A Hierarchical Multi-module Learning System Based on Self-interpretation of Instructions by Coach

Yasutake Takahashi[1,2], Koichi Hikita[1], and Minoru Asada[1,2]

[1] Dept. of Adaptive Machine Systems, Graduate School of Engineering
[2] Handai Frontier Research Center
Osaka University
Yamadagaoka 2-1, Suita, Osaka, Japan
{yasutake,khikita,asada}@er.ams.eng.osaka-u.ac.jp
http://www.er.ams.eng.osaka-u.ac.jp

Abstract. We propose a hierarchical multi-module leaning system based on self-interpretation of instructions by coach. The proposed method enables a robot to decompose (i) a long term task which needs various kinds of information into a sequence of short term subtasks which need much less information through its self-interpretation process for the instructions given by coach, (ii) to select sensory information needed to each subtask, and (iii) to integrate the learned behaviors to accomplish the given long term task. We show a preliminary result of a simple soccer situation in the context of RoboCup.

1 Introduction

Reinforcement learning (hereafter, RL) is an attractive method for robot behavior acquisition with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors [2]. However, single and straightforward application of RL methods to real robot tasks is considerably difficult due to its almost endless exploration which is easily scaled up exponentially with the size of the state/action spaces, that seems almost impossible from a practical viewpoint.

Fortunately, a long time-scale behavior might be often decomposed into a sequence of simple behaviors in general, and therefore, the search space is expected to be able to be divided into some smaller ones. Connell and Mahadevan [3] decomposed the whole behavior into sub-behaviors each of which can be independently learned. However, task decomposition and behavior switching procedure are given by the designers. Takahashi and Asada [4, 5] proposed a multi-layered RL system. The modules in the lower networks are organized as experts to move into different categories of sensor output regions and learn lower level behaviors using motor commands. In the meantime, the modules in the higher networks are organized as experts which learn higher level behaviors using lower modules. However, this system tends to produce not only purposive behavior learning modules but also many non-purposive ones, and as a result, to require large computational resources.

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 576–583, 2004.
© Springer-Verlag Berlin Heidelberg 2004

When we develop a real robot which learns various behaviors in its life, it seems reasonable that a human instructs or shows some example behaviors to the robot in order to accelerate the learning before it starts to learn. Whitehead [6] showed that instructions given by coach significantly encourages the learning and reduces the learning time. This method, called LBW (Learning By Watching), reduces the exploration space and makes learner have experiences to reach the goal frequently. Asada et al. [1] proposed a method, called LEM (Learning from Easy Mission). The basic idea is that a learning scheduling such as a robot starts to learn in easy situations to accomplish a given task at the early stage and learns in more difficult situations at the later stage accelerates learning the purposive behaviors. They applied this idea to a monolithic learning module. In order to cope with more complicated tasks, this idea can be extended to a multi-module learning system. That is, the robot learns basic short term behaviors at the early stage and learns complicated long term behaviors at the later stage based on instructions given by coach.

In this paper, we propose a behavior acquisition method based on hierarchical multi-module leaning system with self-interpretation of coach instructions. The proposed method enables a robot to

1. decompose a long term task into a set of short term subtasks,
2. select sensory information needed to accomplish the current subtask,
3. acquire a basic behavior to each subtask, and
4. integrate the learned behaviors to a sequence of the behaviors to accomplish the given long term task.

We show a preliminary result applied to a simple soccer situation in the context of RoboCup.

2 A Basic Idea

There are a learner and a coach in a simple soccer situation (Fig. 1). The coach has *a priori* knowledge of tasks to be played by the learner. The learner does not have any knowledge on tasks but just follows the instructions. After some instructions, the learner segments the whole task into a sequence of subtasks, acquire a behavior for each subtask, find the purpose of the instructed task, and acquire a sequence of the behaviors to accomplish the task by itself. It is reasonable to assume that the coach will give instructions for easier tasks at the early stage and give ones for complicated tasks at the later stage although it does not have any *a priori* knowledge about the learning system on the agent.

Fig. 2 shows a perspective of development of the learning system through instructions given by coach at three stages. When the coach gives new instructions, the learner reuses the learning modules for familiar subtasks, generates new learning modules for unfamiliar subtasks at lower level and a new module for a sequence of behaviors of the whole instructed task at the upper level. After the learning at one stage, the learner adds newly acquired learning modules to the learning module database. The learning system iterates this procedure from easy tasks to more complicated ones.

