Learning Algorithms for Robots Behaving Flexibly in Dynamic Environments

Masahiro Ono, Hiroshi Ando, Mamoru Sasaki and Atsushi Iwata

Graduate School of Advanced Sciences of Matter, Hiroshima University,
1-3-1 Kagamiyama, Higashi-Hiroshima-shi 739-8527, Japan
Phone and Fax: +81-824-22-7358, Email: {masa, ando, sasaki, iwa}@dsl.hiroshima-u.ac.jp

1. Introduction

Recently, the progress of robots is remarkable in the field of high speed operation, humanoid, imitation of behavior and entertainment. Nevertheless, few autonomous robots having behavior-learning capability are developed. The main reason is that learning is insecure to environmental change and the learning speed is slow. The main reason is that robots can't follow the change of environments where robots execute their task because the learning speed is slow.

In order to solve the problem, we utilized "module learning"[1][2]. Figure 1 shows the block diagram. In module learning, there are multiple modules, each of which consists of an environmental model (EM), a learning function and a policy. The policy means the map from a situation that robots face with to an action. The learning module modifies the policy. An EM has features of a specific environment. The learning process is as follows: 1. Selection a module which has an EM which shows features most similar to the current environment. 2. Action based on the policy of the module, 3. Modification of the policy by learning module based on the reaction from the current environment. 4. Addition of the module if necessary.

To improve learning capability of the conventional learning algorithm using only one module, the new learning algorithm utilizing more than two modules has been proposed. However, it is difficult for the proposed module learning to be implemented into LSI because huge memory capacity is required. There are two reasons. One (P1) is huge memory capacity required for an EM. The other (P2) is that the proposed module learning basically requires the same number of modules as that of environments. Therefore, we propose an effective algorithm to solve each of P1 and P2. The effectiveness of the proposed algorithms is quantitatively evaluated in the simulation

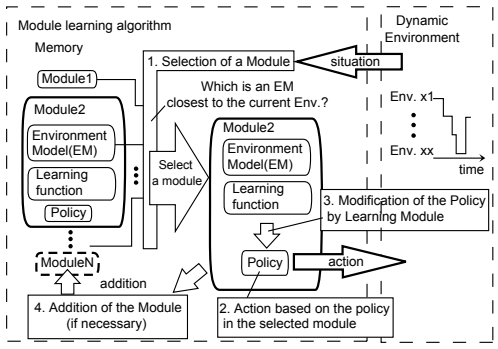


Figure 1: Schematic diagram of standard module learning.

2. A module learning algorithm using environment models with small data size

We explain the first problem (P1) in module learning in detail: an EM requires a huge memory capacity. Typically, a set of probabilities $p(\mathbf{s}(t+1) \mid \mathbf{s}(t), \mathbf{a}(t))$ is used as an EM. p means the probability that next situation $\mathbf{s}(t+1)$ appears by taking action $\mathbf{a}(t)$ on the present situation $\mathbf{s}(t)$. That is, p is unique to an environment. Hence, p expresses the features of the environment. Here, a situation and an action are expressed as a vector because a situation (an action) is expressed as a set of values detected by some sensors (outputted by some actuators). The memory capacity required for p is estimated as $N_a(2^{rNse})^2 \cdot m_p$ (N_a: the number of actions, r: sensor resolution [bit], N_{se}: the number of sensors, m_p: unit data size of p [bit]). Thus, the memory capacity for an EM becomes huge if the number of sensors and their resolution are large. Hence, the number of modules that robots can store has to be reduced because of physical limit of memory capacity as the number of sensors and their resolution are larger. However, reduction of modules makes performance worse in module learning.

2.1. Proposed algorithm

In order to solve the problem, we propose that a set of Q-functions $Q(\mathbf{s}_i, \mathbf{a}_j)$ is used, instead of p as an EM. It expresses the quality of an action $\mathbf{a}_j$ in a situation $\mathbf{s}_i$ (larger Q means that an action is better). In other words, a set of unique Q-function to an environment is decided because the quality of an action in a situation depends on an environment. If we use a set of Q as a EM, memory capacity for Q is estimated as $N_a(2^{rNse}) \cdot m_n$ (m_n : unit data size of p [bit]). The memory capacity for an EM composed of Q is equal to $(1/2)^{rNse}$ times of the one for an EM composed for p. That is, an EM composed of Q allows more modules to be store than using p. We show the comparison with Q and p about memory capacity for an EM in Figure 2 in case that resolution of each sensor is 3bits, N_a is 100 and m_p and m_n are 8 bits.

