reinforcement learning. In the proposed model, the patterns are trained by the successive learning algorithm of the conventional KFM associative memory. The proposed model has two kinds of recall methods, and one of them is selected based on whether or not the input pattern is the trained pattern. In one of the recall method, the output of the input/output layer is calculated as the weighted sum of the connection weights of the fired neuron in the map layer according to their internal states. In the other one method, one of the weight-fixed neurons are selected in the map layer, and the output of the input/output layer is determined based on the connection weights of the neuron. In the reinforcement learning, the proposed model can select the trained corresponding action if the known environment is given. Moreover, it can select appropriate action based on the trained similar situation even if the unknown environment is given.

II. Multi-Winners KFM Associative Memory

Here, we explain the proposed Multi-Winners KFM (Kohonen Feature Map) Associative Memory (MW-KFMAM).

A. Structure

Figure 1 shows the structure of the proposed model. As shown in Fig.1, the proposed model has two layers; (1) Input/Output(I/O) layer and (2) map layer, and the I/O layer is divided into some parts.

B. Learning Process

The learning algorithm of the proposed model is based on the conventional sequential learning algorithm for the KFM associative memory[5].

In the sequential learning algorithm of the proposed model, the connection weights are learned as follows:

(1) The initial values of weights are chosen randomly.
(2) The Euclid distance between the learning vector $\boldsymbol{X}^{(p)}$ and the connection weights vector $\boldsymbol{W}_i$, $d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_i)$ is calculated.

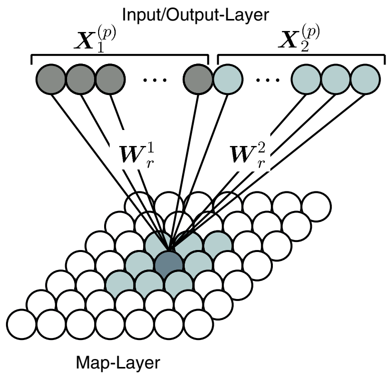


Fig. 1. Structure of Proposed Model.

(3) The winner neuron c whose Euclid distance is minimum is found.

$$c = \underset{i}{\operatorname{argmin}}\, d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_i) \tag{1}$$

(4) If $d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_c) > \theta^c$, the connection weights except those of fixed neurons are updated by

$$\Delta \boldsymbol{W}_i(t) = H(d_i)\alpha(t)h_{ci}(\boldsymbol{X}^{(p)} - \boldsymbol{W}_i(t)) \tag{2}$$

where θ^c is the threshold for internal states of winner neurons. $H(d_i)$ is given by

$$H(d_i) = \frac{1}{1 + \exp\left(-\dfrac{d_i - D}{\varepsilon}\right)} \tag{3}$$

where d_i is the distance between the neuron i and the nearest fixed neuron, D is the constant and ε is the steepness parameter of the function $H(d_i)$. Owing to $H(d_i)$, weights of neurons close to the fixed neurons are semi-fixed, that is, they become hard to be learned. In Eq.(2), $\alpha(t)$ is the learning rate and is given by

$$\alpha(t) = \frac{-\alpha_0(t - T)}{T}. \tag{4}$$

where $\alpha_0$is the initial value of $\alpha(t)$, T is T is the upper limit of the learning iterations. In Eq.(2), h_{ci} is the neighborhood function and is given by

$$h_{ci} = \exp\left(\frac{-\|\boldsymbol{c} - \boldsymbol{i}\|^2}{2\sigma(t)^2}\right) \tag{5}$$

where $\sigma(t)$ is given by

$$\sigma(t) = \sigma_i\left(\frac{\sigma_f}{\sigma_i}\right)^{t/T} \tag{6}$$

In this equation, σ_i is the initial value of $\sigma(t)$ and $\sigma(t)$ varies from σ_i to $\sigma_f(\sigma_i > \sigma_f)$.

(5) (2)$\sim$(4) are iterated until $d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_c) \leq \theta^c$ is satisfied.

(6) The connection weights of the winner neuron c, $\boldsymbol{W}_c$ are fixed.

(7) (2)$\sim$(6) are iterated when a new pattern set is given.