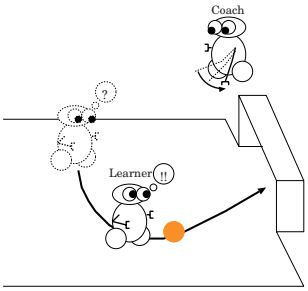


Fig. 1. Basic concept: A coach gives instructions to a learner. The learner follows the instruction and find basics behaviors by itself

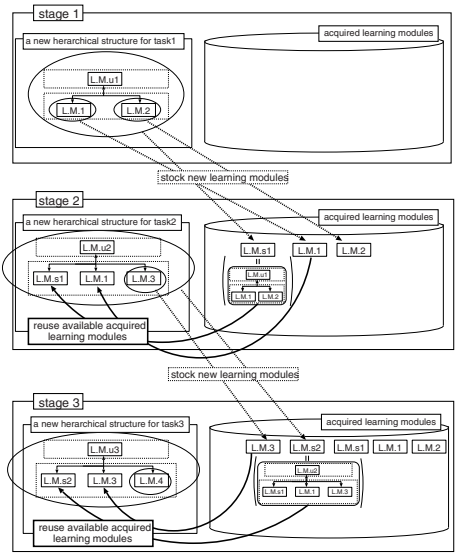


Fig. 2. A perspective of of development of the learning system with staged instructions

3 Hierarchical Multi-module Learning System

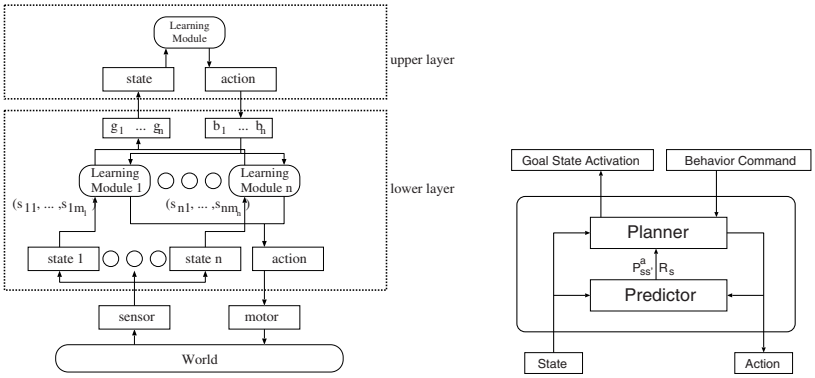
3.1 An Architecture

The basic idea of multi-layered learning system is similar to [4, 5]. The details of the architecture has been extended. The robot prepares learning modules of one kind, makes a layer with these modules, and constructs a hierarchy between the layers. The hierarchy of the learning module's layers can be regarded as a role of task decomposition. Each module has a forward model (predictor) which represents the state transition and reward models ($\hat{\mathcal{P}}^a_{ss'}$, $\hat{\mathcal{R}}^a_{ss'}$), and a behavior learner (policy planner) which estimates the state-action value function ($Q(s, a)$) based on the forward model in an RL manner (Fig. 3(b)). The state and the action are constructed using sensory information and motor command, respectively at the bottom level.

The input and output to/from the higher level are the goal state activation and the behavior command, respectively, as shown in Fig. 3. The goal state activation g is a normalized state value, and $g = 1$ when the situation is the goal state. When a module receives the behavior command b from the higher modules, it calculates the optimal policy for its own goal, and sends an action command to the lower module. The action command at the bottom level is translated to an actual motor command, then the robot takes an action in the environment.

3.2 A Learning Procedure

The steps of the learning procedure are as follows:



(a) whole system (b) module

Fig. 3. A multi-layered learning system

1. Coach instructs some example behaviors to accomplish a task.
2. Learner evaluates the availability of learned behaviors to accomplish the task by watching the examples.
3. The learner segments the task into subtasks, and produces new learning modules at the lower layer if needed, and learns the behavior for each.
4. The learner produces a learning module at the higher layer and learns the whole behavior to accomplish the task.
5. Go to step 1.

3.3 Availability Evaluation

The learner needs to evaluate the availability of learned behaviors which help to accomplish the task by itself because the coach neither knows what kind of behavior the learner has already acquired directly nor shows perfect example behavior from the learner's viewpoint. The learner should evaluate a module valid if it accomplishes the subtask even if the greedy policy seems different from the example behavior. Now, we introduce $\overline{Q}$ in order to evaluate how suitable the module's policy is to the subtask as follows:

$$\overline{Q}(s, a_e) = \frac{Q(s, a_e) - min_{a'}\, Q(s, a')}{max_{a'}\, Q(s, a') - min_{a'}\, Q(s, a')}\ , \tag{1}$$

where a_e indicates the action taken in the instructed example behavior. $\overline{Q}$ becomes larger if a_e leads to the goal state of the module while it becomes smaller if a_e leaves the goal state. Then, we prepare a threshold $\overline{Q}_{th}$, and the learner evaluates the module valid for a period if $\overline{Q} > \overline{Q}_{th}$. If there are modules whose $\overline{Q}$ exceeds the threshold $\overline{Q}_{th}$ simultaneously, the learner selects the module which keeps $\overline{Q} > \overline{Q}_{th}$ for longest period among the modules (see Fig. 4).

