Modeling of autonomous problem solving process by dynamic construction of task models in multiple tasks environment

Yu Ohigashi

Graduate School of Information Science, Hokkaido Univ.

Kita 14 jyou Nishi 9 chome, Kita, Sapporo, Hokkaido, 060–8628, Japan

y_ohigashi@complex.eng.hokduai.ac.jp

Takashi Omori

Tamagawa University Research Institute, Tamagawa University

Tamagawagakuen 6-1-1, Machida, Tokyo, 194-8610

omori@lab.tamagawa.ac.jp

Abstract

Traditional Reinforcement Learning (RL) supposes a complex but single task to be solved. When a RL agent faces a task similar to a learned one, the agent must relearn the task from the beginning because it doesn't reuse the past learned results. This is the problem of quick action learning, which is the foundation of decision making in real world. In this paper, we suppose agents that can solve a set of tasks similar to each other in multiple tasks environment, where we encounter various problems one after one, and propose a technique of action learning that can quickly solve similar tasks by reusing previously learned knowledge. In our method, a model-based RL uses a task model constructed by combining primitive local predictors for predicting task and environmental dynamics. To evaluate the proposed method, we performed a computer simulation using a simple ping-pong game with variations.

keywords: Model-based reinforcement learning; Multiple tasks; Task model; Reuse of knowledge

1 Introduction

In the field of machine learning, various learning methods have been proposed for each learning task to enable effective learning even for complex tasks. However, for each learning task, the setting of learning procedures, including input information, a priori knowledge, and the procedure are decided and embedded in the learning program by human engineers depending on the current task's requirements, not autonomously by the learning machine.

As researchers tend to work to solve a current problem, many machine learning models are assumed to be applied to difficult but single tasks. This causes the resulting knowledge from task learning to be specialized to the task and prevents knowledge acquired from similar but different tasks to be reused in new learning task situations. But in the real world we encounter many different tasks one after one and the exact same task or situation rarely happens: this is the definition of a multiple tasks environment. Though people may feel that reusing an experience in task solving is easy for different tasks, human engineers are actually reusing their own experiences, not the learning machine. Of course occasionally acquired knowledge can be reused, but it usually requires the intervention of human engineers who design and set the reuse process. In principle, current machine learning methods themselves are relatively inflexible for multiple task learning.

These considerations suggest that the important factors that enable application of a learning model to various learning tasks are the analysis of target tasks and the setting of learning procedures by human engineers before learning. Then it is humans who really exhibit real intelligence in divergent process of problem solving and current learning machines are not so flexible for autonomous setting to new problems.

However, regarding possible future situations, where intelligent machines such as robots will play active roles in everyday life, robots will probably encounter various learning tasks in a changing real world for which they are unprepared. Under these conditions, a robot has to autonomously decide what and how to learn and then conduct the learning. Rather than making learning models more efficient for a single task, it becomes crucial to develop a method that autonomously finds and sets learning procedures for new tasks in the multiple tasks environment.

In contrast, humans can choose suitable actions when facing various new tasks in their daily lives. In particular, humans exhibit very quick learning ability for tasks similar to past experiences. This ability reflects the results of human functions that find and identify learning tasks in everyday life, decide learning methods for them, and quickly setup learning/action-decision brain networks based on past experiences

autonomously. We call this "task setting ability."

We will realize an intelligent learning machine in multiple tasks environment when we can clarify information processing of human task setting and embed the process in machine learning algorithms. In this study, we aim to achieve a learning machine that autonomously sets up its learning tasks and quickly learns its actions for new tasks that resemble past experiences.

We approach this problem through a framework of model-based reinforcement learning (MBRL), which is a kind of reinforcement learning that realizes effective action learning using a model of environmental dynamics. The environmental model has such functions as next state prediction from current state and action and prediction of reward acquisition for a given state. In MBRL the construction of an environmental dynamics model necessary for task solving shares major parts of the task setting process.

In conventional MBRL methods, the environmental dynamics model is learned slowly through a large amount of interactive experiences with its environment or is given by human engineers who already got it by another method. The latter is beyond the scope of this study. The former requires a long learning time with conventional methods. So an autonomous and quick task setting method that reuses knowledge from past similar experiences is a fundamental solution to the problem.

In the field of multiple task learning in robotics, some people advocate autonomous task representation setting through experiences [S. Thrun and L. Pratt 1998; J. Weng et al. 2000; K. C. Tan et al. 2005]. For such realization, methods are proposed in which a robot acquires behavior knowledge through a set of similar tasks and reuses the knowledge for other similar new tasks [S. Thrun and J. O'Sullivan 1996; S. Thrun and L. Pratt 1998; J. Weng et al, 2000; K. C. Tan et al. 2005]. In these methods, tasks are classified into clusters based on their similarity. For all clusters, a set of proper features are defined, extracted, and used for action learning, and the learning result is reused for new tasks belonging to the same cluster. But since this knowledge is dependent on both the task and the learning algorithm, its applicable range is narrow. Though those researches succeeded in multiple task learning, their methods don't contain effective techniques for task setting when viewed from the human task setting ability being discussed in this paper.