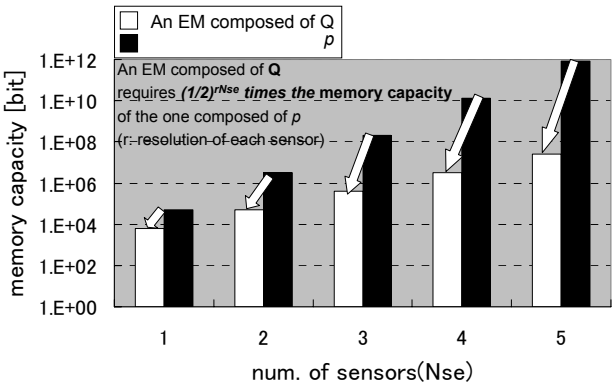


Figure 2: Comparison of memory capacity for an EM composed of Q-function and that for an EM composed of probabilities p (resolution of each sensor is 3bits, num. of actions is 100 and unit data size of Q and p is 8bits).

We explain processing procedure in our module learning algorithm with EMs composed of Q in Figure 3. We assume

that effective modules for some environment already have been obtained.

We prepare $Q_{obs}(\mathbf{s}_i, \mathbf{a}_j)$ which is Q-function, for evaluating features in the current environment. $Q_{obs}(\mathbf{s}_i, \mathbf{a}_j)$ is updated as follows: $Q_{obs}(\mathbf{s}, \mathbf{a}) \leftarrow Q_{obs}(\mathbf{s}, \mathbf{a}) + \alpha(R_w + \gamma \max_{\mathbf{a}'} Q_{obs}(\mathbf{s}', \mathbf{a}') - Q_{obs}(\mathbf{s}, \mathbf{a}))$, where $\mathbf{s}$, $\mathbf{a}$, $\mathbf{s}'$, $\mathbf{a}'$ means the last situation, the last action, the current situation and the next action to select, respectively. α and $\gamma(0< \alpha, \gamma<1)$ are called discount parameter and step size parameter. R_w is reward given by the current environment when a robot achieve to goal situation. This update rule is called Q-Learning. Q_x which is Q-function in a module is also updated in the same way(x: index of module).

The way to select a module is as follows. R_X, which is reciprocal of the difference between the updated Q_X and Q_{obs}, is calculated. The module that has the maximum value of R_x is selected. This means that the module with the closest Q_x to Q_{obs} is selected. However, if some modules have the values being close to the maximum R_x, the proper module may not be selected. So, we define another measure V_x, which is the sum of reward. It can express the justice of the selected module. When the module is selected, we consider also the measure V_x as follows. If V_x is the smallest, the module must not be selected. In order to avoid too large value of V_x, it is reset when the value is larger than a constant.

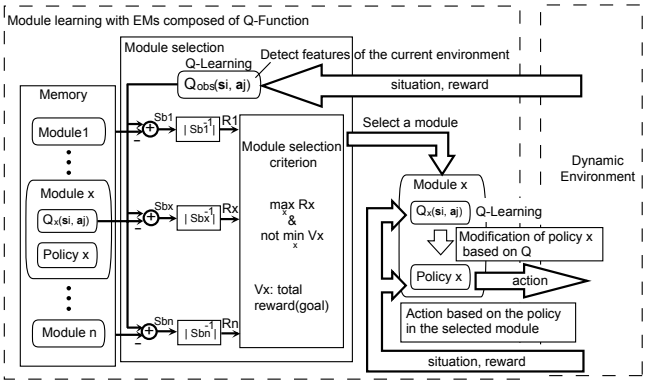


Figure 3: Processing procedure in the proposed module learning using environment models composed of Q-function.

2.2. Simulation experiment and result

In order to confirm the ability of the model, we applied it to air hockey game. The robot considers the opponent as a part of environments. The environment changes when the opponent's feature changes. In an experiment, opponent's feature has been varied by stick position as shown in Figure 4. The opponent is same program as the robot with a simple basic strategy. The strategy is fixed and is not tuned by learning. R_w is +1 or -1 when robot gets or loses a point, respectively.

