Episode-based Reinforcement Learning – an Instance-Based Approach for Perceptual Aliasing

T. Unemi & H. Saitoh[1]
Department of Information Systems Science, Soka University, Hachioji, Tokyo, 192-8577 JAPAN
unemi@iss.soka.ac.jp, ganji@intlab.soka.ac.jp

ABSTRACT

This paper proposes a reinforcement learning method based on memorizing and retrieving episodes of the learner's own experiences. The results of the computer simulation on a simple but typical non-Markovian environment is shown to clarify the performance. An instance-based reinforcement learning method previously proposed by an author is also based on the learner's experiences memorized without any modification. But it is applicable only to Markovian domain where it is enough for the learner to acquire a reactive policy to achieve the optimal behavior. An episode-based method is not only overcome a perceptual aliasing but also inherit the advantages of instance-based method on flexibility for applicable domain.

1 INTRODUCTION

A reinforcement learning method presented here is the one the author have proposed ten years ago, but it has not revealed in publishing in the international domain. This paper describes its mechanism and a recent experimental results on the computer simulation.

Following the classification of learning strategies by Carbonell[1], the basic approach is a framework of *rote learning* of which origin can be found in Samuel's Checker Player[2]. *Memory-based reasoning* proposed by Stanfill and Waltz[3] and many other approaches named *instance-based*[4, 5] and *exemplar-based learning*[6] and *nearest neighbor classification*[7] can be seen as its successors.

The author has presented a simpler version of reinforcement learning method based on a rote learning scheme as an *instance-based reinforcement learning method*[8]. In all of these methods, the learner's experiences are memorized without any modification and are referred for decision making, in contrast with the other learning paradigm such as inductive learning, chunking and explanation based learning. A difference with neural networks is that they rely on the modification of memory structure and the retrieval by similarity rather than adjustment of connection weights. We refer the approach described here by *Episode-Based Reinforcement Learning method* (EBRL) because the memory is organized in a form of a set which includes time sequences of experiences.

How input/output data are structured is one of the important features to characterize any learning mechanisms as similarly as strategies. Discrete time sequence of relatively small grain size is the data structure mentioned here, which comes from situation of living organisms in the real world. The input is an unlimited sequence of vector sampled in a short time interval, and the output is a similar sequence of signals. One of the distinct characteristics of this setting is that there is no obvious boundaries to extract any data unit from input sequence. The learner must mention some cluster of input data as a unit, because data given at one primitive step are too simple to give them any meanings without any relation between others, and there is no explicit information provided from the environment to obtain what range of data should be treated as one cluster. Clustering the raw data is one of the unavoidable tasks of the learner.

In the rest part of the paper, we show an overview of typical learning environment, describe some detail of the learning mechanism, and present some experimental results and discussion.

2 LEARNING ENVIRONMENT

We had examined the performance of EBRL by an artificial insect world ten years ago[9], but it is not a suitable task to clarify the performance against non-Markovian environment. So, we designed a new simpler type of example task on a mobile robot navigation as shown in Figure 1. The robot learns the navigation strategy to move from the start position to goal area. The action set includes three candidates, turn $-90°, 0°$, and $90°$ than go ahead by a constant distance. The sixteen sensors detect the distances from the robot surface to any obstacle by a precession of six bits integer, that is 0–63. The learner gets reward $r_t = 1$ when it reaches at the goal area, is punished $r_t = -1$ when it collides against the wall, and has nothing $r_t = 0$ otherwise. After the robot reached at the goal area, it is forced to be back to the start position to try moving toward the goal again. The performance can be measured by the average value of rewards par trial steps. In addition to this sort of typical setting for learning from delayed reward, the learner's sensory

[1] The second author is currently working at Nippon Telecommunications System Co.