C. Recall Process

The proposed model has two kinds of recall methods, and one of them is selected based on whether or not the input pattern is the trained pattern. In one of the recall method, the output of the input/output layer is calculated as the weighted sum of the connection weights of the fired neuron in the map layer according to their internal states. In the other one method, one of the weight-fixed neurons are selected in the map layer, and the output of the input/output layer is determined based on the connection weights of the neuron.

The recall process of the proposed model is as follows:

(1) The internal states of neurons the map layer are calculated.

- For Binary Patterns
  The internal state of the neuron i in the map layer u_i^{map} is calculated by

$$u_i^{map} = 1 - \frac{d^{in}(\boldsymbol{X}, \boldsymbol{W}_i)}{\sqrt{N^{in}}} \tag{7}$$

  where $d^{in}(\boldsymbol{X}, \boldsymbol{W}_i)$ is the Euclid distance between the input patterns $\boldsymbol{X}$ and the connection weights $\boldsymbol{W}_i$. In the recall process, since all neurons in the I/O-Layer not always receive the input, the distance for the part where the pattern is given is calculated by

$$d^{in}(\boldsymbol{X}, \boldsymbol{W}_i) = \sqrt{\sum_{k \in C}(X_k - W_{ik})^2} \tag{8}$$

  where C shows the set of the neurons in the I/O-Layer which receive the input. N^{in} is the number of neurons which receive the input.

- For Analog Patterns
  The internal state of the neuron i, u_i^{map} is given by

$$u_i^{map} = \frac{1}{N^{in}} \sum_{\substack{k=1 \\ k \in C}}^{N} g(X_k - W_{ik}) \tag{9}$$

$$g(b) = \begin{cases} 1, & (|b| < \theta^d) \\ 0, & \text{(otherwise)} \end{cases} \tag{10}$$

  where θ^d is the threshold.

(2) The winner neurons are determined.

- One-to-Many Associations

  (a) The output of the neuron i in the map layer, x_i^{map} is given by

$$x_i^{map} = \begin{cases} u_i^{map}, & (u_i^{map} \geq \theta^{map} \text{ and the weights} \\ & \text{of the neuron } i \text{ are fixed}) \\ 0, & \text{(otherwise)} \end{cases} \tag{11}$$

  where θ^{map} is the threshold of the neuron in the map layer.

  (b) The winner neuron c is determined in proportion to the internal states. The probability the the neuron

i is selected as the winner neuron c, $P_{i=c}$ is given by

$$P_{i=c} = \begin{cases} \dfrac{u_i^{map}}{\sum_{j=1}^{L} x_j^{map}}, & (x_i^{map} > 0) \\ 0, & \text{(otherwise)} \end{cases} \tag{12}$$

where L is the number of neurons in map layer.

(c) The winner neuron c is determined based on Eq.(12).

- Average Associations The neurons whose internal state is larger than the threshold θ^{map} are selected as winner neurons, and the output of the neuron i in the map layer x_i^{map} is calculated by

$$x_i^{map} = \begin{cases} u_i^{map}, & (u_i^{map} \geq \theta^{map}) \\ 0, & \text{(otherwise)} \end{cases} \tag{13}$$

If there is no neuron shows internal state is larger than the threshold, the threshold is set to the maximum internal state u_{max}^{map}, and the output is calculated as follows:

$$x_i^{map} = \begin{cases} u_i^{map}, & (u_i^{map} = u_{max}^{map}) \\ 0, & \text{(otherwise)} \end{cases} \tag{14}$$

(3) The output of the neurons in the I/O layer is calculated based on the internal states.

- One-to-Many Associations
The output of the neuron k in the I/O layer is calculated by the weight vector between the winner neuron c and the neurons in the I/O layer $\boldsymbol{W}_c$.

$$x_k^{in} = W_{ck} \tag{15}$$

- Average Associations
The output of the neuron k in the I/O layer is given by

$$x_k^{in} = \sum_{i\,:\,x_i^{map}>0} \frac{x_i^{map}}{\sum_{j\,:\,x_j^{map}>0} x_j^{map}} W_{ik}. \tag{16}$$

III. Reinforcement Learning using MW-KFMAM

Here, we explain the proposed reinforcement learning method using the proposed multi-winners KFM associative memory (MW-KFMAM).