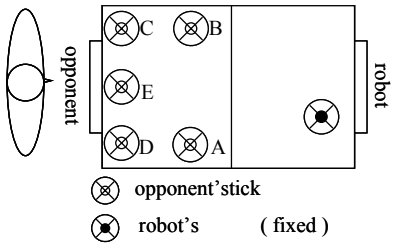
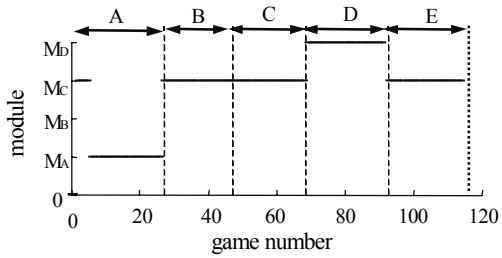
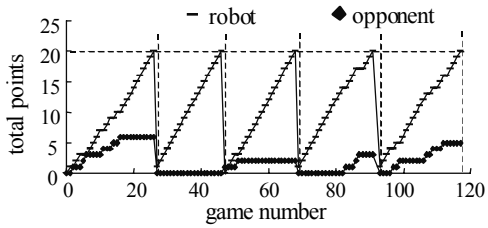


Figure 4: An example of the opponent's feature.

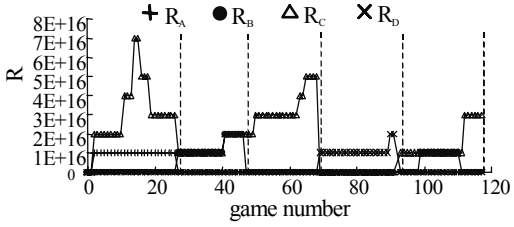
Figure 5 shows the result of the experiment. The stick position of the opponent changes from A to E (see Figure 4) every set. It is one set in 20 points. The number of stored modules is four. Before experiment, the four modules have been obtained by learning in case that the stick positions of opponent are fixed at A, B, C and D, respectively. The modules are called M_A, M_B, M_C and M_D. Figure 5(a) shows that the proper module except B was quickly selected. Here, we have considered the case of the position B. It is the second set. Figure 5(b) shows that the robot overwhelmingly won even by using the M_C. Thus, the model has also selected the proper module in this case. Next, let us consider the case of the position E. Note that no module has been previously prepared for the position E. In order to confirm the selection of M_C for the position E is appropriate, we had another experiment. In the experiment, we applied the modules from M_A to M_D to the position E. Figure 6 shows the result: The M_C is the best module because the robot overwhelmingly won as comparison with the others. From the above discussions, effectiveness of the algorithm with EM using Q-function has been confirmed apparent.



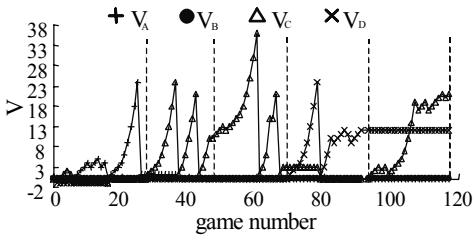
(a) A robot's selecting strategy



(b) The total of the points that the robot and opponent got



(c) Rx(the reciprocal of the difference between Q_x and Q_{obs})



(d) The evaluate function Vx

Figure 5: The result of the experiment where the opponent's stick position changes A from E by one set (=twenty points).

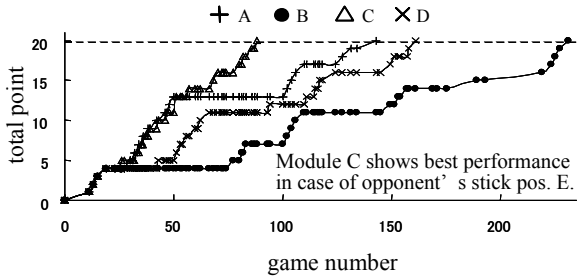


Figure 6: The total point given to the robot by using each module during the opponent's stick position E.

3. A module learning algorithm with fast learning and compact storage of modules

Another problem (P2) in module learning is that it is basically difficult for one module for over two environments. Increasing the number of environments, a large number of modules and a large memory capacity are required. For example, in *Robocup Soccer*, a hard disc memory with about 100 GB was used [5]. Especially, in autonomous robot applications, memory capacity is limited with power dissipation and physical size. It is difficult to apply to the complex real environments.

Then, we propose another advanced module based learning that reduces memory capacities. In our algorithm, one module is allocated for some environments, features of which are similar each other. Here, we call a module in this algorithm a representation module, "RM". After selection of a RM, the policy in the RM selected is modified by reinforcement learning (RL) [3] based on partial policy correction[4]. RL based on partial policy correction is one of methods to learn Q-function and much faster than Q-Learning. Based on such a concept, this module learning algorithm can reduces memory capacities and keeps high adaptation to environmental change.