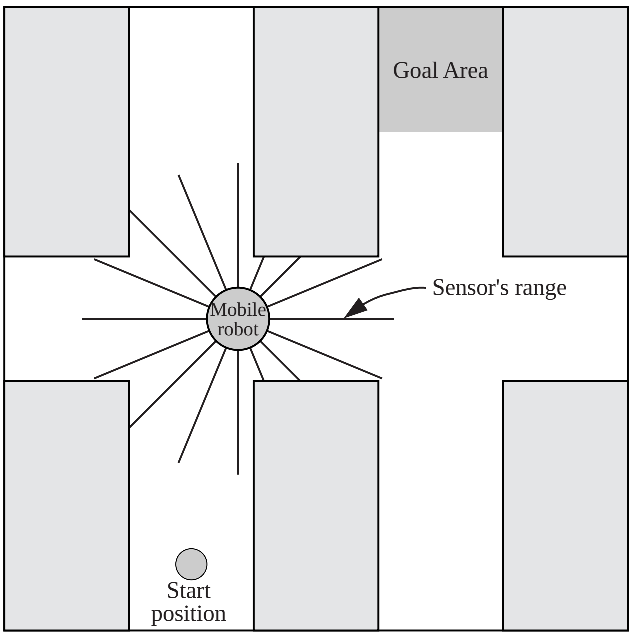


Figure 1: Environment of experimental task on a robot navigation.

data are only the local information that cannot give one to one mapping to the absolute position so that the environment includes the problem on *perceptual aliasing*. Two crossings of the corridors ensure this feature because only from the current sensory data the robot cannot distinguish which crossings it locates at. It has to use some information of the context, the history of its moves, to detect the correct position.

3 LEARNING MECHANISM

The learner adjusts the weights of memory elements according to some sorts of heuristics to acquire a plausible strategy through its own experience, and improves its performance in terms of getting ability to make a suitable decision in each situation. The details of this mechanism are shown below.

3.1 Structure of Memory Organization

The memory is organized in a manner of a collection of sequences each of which includes two types of nodes named *Sensory Memory Elements* (SMEs) and *Action Memory Elements* (AMEs), and directed links between them. Each SME consists of three components, sensory data, weight, and expected preference value. Each AME has two components, action data and weight. Each sequence is arranged in the order of time when the data were input or output, such as $S_1 \rightarrow A_1 \rightarrow S_2 \rightarrow A_2 \rightarrow \cdots \rightarrow S_n \rightarrow A_n$ where S_i is the SME acquired at time i and A_i is the AME executed at time i. In addition to this structure, the mechanism employs a set of constant number of pointers to memory elements with weight values. This set is named *Recall Table*. Each weight of *Recall Table Elements* (RTEs) indicates the similarity between pointed memory data and the current situation. It can be said that the recall table shows the learner's view of current situation in each step. One of the RTEs may be the *plan*, of which action data is a candidate of next action. There may or may not be a plan. Figure 2 shows a schematic illustration of the organization of memory. The method to manage recall table and the plan will be described later.

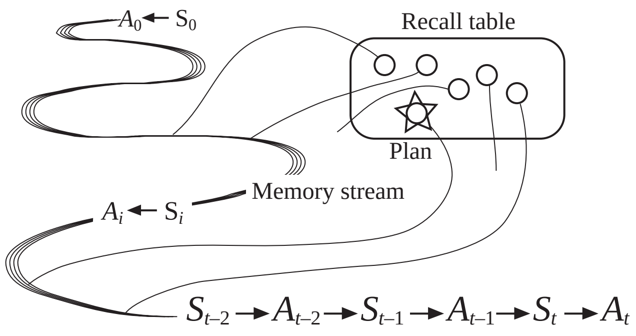

$A_0 \leftarrow S_0$
Recall table
Plan
Memory stream
$A_i \leftarrow S_i$
$S_{t-2} \rightarrow A_{t-2} \rightarrow S_{t-1} \rightarrow A_{t-1} \rightarrow S_t \rightarrow A_t$

Figure 2: Schematic illustration of the memory organization.

