2010 International Symposium on Nonlinear Theory and its Applications
NOLTA2010, Krakow, Poland, September 5-8, 2010

Reinforcement Learning using Improved Kohonen Feature Map Probabilistic Associative Memory based on Weights Distribution

Shingo NOGUCHI† and Yuko OSANA†

†School of Computer Science, Tokyo University of Technology
1404-1 Katakura, Hachioji, Tokyo, 192-0982, Japan Email: osana@cs.teu.ac.jp

Abstract—In this paper, we propose a reinforcement learning method using Improved Kohonen Feature Map Probabilistic Associative Memory based on Weights Distribution (IKFMPAM-WD). The proposed method is based on the actor-critic method, and the actor is realized by the IKFMPAM-WD. The IKFMPAM-WD is based on the self-organizing feature map, and it can realize successive learning and one-to-many associations. The proposed method makes use of this property in order to realize the learning during the practice of task. We carried out a series of computer experiments, and confirmed the effectiveness of the proposed method in the path-finding problem.

1. 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.

The 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]-[5].

In this paper, we propose the reinforcement learning method using Improved Kohonen Feature Map Probabilistic Associative Memory based on Weights Distribution (IKFMPAM-WD)[6]. The proposed method is based on the actor-critic method, and the actor is realized by the IKFMPAM-WD. The IKFMPAM-WD is based on the self-organizing feature map[7], and it can realize successive learning and one-to-many associations. The proposed method makes use of this property in order to realize the learning during the practice of task.

2. Reinforcement Learning using IKFMPAM-WD

Here, we explain the proposed reinforcement learning method using IKFMPAM-WD[6].

2.1. Outline

In the proposed method, the actor in the actor-critic[2] is realized by the IKFMPAM-WD. In this research, the Input/Output Layer in the IKFMPAM-WD is divided into two parts corresponding to the state $\boldsymbol{s}$ and the action $\boldsymbol{a}$, and the actions for the states are memorized.

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 the Temporal Difference (TD) error to the actor. The IKFMPAM-WD 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 1 shows the flow of the proposed method.

2.2. Actor Network

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

2.2.1. Dynamics

In the actor network, when the state $\boldsymbol{s}$ is given to the Input/Output Layer, the corresponding action $\boldsymbol{a}$ is recalled. In the proposed method, as the state, (1) current and previous states (observations) and previous action and (2) current state (observation) are used. Moreover, the other action is also selected randomly (random selection), and the most desirable action from the recalled actions and the action selected in the random selection is chosen as the action finally.

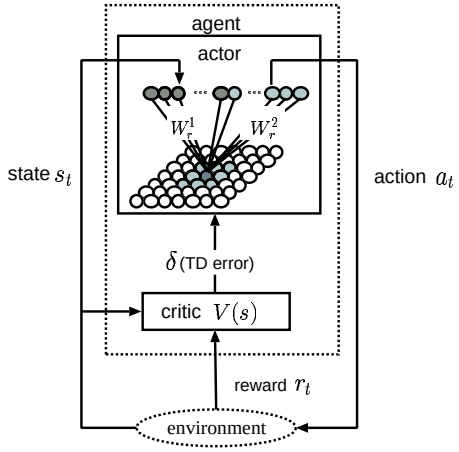


Figure 1: Flow of Proposed Method.

When the actor network uses the current state $\boldsymbol{s}_1(t)$ and the previous state and action $\boldsymbol{s}_2(t)$, the state $\boldsymbol{s}(t)$ is given by

$$\boldsymbol{s}(t) = (\boldsymbol{s}_1(t) \quad \boldsymbol{s}_2(t))^T. \tag{1}$$

When the actor network uses the current state $\boldsymbol{s}_1(t)$, the state $\boldsymbol{s}(t)$ is given by

$$\boldsymbol{s}(t) = (\boldsymbol{s}_1(t) \ -1 \cdots -1)^T. \tag{2}$$

When the pattern $\boldsymbol{X}(t)$ 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

$$x_i^{map} = \begin{cases} 1, & (i = r) \\ 0, & (\text{otherwise}) \end{cases}. \tag{3}$$

In the recall process, the input which does not receive the pattern is set to -1, and the winner neuron r is selected randomly from the neurons which satisfy

$$\frac{1}{N^{in}} \sum_{\substack{k \in C \\ k\,:\,X_k(t) \neq -1}} g(X_k(t) - W_{ik}) \geq \theta^{map} \tag{4}$$

$$g(b) = \begin{cases} 1, & (|b| < \theta^d) \\ 0, & (\text{otherwise}). \end{cases} \tag{5}$$

where N^{in} is the number of neurons which receive the input that not equal -1, C is the set of neurons in the Input/Output Layer which receive the input that not equal -1, and θ^{map} is the threshold of the neuron in the Map Layer. And the input vector is given by

$$\boldsymbol{X}(t) = (\boldsymbol{s}(t) \ -1 \cdots -1)^T. \tag{6}$$

The output of the neuron k in the Input/Output Layer at the time t, $x_k^{io}(t)$ is given by

$$x_k^{io}(t) = \begin{cases} 1, & 0.5 \leq W_{rk} \\ 0, & 0 \leq W_{rk} < 0.5 \\ -1, & W_{rk} < 0. \end{cases} \tag{7}$$

2.2.2. Learning

The actor network is trained based on the TD error from the critic and the eligibility.