A. Outline

In the proposed method, the actor in the Actor-Critic[2] is realized by the proposed MW-KFMAM. In this research, the I/O layer in the MW-KFMAM is divided into two parts corresponding to the state $\boldsymbol{s}$ and the action $\boldsymbol{a}$, and the actions for the states are memorized(Fig.2).

In this method, the critic receives the states which are obtained from the environment, the state is estimated and the value function is updated. Moreover, the critic outputs Temporal Difference(TD) error to the actor. The MW-KFMAM which behaves as the actor (we call this "actor network") is trained based on the TD error, and selects the action from the state of environment. Figure 3 shows the flow of the proposed method.

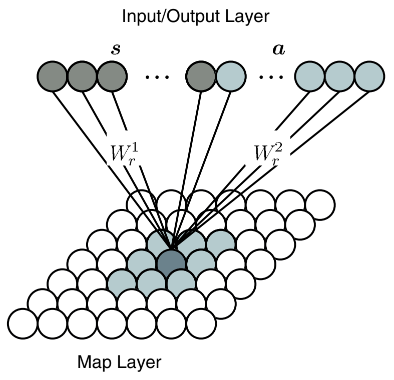


Fig. 2. Structure of Actor Network.

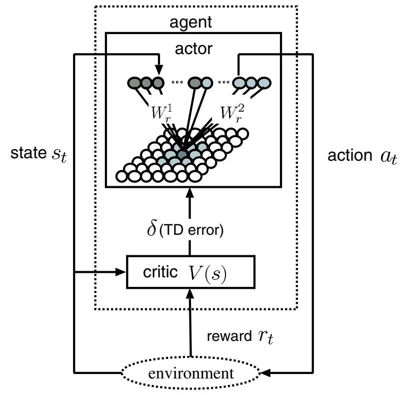


Fig. 3. Flow of Proposed Method.

B. Actor Network

In the proposed method, the actor in the Actor-Critic[2] is realized by the MW-KFMAM.

1) Dynamics: In the actor network, when the state $\boldsymbol{s}$ is given to the I/O-Layer, the corresponding action $\boldsymbol{a}$ is recalled. In the proposed method, the action is selected randomly (random selection), and the more desirable action from the recalled action and the action selected in the random selection is chosen as the action finally.

When the pattern $\boldsymbol{X}$ is given to the network, the output of the neuron i in the Map-Layer at the time t $x_i^{map}(t)$ is given by Eq.(**??**). In the actor network, only the state information is given, so the input pattern is given by

$$\boldsymbol{X}(t) = \begin{pmatrix} \boldsymbol{s}(t) \\ \boldsymbol{0} \end{pmatrix} \tag{17}$$

where $\boldsymbol{s}(t)$ is the state at the time t.

2) Learning: The actor network is trained based on the TD error and the reward from the critic.

The learning vector at the time t $\boldsymbol{X}^{(t)}$ is given by the state $\boldsymbol{s}(t)$ and the corresponding action $\boldsymbol{a}(t)$ as follows.

$$\boldsymbol{X}^{learn}(t) = \begin{pmatrix} \boldsymbol{s}(t) \\ \boldsymbol{0} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{a}(t) \end{pmatrix} \tag{18}$$

The learning vector $\boldsymbol{X}^{learn}(t)$ is trained the following procedures.

(1) When the TD error is larger than the threshold θ^{learn} and the reward is positive

When the desired action $\boldsymbol{a}(t)$ for the state $\boldsymbol{s}(t)$ is selected, the learning vector $\boldsymbol{X}^{(t)}$ is memorized. If the action is selected by the one-to-many associations, the state $\boldsymbol{s}(t)$ and the action $\boldsymbol{a}(t)$ are regarded as the known pattern. So, the connection weights are not updated. If the action is selected by the average associations or the random selection, the connection weights are updated based on the learning algorithm described in **II-B**.

(2) When the TD error is smaller than 0

When the undesired action $\boldsymbol{a}(t)$ for the state $\boldsymbol{s}(t)$ is selected, the connection weights are updated so that the $\boldsymbol{X}^{learn}(t)$ is not recalled.