3.2 Memorizing and Forgetting

In each step, input and output data tend to be memorized with a constant weight 0.5, and to be connected with the memory element of the last step. To avoid the explosion of memory space, the number of SMEs and AMEs are restricted to a constant, so some data in memory elements must be forgotten to replace them with new data. The candidate for replacement is chosen according to the value of weight, that is, the element with the lowest weight tends to be forgotten. Weights of all of memory elements are diminished in each step by multiplying a constant η of (0, 1). This mechanism follows the heuristic:

Heuristic 1 *Recent data are more reliable than old data.*

Another type of weight adjustment is also done in the phase of reflection. Since the objects for modification of weights, except diminishing, is restricted to few memory elements at one step, it is possible to employ an efficient algorithm to find the element of the lowest weight, using a balanced binary tree where each mother node has lower weight than its daughters so as to keep the lowest weight always at the top node. It takes $O(\log N)$ in time complexity to insert, modify and remove one memory element, where N is the number of elements in the memory, but $O(1)$ to retrieve the lowest one. The memory is organized in a single chain until the number of processing steps reaches the number of maximum memory size, but after that step it becomes a collection of a number of chains since forgetting cuts a chain. Useless nodes, such as a single node with no connection and an AME not followed by any SME, are also forgotten.

3.3 Recalling Similar Experience

One of the key techniques supporting this mechanism is to catch the current situation by recalling similar data from memory. Moore[10] proposed a learning mechanism closely related this approach except that it does not support context sensitive features of environment, where the type of object data is quite same. He pointed out that an efficient algorithm to retrieve similar data have developed in the field of computational geometry, where the similarity between two objects is defined using Euclidean distance. The distance actually used in this mechanism is defined as follows.

$$\mathcal{D}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\mathrm{V}_{\mathrm{X}i}}} \quad (1)$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of which elements is denoted by x_i and y_i, and $\mathrm{V}_{\mathrm{X}i}$ is variance of ith element over the data already memorized. The variances can be calculated as the difference between expected value of square and square of expected value, so it is easy to get its value at each step by keeping the sum of data and sum of square of data which are modified in memorizing and forgetting process. The similarity measure used in this mechanism is defined as follows.

$$\mathcal{S}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\mathrm{E_D} - \mathcal{D}(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{\mathrm{V_D}}} \quad (2)$$

where $\mathrm{E_D}$ is the expected value of distance and $\mathrm{V_D}$ is the variance. The reason why this style of normalization was employed is that the value of similarity is used in combination with preference value in the process of decision making described later. It is somewhat difficult to get the exact value of $\mathrm{E_D}$ and $\mathrm{V_D}$ efficiently in each step, but Fujino[11] suggested a feasible method to estimate these values under the assumption that distribution of each element of object vector is formed in same figure, the number of data is large enough and the number of dimensions is more than four or five. The value of $\mathrm{E_D}$ can be estimated by the following expression.

$$\frac{\mathrm{E_D}}{\sqrt{2n}} \approx 1 - \frac{1+\mu_4}{16n} - \frac{127 - 42\mu_4 + 15\mu_4^2 - 8\mu_6}{512n^2} - \frac{210 - 540\mu_4 + 145\mu_4^2 + 100\mu_6 + 5\mu_8}{1024n^3} \quad (3)$$

where n is the number of elements of one object vector and μ_k is moment of order k of normalized data, which is defined by

$$\mu_k = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{x_i - \mathrm{E_X}}{\sqrt{\mathrm{V_X}}} \right)^k \quad (4)$$

where N is the number of memory elements. The expected value of square of distance is

$$\begin{aligned} \mathrm{E_{D^2}} &= \frac{2}{N(N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i-1} \sum_{k=1}^{n} \frac{(x_{ik} - x_{jk})^2}{\mathrm{V}_k} \\ &= \frac{2Nn}{N-1} \approx 2n \text{ when } N \to \infty \end{aligned} \quad (5)$$

where x_{ik} is the ith element in kth object in memory. The value of $\mathrm{V_D}$ can be calculated as

$$\mathrm{V_D} = \mathrm{E_{D^2}} - \mathrm{E_D^2} \approx 2n - \mathrm{E_D^2} \quad (6)$$