(1) Learning based on Reward

In the learning based on the reward, the pair of the state $\boldsymbol{s}(t)$ and the selected action $\boldsymbol{a}(t)$ is memorized. The learning vector $\boldsymbol{X}^{tr}(t)$ is given by

$$\boldsymbol{X}^{tr}(t) = (\boldsymbol{s}(t) \ \ \boldsymbol{a}(t))^T. \tag{8}$$

If the action which is recalled in the actor network only from the current state (observation) is selected, $\boldsymbol{s}(t)$ in Eq.(2) is used. In the other cases, $\boldsymbol{s}(t)$ in Eq.(1) is used.

(a) When State and Action are not Stored

If the pair of the state and the action are not stored and the positive reward is given, that pattern is trained as a new pattern. If the input vector is generated by Eqs.(2) and (8), the learning vector $\boldsymbol{X}^{tr}(t)$ is generated again by Eqs.(1) and (8).

(b) When State and Action are Stored

If the pair of the state and the action are stored and the positive reward is given, the area corresponding to the pattern is expanded. If the pair of the state and the action are stored and the negative reward is given, the area corresponding to the pattern is reduced. Here, the area corresponding to the pattern is the area composed of the neurons which satisfy $d(\boldsymbol{X}^{tr}(t), \boldsymbol{W}_i) > \theta^t$. When the learning vector $\boldsymbol{X}^{tr}(t)$ is generated by Eqs.(2) and (8), all corresponding areas are updated or reduced.

(b-1) When Positive Reward is Given

If the positive reward is given, the area corresponding to the learning vector $\boldsymbol{X}^{tr}(t)$ is expanded. When the learning vector is generated by Eqs.(1) and (8), the size of the area whose center is the neuron z is updated as follows:

$$a_z^{(new)} = \begin{cases} a_z^{(old)} + \Delta a_1^+, & a_z^{(old)} + \Delta a_1^+ \leq a^{max} \\ a_z^{(old)}, & \text{otherwise} \end{cases} \tag{9}$$

$$b_z^{(new)} = \begin{cases} b_z^{(old)} + \Delta b_1^+, & b_z^{(old)} + \Delta b_1^+ \leq b^{max} \\ b_z^{(old)}, & \text{otherwise} \end{cases} \tag{10}$$

where Δa_1^+ is the increment of a_z, Δb_1^+ is the increment of b_z, a^{max} is the maximum of a_z, and b^{max} is the maximum of b_z.

When the learning vector is generated by Eqs.(2) and (8), the size of the area whose center is the neuron z is updated as follows:

$$a_z^{(new)} = \begin{cases} a_z^{(old)} + \Delta a_2^+, & a_z^{(old)} + \Delta a_2^+ \leq a^{max} \\ a_z^{(old)}, & \text{otherwise} \end{cases} \tag{11}$$

$$b_z^{(new)} = \begin{cases} b_z^{(old)} + \Delta b_2^+, & b_z^{(old)} + \Delta b_2^+ \leq b^{max} \\ b_z^{(old)}, & \text{otherwise} \end{cases} \tag{12}$$

where Δa_2^+ ($\Delta a_2^+ \leq \Delta a_1^+$) is the increment of a_z, and Δb_2^+ ($\Delta b_2^+ \leq \Delta b_1^+$) is the increment of b_z.

When the area size is updated, the connection weights are updated as follows:

$$\boldsymbol{W}_i(t+1) = \begin{cases} \boldsymbol{W}_z(t), & d_{zi} \leq D_{zi} \\ \boldsymbol{W}_i(t), & \text{otherwise} \end{cases} \tag{13}$$

where d_{zi} is the distance between the neuron i and the neuron z.