- One-to-Many Associations

  If the undesired action selected more than θ^{bad} times by one-to-many associations, the connection weights of that neuron are unlocked.
- Average Associations

  The connection weights between the neurons whose internal state is larger than the threshold θ^{m} and the neuron in the I/O layer are initialized randomly. Here, θ^{m} ($\theta^{map} \leq \theta^{m}$) is the threshold for the weight update. If the connection weights of the neurons whose connection weights are fixed are initialized randomly, they are unlocked.
- Random Selection Since the action selected by the random selection is not stored, the connection weights are not updated.

(3) Otherwise The connection weights are not updated.

C. Reinforcement Learning using MW-KFMAM

The flow of the proposed reinforcement learning method using MW-KFMAM.

(1) The initial values of connection weights in the actor network are chosen randomly. The state-value function is set to 0.

(2) The agent observes the environment $\boldsymbol{s}(t)$, and the actor $\boldsymbol{a}(t)$ is selected by the actor network.

(3) The state $\boldsymbol{s}(t)$ transits to the $\boldsymbol{s}(t+1)$ by action $\boldsymbol{a}(t)$.

(4) The critic receives the reward $r(\boldsymbol{s}(t+1))$ from the environment $\boldsymbol{s}(t+1)$, and outputs the TD error δ to the actor.

$$\delta = r(\boldsymbol{s}(t+1)) + \gamma V(\boldsymbol{s}(t+1)) - V(\boldsymbol{s}(t)) \tag{19}$$

where γ $(0 \leq \gamma \leq 1)$ is the decay parameter, $V(\boldsymbol{s}(t))$ is the value function for the state $\boldsymbol{s}(t)$.

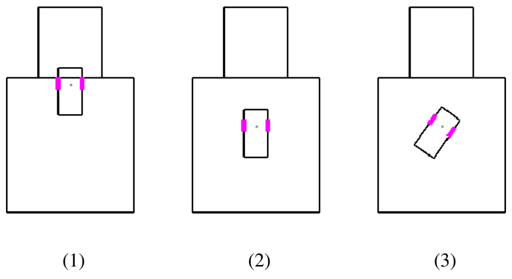


Fig. 4. Initial Position of Car Parking Problem.

(5) All values for states $V(\boldsymbol{s})$ are updated by

$$V(\boldsymbol{s}(t)) \leftarrow V(\boldsymbol{s}(t)) + \xi\delta \tag{20}$$

where ξ $(0 \leq \xi \leq 1)$ is the learning rate.

(6) The connection weights in the actor network are updated based on the TD error.

(7) Back to (2).

IV. Computer Experiment Results

Here, we show the computer experiment results to demonstrate the effectiveness of the proposed method.

We applied the proposed method to the car parking problem. In this experiment, an agent (car) moves toward the upper parking space. The car can observe the distance to the corner of the parking space, and can hand the steering wheel. Figure 4 shows the initial car position of the car parking problem.

A. Variation of Steps

Figure 5 shows the variation of steps to reach the goal. As shown in this figure, the proposed method can learn to go to the parking space. In the cases of (2) and (3), the car could not reach the goal by the conventional actor-critic method.

B. Variation of Recall Methods

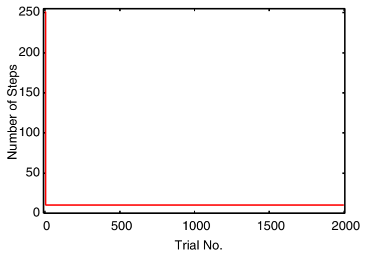
Figure 6 shows the variation of recall methods in the proposed method. As shown in Fig.6(a)$\sim$(c), in the case (1), only one-to-many associations are used after 7 trials. In the case (2), only the one-to-many associations (34.8%) and the average associations (65.2%) are used after 1321 trails. In the case (3), only the one-to-many associations (34.8%) and the average associations (65.2%) are used after 1214 trails.