The initial value of RTE's weight is $\mathcal{S}(\boldsymbol{x}, \boldsymbol{m})$ where $\boldsymbol{x}$ is the currently input data and $\boldsymbol{m}$ is the data recalled. To catch a context of current situation, data involved in any RTE are replaced to new one which is connected with old one by a link, in each step. At the same time, RTE's weight is modified by combining old weight and similarity between new data and the data newly coming from environment. The combination is the following weighted summation.

$$W_t^i = \alpha \cdot W_{t-1}^i + (1-\alpha) \cdot \mathcal{S}(\boldsymbol{x}_t, M_t^i) \quad (7)$$

where W_t^i is the weight of ith RTE at time t, $\boldsymbol{x}_t$ is the input data at time t, and M_t^i is the input data pointed by ith RTE associated from M_{t-1}^i following two links. α is a constant of (0, 1) which indicates how many steps the learner considers as a context. The smaller α becomes the longer interval of context considered. No consideration would be given to the context if the value was one. The newly retrieved data with relatively high similarity participate the competition with old ones in each step, and RTEs of height weight survive in recall table.

3.4 Prediction and Reflection

To decide the next action, the insect must make a prediction to see what will be happened in near future in terms of causal relation between alternative actions and preference values. The prediction will be established by looking data ahead through link by link, under support of the following heuristic rule.

Heuristic 2 *Similar situation likely follows similar situation.*

Instead of inefficient association of nodes through many links in each step, actually it assigns the expected preference values for each SME in order to get a feel about aspects in future by referring them. This assignment is done when the learner encounters reward or punishment, by propagating the reward value backward through links from the last node diminishing the value gradually, until its value becomes less than the threshold or it reaches the node already assigned the expected preference value. This propagation stands on the following heuristic rule.

Heuristic 3 *The more the event is closely followed by a good event, the more it seems good but less good than that.*

New expected preference value (EPV) of SME formally represented by

$$E^{\tau} = \beta^{t-\tau} \tag{8}$$

where E^{τ} is the EPV of SME memorized at time τ, and β is a constant of (0, 1). The learner should do some sorts of modification of its knowledge when it encounters a valuable or detestable experience, as any inductive leaner do for positive or negative examples. One of the operations is assignment of expected preference values described above. The others are for the plan currently executed. The plan, if exists, recommends the action to get a valuable object in near future. So, if the learner gets some reward, more support should be given to the plan, and if punishment then its reliability should shrink, based on the following heuristic rule.

Heuristic 4 *The more the rule is frequently useful and the less it leads failure, the more it seems reliable.*

This principle is realized by modifying weight and preference values of SMEs tracing through some links backward from SME on the current plan. When it encounters a punishment as a result of the plan of last step, the EPVs of SMEs connected to the plan are revised by the following equation.

$$E^i_t \;=\; E^i_{t-1} - (1-\gamma_1^{W^p}) \cdot \gamma_2^s \cdot (E^i_{t-1}+1) \tag{9}$$

where W^p is the weight of RTE of the current plan, is the number of steps to associate the current plan, and γ_1 and γ_2 are constants of (0, 1). This equation says that the more the SME is closely connected to the current plan and the more the current plan is similar to current situation, the more its expected preference value is decreased. Simultaneously, weight of SME are modified using the following equation.

$$w^i_t = \begin{cases} w^i_{t-1} + \rho_1 \cdot \rho_2^s \cdot (1 - w^i_{t-1}) & \text{if } r_t = 1 \\ w^i_{t-1} - (1-\rho_1) \cdot \rho_2^s \cdot w^i_{t-1} & \text{if } r_t = -1 \end{cases} \tag{10}$$

where w^i_t is the weight of ith SME at time t, and ρ_1 and ρ_2 are constants of (0, 1). The weight increases if it gets reward, and decreases if punished, keeping that value to be in the range from 0 to 1. Thus, more reliable sequence gets more weight and less reliable one loses some weight, so as to remain the useful sequences in the memory. It seemed that the learner should do something when it expected a food but actually got nothing. But, currently it is not implemented yet.