Mixture of experts model [R.A.Jacobs and M.I.Jordan et al. 1991] is known as a learning system using modular architecture, and has been applied to nonlinear or nonstationary control tasks. However, success of such modular architecture depends strongly on a capability of gating network to decide which of given modules should be recruited at any particular moment. In general, gating network requires long learning time to acquire all of necessary input-output relations, and is difficult to realize human like quick

adaptation. MOSAIC model [D.M.Wolpert and M.Kawato. 1998; M.Haruno and D.M.Wolpert 2001] is composed of multiple modules in which a forward model and an inverse model are forming a unit. This system decides output of each module and learns each module in proportion as a prediction error of the forward model. MOSAIC model is categorized into supervised learning that desired output is given in advance. However, in many problems of the real world, desired input-output relation is often unknown. The multiple model-based reinforcement learning (MMRL) [K.Doya and K.Samejima et al. 2002] can be applied for this case. MMRL is composed of multiple prediction models and reinforcement learning controllers. MMRL adaptively switches and combines pairs of prediction model and controller based on prediction error of prediction models. However, all state variables are used for input of the prediction model, and it causes difficulty for reusing the prediction models to other tasks in which environment and state variables change. Additionally, MOSAIC and MMRL deal with problems of motor control but we deal with more cognitive problem solving situations.

In this paper, we propose a new method of autonomous and quick task model construction for new problems based on reusing knowledge from past experiences. The task model is a model of environmental dynamics in MBRL that can be used to predict the change of environmental status from a current one to a rewarded one. By using the task model, we can predict how the task state develops from an action in the current state and predict potential reward acquisition before the action execution. Its use in MBRL results in drastic acceleration of learning.

To construct the task model, we must find causalities, such as the physical rules for objects and state and action relations for reward acquisition, embedded in the task and represent them as a reusable form of knowledge. For that purpose, we assumed the following two hypotheses about task environments: (a) Regularity exists based on physical laws and/or causalities between variables representing the task environment. (b) Possible variations of tasks share those regularities.

Based on these hypotheses, we assumed that humans construct the task model for a task by following procedures and realize action learning using them. (1) Perceive sensory information from the current task scene and extract possible features. (2) Select a subset of the features necessary for task processing. (3) Find regularity and causality between the features. (4) Construct the task model by combining them online. (5) Learn proper actions for the task using the task model.

The following two information processing functions were necessary to realize these procedures in a learning program. (i) Finding local regularity between variables in each task scene and setting prediction

learning between them. (ii) Construction of the task model representing the entire task process by a dynamic combination of those local predictions.

We used a local prediction model to represent causalities between subsets of variables. In our study, the knowledge of causal relations in the environment is acquired through local prediction learning and is accumulated in as a local predictor corresponding to each causality. In our model, each local prediction model is composed of a Predictor for local prediction and a Selector that encodes the Predictor's application conditions. After the Selector and Predictor sets are learned, we can select applicable knowledge by observing the current environmental state and predict changes in the state by applying local prediction models. The system automatically selects and combines the local prediction models bottom-up and dynamically predicts the state of task related part of the environment. When Predictors and Selectors are shaped by previous experience, this process is the autonomous construction of the task model for new task reusing knowledge of an environment.

In this study, we conducted a computer simulation on simple TV games that included causal relations and evaluated the effectiveness of the proposed method. In the experiments, the system first acquired knowledge of regularities in the task environment by learning a standard game. Next, the system tackled action learning on variations of the standard game. Through the experiments, we expected the appearance of the quick construction of task models and quick action learning of a new task by the reuse of knowledge. Though this experiment is merely a case study on a toy environment, it still poses a challenging problem whose realization is difficult without human intervention for current machine learning task setting strategies.

In this paper, explanations proceed as follows. In section two, we give a conceptual explanation of the proposed method. After a detailed description of its implementation in section three, we show computer simulation results in section four. Then we discuss the results in section five and conclude this study in section six.

2 Dynamic construction of task models in reinforcement learning

2.1 Structure of model-based reinforcement learning

Model-based reinforcement learning (MBRL) accelerates the learning of simple reinforcement learning (RL) using simulated virtual action and learning through an internal environment model in addition to the usual RL in which an agent actually interacts with its environment. The simulation function can be realized by

adding a prediction model of an environment to the components of a simple RL.

Figure 1 shows the basic structure of DYNA architecture as a typical example of MBRL (Sutton 1990; Sutton 1991). In DYNA, two RL methods are included in parallel. One is a direct action and experience based RL, as shown on the left side of Figure 1. Another is an RL using virtual action and simulated experience by an environmental model on the right side of Figure 1. By learning the environmental model through direct experience, the agent begins to use the simulated RL, which reduces the number of actual interactions with the real environment that are time-consuming and high in processing load costs. It enables quicker learning and simplified application of RL to the real world

The environmental model in DYNA is learned through direct interaction with the environment. Consider environmental model acquisition in maze tasks that are often used as a benchmark of RL. As the initial state, the agent doesn't have any knowledge about the environment. It tries an action based on the current state and perceives the next state and the acquired reward caused by the action. By using conventional RL, the agent evaluates the expected reward value of the action from the actual reward and decides its action with the value. Parallel to the actual RL process, the agent memorizes its experience as a set of Current state, Applied action, Next state, Acquired reward. Through trial and error in the RL, it accumulates sufficient information in its environmental model so that it can work as an environment map and be used as an environmental model. Next, the agent tries a simulation based state update and action search using the environmental model to realize action value evaluation without actual interaction with the physical environment.