The processing procedure in this algorithm is shown in Fig. 7. It is as follows: (1) detection of the current environment, (2) selection of one ~~a~~ RM from stored RMs which is suitable to adapt multiple environments, (3) modification of the policy in the selected RM by RL based on partial policy correction and (4) addition of the RM modified or elimination of it.

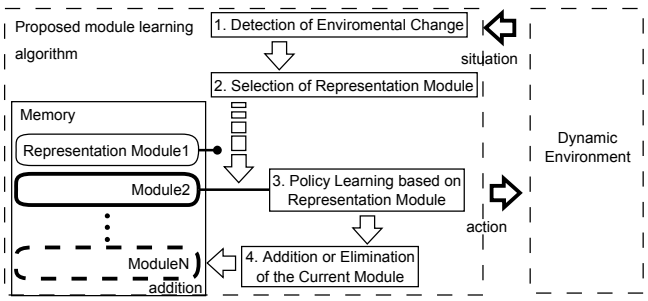


Figure 7: Processing procedure in the proposed module learning with fast learning and compact storage of modules.

3.1 Detection of the environmental change and selection of a RM

In this algorithm, we simply use the rate of achievement (R) which means how often robots have achieved the goal to detect environmental change and select a RM, instead for EM. This is because the purpose of this algorithm is not detection of the environmental change but fast learning and module storage. Hence, R is large if the current policy works effectively. If it is lower than a threshold Rg during some periods, the system selects another RM which is added more newly than the current one.

3.2. Learning of RM and addition of a new RM

The RL based on partial policy correction uses Q-Learning, RL based on partial policy correction has only one policy. (The policy means the map from a situation to an action.) It applies the policy constructed in the last environment to the current environment. When R is lower than Rg, some Q(**s**, **a**) including the main factor is corrected partially. Therefore, if both of the last and the current environments are alike, learning is very fast and moreover, the suitable policy to both environments is obtained. Consequently the memory is also saved because on policy can be utilized to at least two environments. In contrast, when they are very different, learning is very slow. In order to solve this problem, in the proposed algorithm, multiple RM, that is multiple policies can be stored.

Addition of a new RM is carried out as follows. The correlation is calculated between each memorized module and the selected one, the policy in which has just been modified by RL based on partial policy correction. If the correlation is lower than a threshold, the new one is added to the memory. On the other hand, if the correlation is higher, the new one is eliminated.

3.4. Simulation experiment and result

In order to evaluate the ability of the proposed algorithm, we applied it to Maze Problem with some simulation experiments. Maze Problem is often used as an exercise for RL. We assumed a 5x5 maze in Figure 8. The purpose of the robot is to learn the shortest path from the start to the goal. The robot can select one of four actions :{ up, down, left, right} (selecting one action is defined as one step.). If the robot takes a hundred steps or arrives at the goal, the robot is returned to the start. The reward (R_w) is +1, or 0 if the robot reach the goal or not.

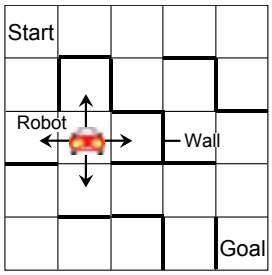


Figure 8: Maze problem (6 mazes are prepared and they change in some interval in this experiment).

In order to evaluate the learning performance of the system for environmental changes, six kinds of mazes (a-f) are used. The maze changes randomly after Nc steps are taken. In order to compare the learning capability of the proposed algorithm with conventional methods, we applied Q-Learning and RL based on partial policy correction to this maze problem.

Figure 9 shows the result during 20000 steps in case that Nc is 1000. The horizontal axis expresses steps and the vertical axis expresses the following: (a): environmental change, (b)-(d): number of steps required from the start to the current

situation with the proposed system, RL with partial policy correction and Q-Learning, respectively. In (b)-(d), as the value of the vertical axis is smaller, the system has an effective policy. While Q Learning and RL based on partial policy correction constructed the effective policy only for the maze "a", the proposed algorithm learned the multiple effective policies for almost all mazes except maze "b".

Figure 10 shows simulation results of total rewards vs. Nc in 20000 steps. From Figure 10, the performance of the proposed system is better than the other methods over all of Nc. Additionally, the system constructed only two RM over all of Nc.

Above the discussion, our module learning algorithm can use the memory effectively and keep high adaptation to environmental change.