3.5 Decision Making

The insect decides its own action step by step. That decision depends on the plan if exists, otherwise on the short term prediction of expected preference value. If there is no evidence of some strength to choose a single action, it randomly selects the next action from a finite set of alternatives. The following heuristic is important for this kind of learning by exploration.

Heuristic 5 *It is better to do anything than nothing.*

When it has a plan, which is one of RTEs as described above, it selects the data in AME of that plan as the next action, as it revises each of SMEs involved in RTEs into next AMEs connected with them before it makes a decision. Acceptance or rejection of the plan is judged referring a measure computed from the weight of RTE and the EPV of SME in RTE, which is formally represented

$$m_i \;=\; \min(W_i, E_j) \tag{11}$$

where i is the index of RTE and j is the index of SME pointed by ith RTE. The reason why we take a minimum is that it seems natural to make a conjunction of these two measures and a minimum corresponds to conjunction in a similar sense of Fuzzy logic. That is, it will be desirable to accept the RTE as a new plan of which both of these values are large. The current plan is rejected if its value of m_i becomes less than the threshold value θ, which indicates the learner's satisfaction. The value of θ increases in each step without reward, and reset to the lowest value when it gets reward. When m_i is less than θ, it tries to employ another RTE as a new plan by investigating values of m_i for all RTEs to choose the maximum value. Of course, nothing is employed if that maximum value is less than the threshold. In current implementation, the learner tends to keep current plan so that some length of action sequence can be executed continuously, instead to select the best plan from recall table in every steps. The length of that sequence depends on some parameters such as θ and α.

In the initial state, there is no plan. Whenever it has no plan, it tends to see which RTE is best as a new plan, as similarly as the case of rejection. In the case that it even fails to select a new plan, it tries to evaluate each of alternative actions by checking EPV of next SME connected with AME in all of RTEs, based on the following heuristic rule.

Heuristic 6 *If you don't have any good alternatives, find less bad one.*

It calculates the value of m_i for all of RTEs using EPV of the next step SME, makes the summations of m_i of same action for each of alternatives, and recommends an action with relatively height value as the next one. If that value is less than θ then the recommendation is rejected, that is, the learner will put its next step toward a random direction.

4 EXPERIMENTS

We examined the proposed learning method described in the previous section through computer simulation of a type of robot navigation shown in section 2.

The environmental settings are as follows. The size of the room where the robot moves around is 320 units by 320 units square. The width of the corridors is 64 units. The size of the robot is 32 units in diameter. The length of move in one step is 16 units. The range of the distance sensors is within 64 units.

The learning parameters are set as follows. The diminishing factor η for aging is 0.98. The adjustment factor α of the weight of RTE in equation (7) is 0.4. The discount factor β of back-propagation of reinforcement signal in equation (8) is 0.98. The discount factor γ_1 is 0.8 and γ_2 for unreliable SME in equation (9) is 0.85. The adjustment factor ρ_1 and ρ_2 for the weight of SME in equation (10) are 0.5 and 0.95 respectively.

We investigated the effects of the size of memory and the size of recall table. Figure 3 shows the learning curves drawn through experiments on the computer simulation. This learning mechanism works well even if the recall table includes only three elements. In the case where the size of recall table is 15, it performs better in any size examined here than the case of smaller size of recall table. Larger capacity of the memory causes better performance but slow improvement in early stage of learning.