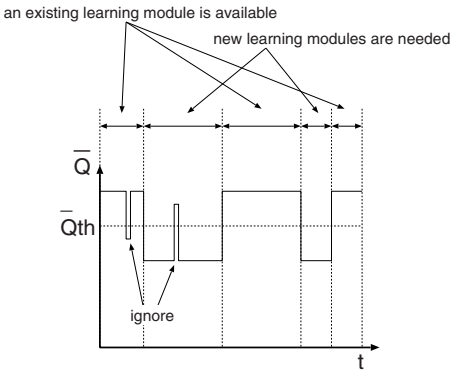


Fig. 4. Availability identification during the given sample behavior

3.4 Producing New Learning Modules

If there is no module which has $\overline{Q} > \overline{Q}_{th}$ for a period, the learner creates a new module which will be assigned to the not-learned-yet subtask for the period. In order to assign a new module to such a subtask, the learner identifies the state space and the goal state. The following shows the steps briefly.

1. Prepare a set of state spaces $\boldsymbol{S}$ and, set their priorities as $\boldsymbol{S}_i : i = 1, 2, \cdots$.
2. For each state space $\boldsymbol{S}_i$,
   (a) Estimate a goal state space $\boldsymbol{G}$ in the state space $\boldsymbol{S}_i$ based on the instructed example behaviors.
   (b) If the estimated goal state space $\boldsymbol{G}$ covers all of the state space $\boldsymbol{S}_i$, increment i and goto step (a).
   (c) Construct a learning module and calculate Q values.
   (d) Check the performance of the learned behavior for the subtask. If the success rate is low, increment i and go to step (a).
3. Add a new module based on the state space $\boldsymbol{S}_i$ and the goal state space $\boldsymbol{G}$.
4. Check the availability of modules over the given task. If there is a period where there is no available module, go to step 1.
5. Exit.

State Variables Selection. We introduce heuristics and set priorities to the set of state spaces as follows:

1. Only a few state variables are needed for all subtasks even if large number of state variables are necessary for the whole task: We limits the number of variables to only three in this study.
2. Higher priority is assigned to the state variable which changes largely from the start to the end during the example behaviors because it can be regarded as an important variable to accomplish the subtask.

3. Higher priority is assigned to the state space which has smaller average of entropy $H(s,a)$ (see equation 2) of the state transition probability $P^a_{ss'}$ for the experienced transition. The reason is that the learning module acquires a more purposive behavior with more stable state transition probability which has lower entropy.

$$H(s,a) = -\sum_{s' \in S} P^a_{ss'}(s,a,s') log_2 P^a_{ss'}(s,a,s') \tag{2}$$

Goal State Space Selection. It is hard to specify the goal state of the subtask with limited number of experiences of example behaviors. We need other heuristics here.

- A state variable of the goal state tends to be the maximum, the minimum, or the medium.
- If the value of a variable has no consistent one at the terminate state of the example behavior, the variable is independent of the goal state.

The system produce a reward model based on these heuristics.

3.5 Learning Behavior Coordination

After the procedures mentioned above, there should be necessary and sufficient modules at the lower layer, then the learning system puts a new learning module at the upper layer, and the module learns to coordinate the lower modules. The upper module has a state space constructed with the goal state activations of the lower modules. A set of actions consists of the commands to the lower modules.

4 Experiments

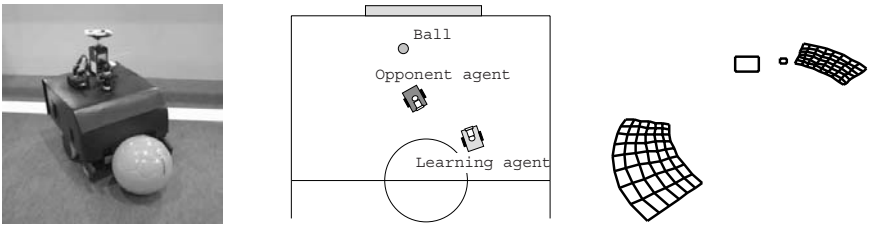
4.1 Setting

Fig. 5 (a) shows a mobile robot we have designed and built. The robot has an omni-directional camera system. A simple color image processing is applied to detect the ball area and an opponent one in the image in real-time (every 33ms). Fig. 5 (b) shows a situation with which the learning agent can encounter and Fig. 5 (c) shows the simulated image of the camera with the omni-directional mirror mounted on the robot. The larger and smaller boxes indicate the opponent and the ball, respectively. The robot has a driving mechanism, a PWS (Power Wheeled Steering) system.