(b-2) When Negative Reward is Given

If the negative reward is given, the area corresponding to the learning vector $\boldsymbol{X}^{tr}(t)$ is expanded. When the learning vector is generated by Eqs.(1) and (8), the size of the area whose center is the neuron z is updated as follows:

$$a_z^{(new)} = \begin{cases} 0, & a_z^{(old)} - \Delta a_1^- < 0 \\ a_z^{(old)} - \Delta a_1^-, & \text{otherwise} \end{cases} \tag{14}$$

$$b_z^{(new)} = \begin{cases} 0, & b_z^{(old)} - \Delta b_1^- < 0 \\ b_z^{(old)} - \Delta b_1^-, & \text{otherwise} \end{cases} \tag{15}$$

where Δa_1^- is the decrement of a_z, and Δb_1^- is the decrement of b_z.

When the learning vector is generated by Eqs.(2) and (8), the size of the area whose center is the neuron z is updated as follows:

$$a_z^{(new)} = \begin{cases} 0, & a_z^{(old)} - \Delta a_2^- < 0 \\ a_z^{(old)} - \Delta a_2^-, & \text{otherwise} \end{cases} \tag{16}$$

$$b_z^{(new)} = \begin{cases} 0, & b_z^{(old)} - \Delta b_2^- < 0 \\ b_z^{(old)} - \Delta b_2^-, & \text{otherwise} \end{cases} \tag{17}$$

where Δa_2^- is the decrement of a_z, and Δb_2^- is the decrement of b_z.

If a_z and b_z become to 0, the area whose center is z disappears and the neuron z is unlocked.

When the area size is updated, the connection weights are updated as follows:

$$\boldsymbol{W}_i(t+1) = \begin{cases} R, & D_{zi}^{(after)} < d_{zi} \leq D_{zi}^{(before)} \\ \boldsymbol{W}_i(t), & \text{otherwise} \end{cases} \tag{18}$$

where R is small random value.

(2) Learning based on Eligibility

In the learning based on the eligibility, the areas corresponding to the state $\boldsymbol{s}_1$ which satisfy

$$e(\boldsymbol{s}_1) < \theta_e \tag{19}$$

are reduced. Here, $e(\boldsymbol{s}_1)$ is the eligibility for the state $\boldsymbol{s}_1$ and θ_e is the threshold for the eligibility.

The area size is updated as follows:

$$a_z^{(new)} = \begin{cases} 0, & a_z^{(old)} - \Delta a_3^- < 0 \\ a_z^{(old)} - \Delta a_3^-, & \text{otherwise} \end{cases} \tag{20}$$

$$b_z^{(new)} = \begin{cases} 0, & b_z^{(old)} - \Delta b_3^- < 0 \\ b_z^{(old)} - \Delta b_3^-, & \text{otherwise} \end{cases} \tag{21}$$

where Δa_3^- is the decrement of a_z, and Δb_3^- is the decrement of b_z.

When the area size is updated, the connection weights are updated by Eq.(18).

2.3. Reinforcement Learning using IKFMPAM-WD

The flow of the proposed reinforcement learning method using IKFMPAM-WD is as follows:

(1) The initial values of weights in the actor network are chosen randomly.

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

(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.

$$\delta = r(\boldsymbol{s}(t+1)) + \gamma V(\boldsymbol{s}(t+1)) - V(\boldsymbol{s}(t)) \tag{22}$$

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

(5) The eligibility $e(\boldsymbol{s})$ is updated.

$$e(\boldsymbol{s}) \leftarrow \begin{cases} \gamma\lambda e(\boldsymbol{s}) & (\text{if } \boldsymbol{s} \neq \boldsymbol{s}(t+1)) \\ \gamma\lambda e(\boldsymbol{s}) + 1 & (\text{if } \boldsymbol{s} = \boldsymbol{s}(t+1)) \end{cases} \tag{23}$$

where γ ($0 < \gamma < 1$) is the decay parameter, and λ is the trace decay parameter.

(6) All values for states $V(\boldsymbol{s})$ are updated based on the eligibility $e_t(\boldsymbol{s})$ ($\boldsymbol{s} \in S$).

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \xi\delta e(\boldsymbol{s}) \tag{24}$$

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

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

(8) Back to (2).

3. Computer Experiment Results

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

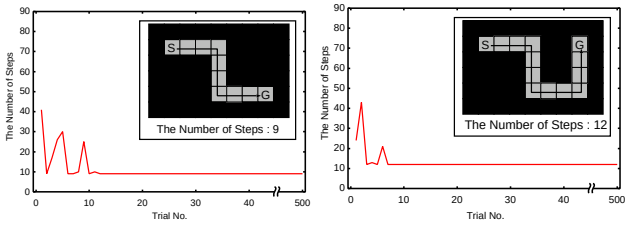
3.1. Path-Finding Problem

We applied the proposed method to the path-finding problem. In this experiment, a agent moves from the start point (S) to the goal point (G). The agent can observe the states of three cells in the lattice, and can move forward/left/right. As the positive reward, we gave 3 when the agent arrives at the goal and 2 when the agent moves. And as the negative reward, we gave -1 when the agent hits against the wall.