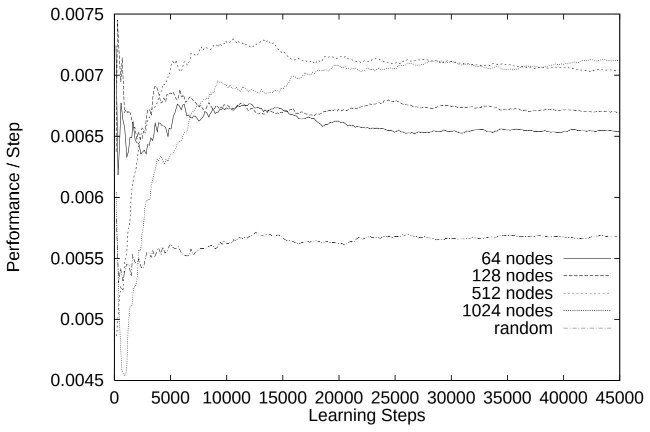
The other experiments we did is on the adaptability against a type of environmental change. We examined to change the start position from A to B in Figure 4 after 30,000 steps of learning. As Figure 5 shows, it is difficult to adapt to the new situation when the memory size is large. The reason could be that the effects of forgetting is works more effectively when the memory size is small. It might be a dilemma between performance and adaptability, namely exploitation vs. exploration. We should introduce an additional function to reinforce forgetting when the learner detects some degree of environmental change.

We also examined other reinforcement methods, Window-Q and Recurrent-Q proposed by Lin[12] for non-Markovian environment, to the same problem. Both of them could perform better than a random behavior but worse than the proposed method. Unfortunately they would not be the best performance because they also have parameters to adjust to the problem domain.

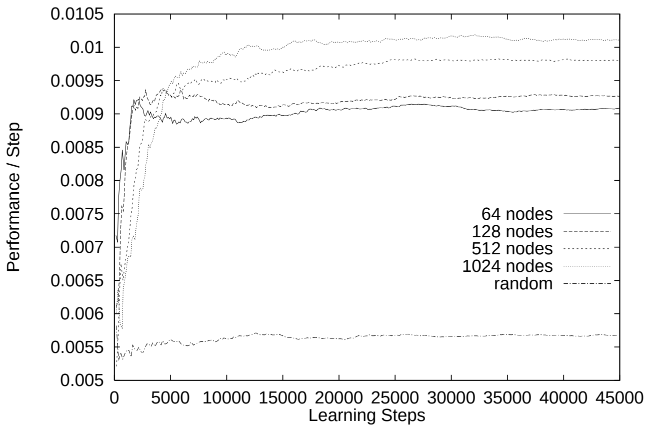
5 DISCUSSION

We proposed a learning mechanism, which can be classified into non symbolic approach, but neither neural networks nor connectionist's model. In a sense of architecture, this is directed to numerical computation and manipulating data structures on traditional Neumann machine, rather than brain or symbolic computation. From this point of view, our approach has a similarity with genetic algorithm, adaptive automaton, and so on, even though their basic strategies are quite different.

One of further extensions of this model is to in-



a. The case the size of recall table is 3.



b. The case the size of recall table is 15.

Figure 3: Average performances over 30 trials of distinct random number sequences for a variety of the memory size.

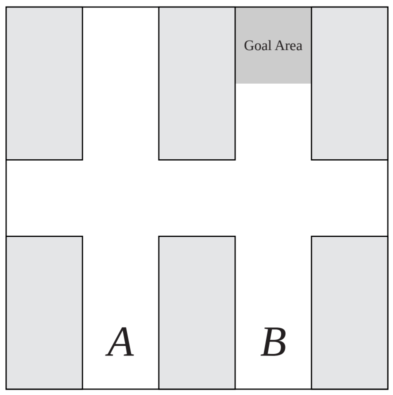


Figure 4: Start position A and B for the experiments on the adaptability against environmental change.