4.2 Learning Scheduling and Experiments

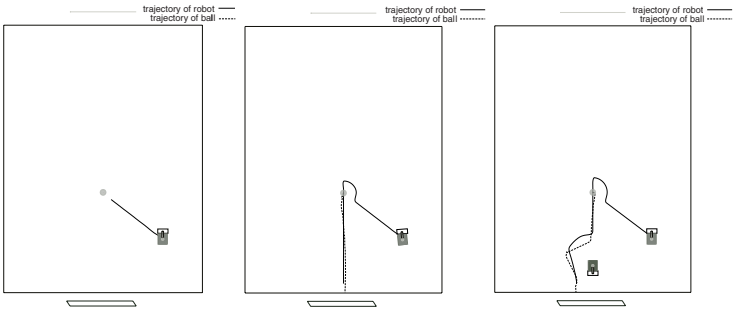
The robot receives instructions for the tasks in the order as follows:

Task 1: ball chasing
Task 2: shoot a ball into a goal without obstacles
Task 3: shoot a ball into a goal with an obstacle



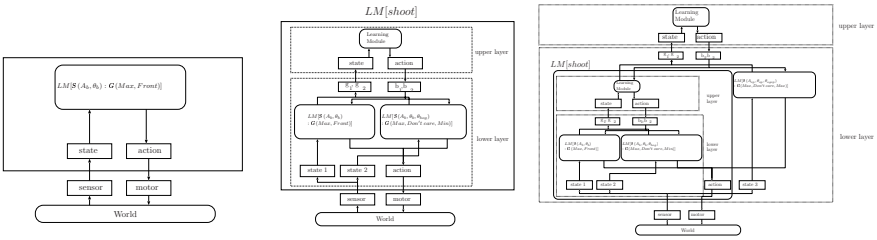
(a) A real robot and a ball (b) Top view of the field with a omni-directional mirror (c) Simulated camera image

Fig. 5. Real robot and simulation environment



(a) Task 1 (b) Task 2 (c) Task 3

Fig. 6. Example behaviors for tasks



(a) Task 1 (b) Task 2 (c) Task 3

Fig. 7. The acquired hierarchical structure

Figs. 6 (a), (b), and (c) show the one of the example behaviors for each task. Figs. 7 show the constructed systems after the learning of each task. First of all, the coach gives some instructions for the ball chasing task. According to the learning procedure mentioned in **3**, the system produce one module which acquired the behavior of ball chasing. At the second stage, the coach gives some instructions for the shooting task. The learner produces another module which has a policy of going around the ball until the directions to the ball and the goal become same. At the last stage, the coach gives some instructions for the

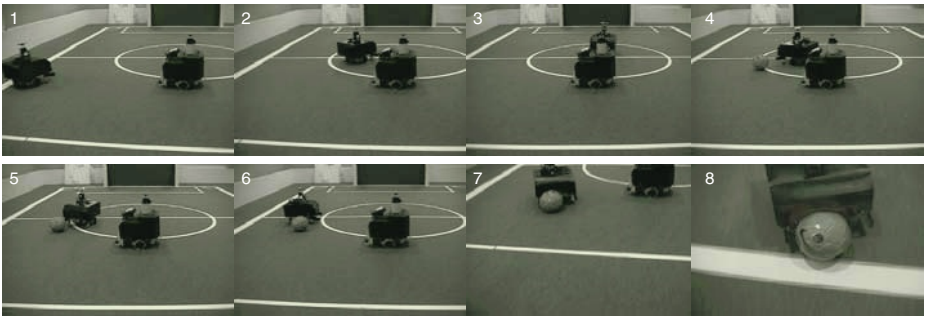


Fig. 8. A sequence of an experiment of real robots (task3)

shooting task with obstacle avoidance. The learner produces another module which acquired the behavior of going to the intersection between the opponent and the goal avoiding the collision. Fig.8 shows a sequence of an experiment of real robots for the task.

References

1. M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Vision-based reinforcement learning for purposive behavior acquisition. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 146–153, 1995.
2. Jonalthan H. Connell and Sridhar Mahadevan. *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.
3. Jonalthan H. Connell and Sridhar Mahadevan. *ROBOT LEARNING*, chapter RAPID TASK LEARNING FOR REAL ROBOTS. Kluwer Academic Publishers, 1993.
4. Y. Takahashi and M. Asada. Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 395–402, 2000.
5. Y. Takahashi and M. Asada. Multi-controller fusion in multi-layered reinforcement learning. In *International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI2001)*, pages 7–12, 2001.
6. Steven D. Whitehead. Complexity and cooperation in q-learning. In *Proceedings Eighth International Workshop on Machine Learning (ML91)*, pages 363–367, 1991.