4. Conclusion

We proposed two types of module learning algorithm for robot in order to adapt environmental change faster: 1. a module learning algorithm using environment models with small data size, 2. another module learning algorithm with fast learning and compact storage of modules for module learning. It was confirmed by the simulation experiments that both algorithm are robust to environmental change and uses its memory more effectively. As next step, we'll incorporate EM used in the first algorithm into detection of environmental change and selection of RM in the second algorithm.

References

[1] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive mixture of experts", Neural Computation, vol.3, pp.79-87, 1991.

[2] K. Doya et al., Technical report, Kawato Dynamic Brain Project Technical Report, KDB-TR-08, Japan Science and Technology Corporation, June 2000.

[3] Richard S.Sutton and Andrew G.Barto, Reinforcement Learning, MIT Press, 1998.

[4] T. Minato et al., Journal of the Robotics Society of Japan, Vol. 18, No. 5, pp. 706-712, 2000.

[5] Yasutake Takahashi, Kazuhiro Edazawa, and Minoru Asada. Modular Learning System and Scheduling for Behavior Acquisition in Multi-Agent Environment. RoboCup 2004 Symposium papers and team description papers.

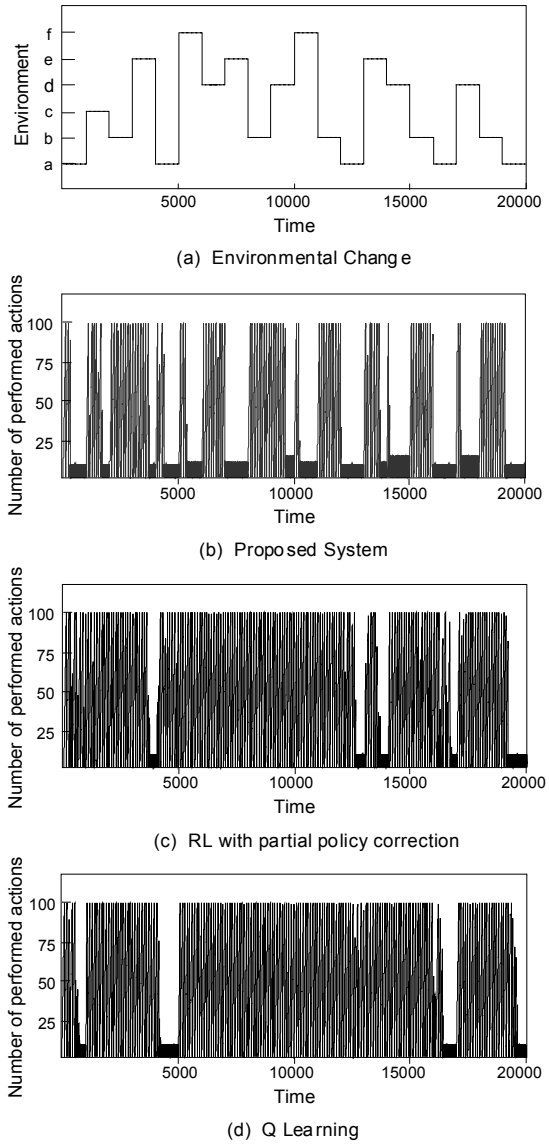


Figure 9: Result of the simulation experiment 1: the learning performance during 20000 steps (the environment changes randomly after the robot experience 1000 steps, (a): proposed algorithm, (b) RL based on partial policy correction (c): Q-Learning)

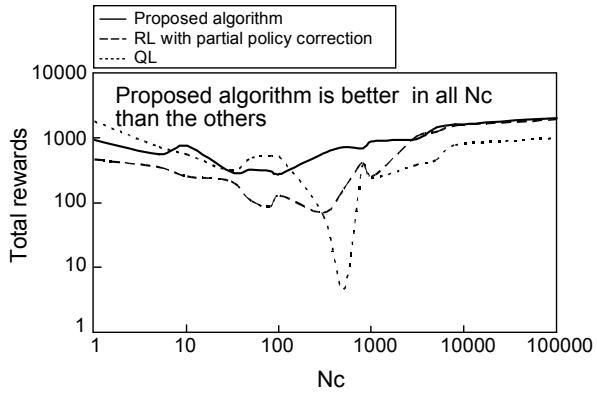


Figure 10: Result of the simulation experiment 2: Total rewards vs. Nc in 20000 steps (a maze changes randomly after the robot experiences Nc steps).