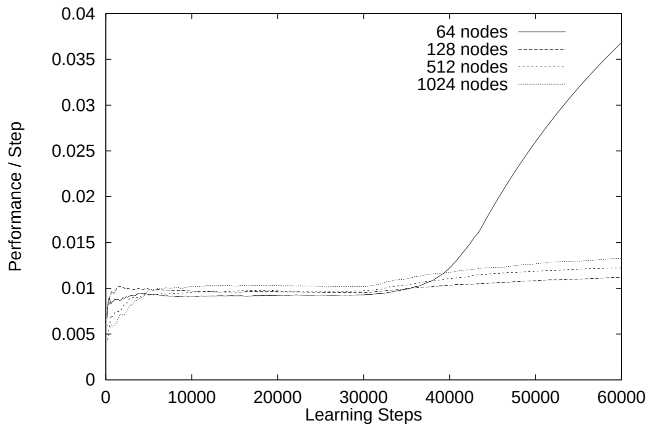


Figure 5: Performances against environmental change. Each lines index average over 30 trials for a variety of the memory size.

troduce a mechanism to provide any sense of data abstraction. Claim from symbolism may be that it must be much difficult for these kinds of approach to explain human intelligence observed in many sorts of problem solving. However, we believe that it may be an entrance to human intelligence to add data abstraction mechanism to our framework. It may be possible to apply this mechanism to another practical domain such as numerical prediction in financial activities. Our primary motivation is to search a model of intelligent organism, but any practical challenge is also our interest.

6 ACKNOWLEDGEMENT

The authors appreciate Atsunobu Ichikawa, Shigenobu Kobayashi, Kiyoshi Akama, Susumu Kunifuji, Yuichiro Anzai, Hideyuki Nakashima, Kouichi Hori, Satoshi Satoh, Hiroshi Deguchi, Hitoshi Matsubara and Natsuki Oka for their helpful comments and encouragement for this research. The first author specially wish to dedicate this article to Yoritake Fujino who closed his life in 1992.

REFERENCES

[1] J. G. Carbonell, "Introduction: Paradigms for Machine Learning," *Artificial Intelligence,* Vol. 40, pp. 1–9, 1989.

[2] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal on Research and Development,* Vol. 3, pp. 210–229, 1959.

[3] C. Stanfill and D. Walz, "Toward Memory Based Reasoning," *Communications of the ACM*, Vol. 29, pp. 1213–1228, 1986.

[4] D. W. Aha, D. Kibler and M. K. Albert, "Instance-Based Learning Algorithm," *Machine Learning*, Vol. 6, pp. 37–66, 1991.

[5] R. A. McCallum, "Instance-Based Utile Distinctions for Reinforcement Learning with Hidden State," *Proceedings of the 12th International Conference on Machine Learning*, pp. 387–395, 1995.

[6] S. Salzberg, "A Nearest Hyperrectangle Learning Method," *Machine Learning*, Vol. 6, pp. 251–276, 1991.

[7] B. V. Dasarathy (ed.), "Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques," IEEE Computer Society Press, 1990.

[8] T. Unemi, "An Instance based Reinforcement Learning Method," *Journal of Japanese Society for Artificial Intelligence*, Vol. 7, No. 4, pp. 697–707, 1992. (in Japanese)

[9] T. Unemi, "A Rote Learning Mechanism for Discrete Time Sequences and Its Application for Simulation of Adaptive Behavior," Workshop on Learning '90 in Hokkaido, 1990. (in Japanese)

[10] A. W. Moore, "Acquisition of Dynamic Control Knowledge for a Robotic Manipulator," *Proceedings of the Seventh Conference on Machine Learning*, pp. 244–252, 1990.

[11] Y. Fujino, *private communication*, at Department of Planning and Management Science, Nagaoka University of Technology, 1990.

[12] L.-J. Lin, "Scaling Up Reinforcement Learning for Robot Control," *Proceedings of the Tenth Conference on Machine Learning*, pp. 182–189, 1993.