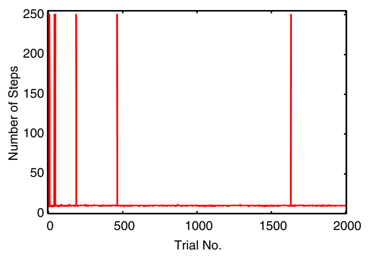
C. Trajectory of Agent

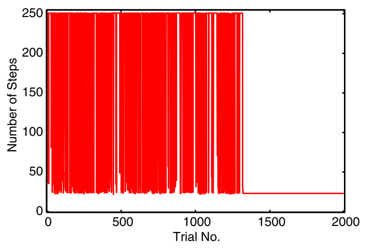
Figure 7 shows the trajectories of the agent before and after learning. As shown in this figure, the proposed method can learn to go to the parking space.

D. Use of Learning Information in Similar Environment

Here, we examined in the actor network that learns in the environment of (3) shown in Fig.4. Here, we used the initial position shown in Fig.8. Figure 9 shows the variation of steps. As shown in Fig.9, the car learns to reach the goal in few steps when the actor network that learns in the environment of (3) shown in Fig.4.


(a) Proposed Method (1)


(b) Actor-Critic (1)


(c) Proposed Method (2)

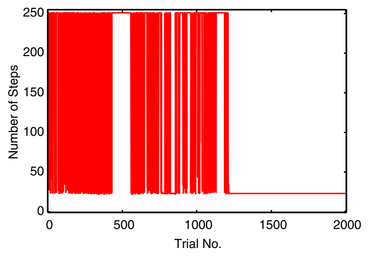
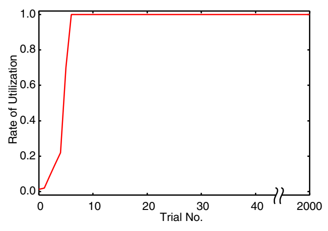
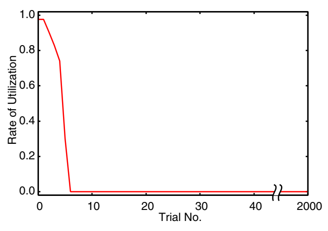
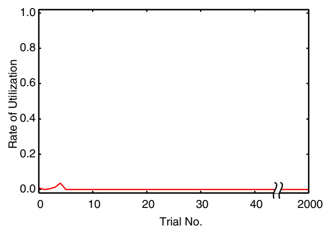
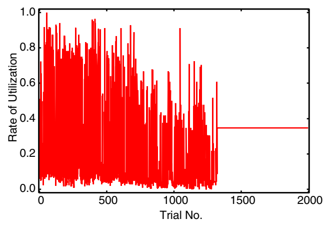

(d) Proposed Method (3)

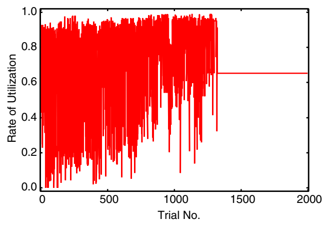
Fig. 5. Variation of Steps.

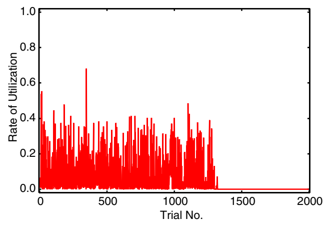

(a) One-to-Many Associations (1)

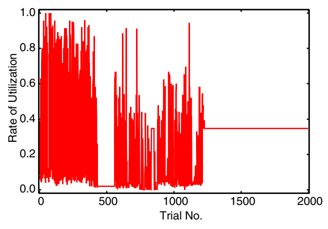

(b) Average Associations (1)

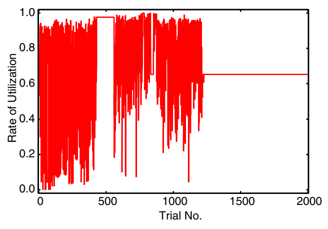

(c) Random Selection (1)

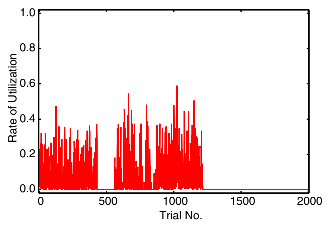

(d) One-to-Many Associations (2)