Figure 2 shows the transition of number of steps from the start to the goal. In Fig.2, the trained routes (arrows) are also shown. Figure 3 shows an example of the trained relation between the state and the action. And Fig.4 shows the area size transition in the same trail. As shown in this figure, some areas are expanded and the other areas are reduced or disappeared.

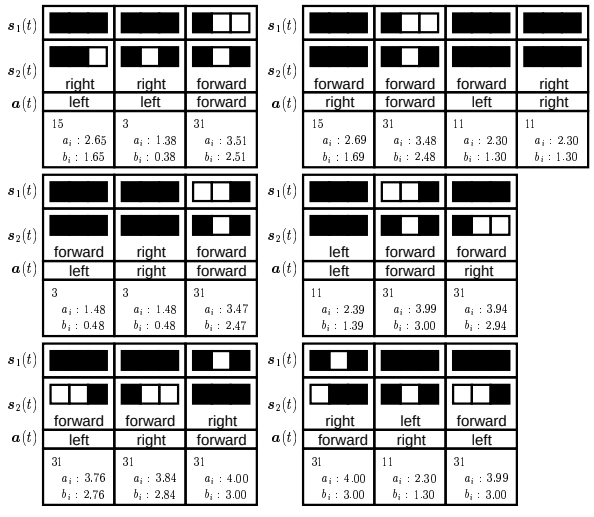
3.2. Learning in Other Environment

Here, the network which was learned in the Map 1 or 2 was used in the Map 3. Figure 5 shows the transition of number of steps from the start to the goal in the Map 3 using the network trained in the Map 1 or 2. As shown in this figure, the proposed method can use the knowledge which was trained in the similar environment.



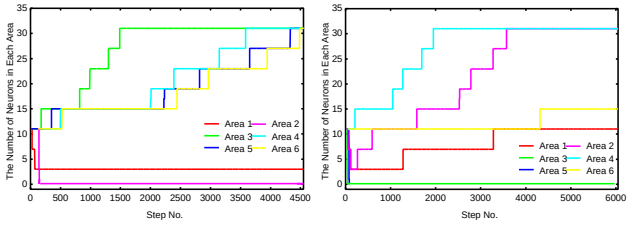
(a) Map 1 (b) Map 2

Figure 2: Trained Route and Transition of Steps.



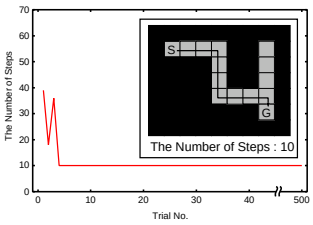
(a) Map 1 (b) Map 2

Figure 3: An example of Trained Relation between State and Action.

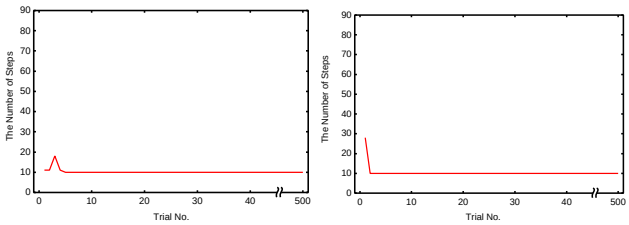


(a) Map 1 (b) Map 2

Figure 4: Transition of Area Size.



(a) Map 3



(b) Map 1 → Map 3 (c) Map 2 → Map 3

Figure 5: Transition of Steps (Map 3).

4. Conclusions

In this paper, we have proposed the reinforcement learning method using IKFMPAM-WD. The proposed method is based on the actor-critic method, and the actor is realized by the IKFMPAM-WD. We carried out a series of computer experiments, and confirmed the effectiveness of the proposed method in path-finding problem.

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] S. Ishii, M. Shidara and K. Shibata: "A model of emergence of reward expectancy neurons by reinforcement learning," Proceedings of the 10th International Symposium on Artificial Life and Robotics, GS21–5, 2005.

[4] A. Shimizu and Y. Osana : "Reinforcement learning using Kohonen feature map associative memory with refractoriness based on area representation," Proceedings of International Conference on Neural Information Processing, Auckland, 2008.

[5] Y. Osana : "Reinforcement learning using Kohonen feature map probabilistic associative Memory based on weights distribution," Proceedings of International Symposium on Nonlinear Theory and its Applications, Sapporo, 2009.

[6] S. Noguchi and Y. Osana : "Improved Kohonen feature map probabilistic associative memory based on weights distribution," Proceedings of IEEE and INNS International Joint Conference on Neural Networks, Barcelona, 2010.

[7] T. Kohonen : Self-Organizing Maps, Springer, 1994.