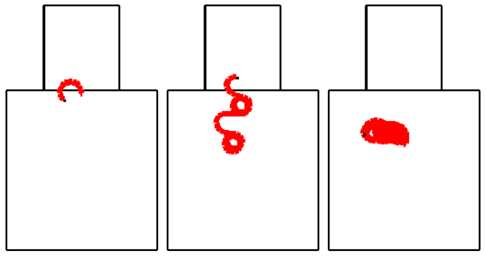

(e) Average Associations (2)


(f) Random Selection (2)


(g) One-to-Many Associations (3)


(h) Average Associations (3)


(i) Random Selection (3)


(a) 1st Trial

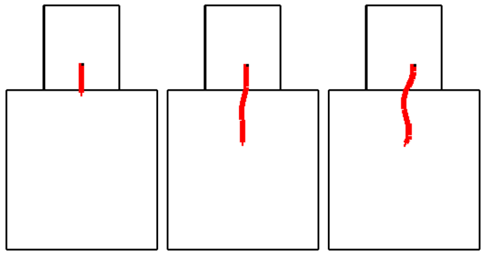

(b) 2000th Trail

Fig. 7. Trajectory of Agent.

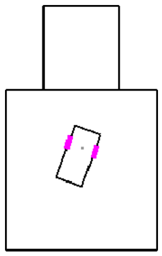
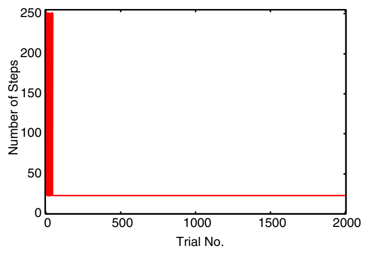


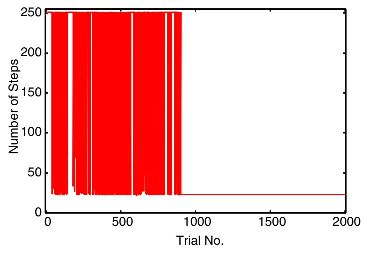
Fig. 8. Similar Initial Position to (3).

V. Conclusions

In this paper, we have proposed a multi-winners Kohonen Feature Map (KFM) associative memory, apply it to reinforcement learning. In the proposed model, the patterns are trained by the successive learning algorithm of the conventional KFM associative memory. The proposed model has two kinds of recall methods, and one of them is selected based on whether or not the input pattern is the trained pattern. In one of the recall method, the output of the input/output layer is calculated as the weighted sum of the connection weights of the fired neuron in the map layer according to their internal states. In the other one method, one of the weight-fixed neurons are selected in the map layer, and the output of the input/output layer is determined based on the connection weights of the neuron. In the reinforcement learning, the proposed model can select the trained corresponding action if the known environment is given. Moreover, it can select appropriate action based on the trained similar situation even if the unknown environment is given.



(a) Learning after (3)



(b) Normal Learning

Fig. 9. Variation of Steps (2).

We carried out a series of computer experiments, and confirmed that the reinforcement learning can be realized using the proposed method.

References

[1] R. S. Sutton and A. G. Barto : Reinforcement Learning, An Introduction, The MIT Press, 1998.
[2] I. H. Witten : "An adaptive optimal controller for discrete-time Markov environments," Information and Control, Vol.34, pp. 286–295, 1977.
[3] K. Shibata, M. Sugisaka and K. Ito : "Fast and stable learning in direct-vision-based reinforcement learning," Proceedings of the 6th International Sysmposium on Artificial Life and Robotics, Vol.1, pp.200–203, 2001.
[4] S. Ishii, M. Shidara and K. Shibata: "A model of emergence of reward expectancy neurons by reinforcement learning," Proceedings of the 10th International Sysmposium on Artificial Life and Robotics, GS21–5, 2005.
[5] T. Yamada, M. Hattori, M. Morisawa and H. Ito : "Sequential learning for associative memory using Kohonen feature map," Proceedings. of IEEE and INNS International Joint Conference on Neural Networks, paper no.555, Washington D.C., 1999.