In DYNA architecture, the agent stores the current and next states of its environment as all channels of the sensory information it perceives, or it is given. Then, the accumulated knowledge of the environment is difficult to reuse when the agent encounters new but similar tasks in which channels of sensory information are partially modified. Sensory information for RL is often modified depending on the task. It causes changes in the state space and corresponding learned RL results that directly depend on the given sensory channels and environment. In this case, the agent has to reconstruct the state space and relearn the entire environmental model for the new task. When the state space is large, learning of the environmental model itself takes a long time, and DYNA is actually not applicable because the environmental model just covers the part of state space that the agent actually experienced. Realization of such deft adaptation as humans in multiple tasks environment is not easy using an environmental model construction method with immediate experience and accumulation method like DYNA.

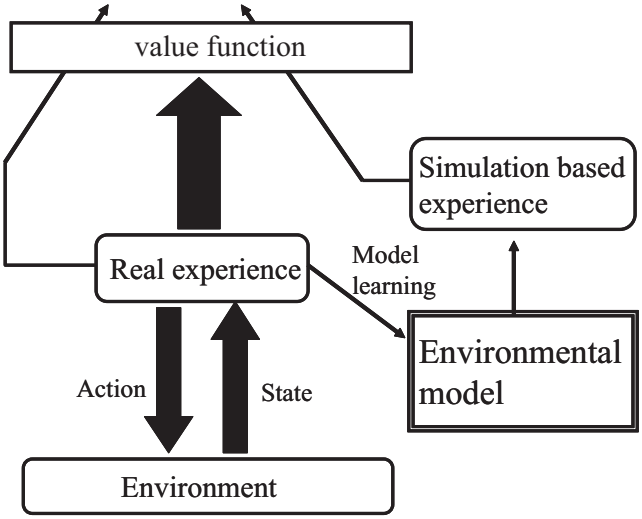

Figure 1: DYNA architecture: example of model-based reinforcement learning. Two RL methods are included in parallel. One is direct action and experience based RL, as shown on the left of Figure 1. Another is an RL using virtual action and simulated experience by environmental model on the right.

Situated in this state of the MBRL study, many past MBRL researches focused on the simultaneous learning of environmental model and their action, or on its effective use for the acceleration of RL learning. But to acquire the environmental model itself, most used conventional methods or given by other methods, not by experience based learning. However, quick construction of an environmental model is one unavoidable yet critical problem for intelligent machines behaving in the real world where environment changes dynamically and multiple tasks exist.

Therefore, in this study, we discuss a method that quickly and autonomously acquires an environmental model for MBRL. For an RL method that exploits environmental models, we use a conventional one because known methods are sufficiently effective.

2.2 Dynamic construction of environmental model by knowledge reuse in a ping-pong game

To consider environmental model acquisition for MBRL, we use a simple ping-pong game (Fig. 2) that features a ball and a paddle in the play panel and rewards for players who return the ball by controlling a paddle's motion.

For this game a typical strategy acquired by RL machines is the left or right paddle motion that chases the ball's current horizontal position. But learning takes a long time because the game's state space is

large, including the variables of the ball's 2D position, speed and motion direction, and paddle position. Of course, human intervention is necessary to set the state space, actions, and learning procedures.

In contrast, humans play the game well after observing or practicing it for a short time. In many cases, human strategy involves waiting for the ball by moving the paddle to its predicted trajectory. Within the process, human brain must analyzing the game and seeking a strategy. The brain process should be very similar with the brain process when the same person is setting RL processing for the game learning program. However, it is evident from observed behavior that human action decision strategy differs from RL.

Using predictions for the current state is a big difference between RLs and humans. People have talent of insights to find possible strategies for reward acquisition in a short time task analysis and discover a strategy of moving the paddle to the ball's anticipated falling position, predicted by applying physical laws. Humans construct a neural network that actually conducts the learning procedure in his/her brain based on the discovered strategy. Humans become able to play the game without conscious intervention in the action decision process: automatization.

An RL can also learn the same behavior with prediction based action decisions of humans when tuning RL parameters. But such a result of optimization after numerous trials and errors is not a result of strategy discovered by task analysis and insight as humans do.

In this paper, we model the task analysis and learning procedure setting ability shown by human behavior when learning new tasks. Humans find the process of action learning and generation for problem solving based on causal insights within the task acquired through observation and analysis of the task. We assume that the prediction of target tasks by humans occupies an important role in the process. Humans construct Predictors in their brains by finding causalities through analysis. By observing the effects of virtual action on Predictors, they discover the relationship between actions and reward and a proper action strategy from them. Of course, predicting a phenomenon unrelated to rewards has no connection to strategy finding. What is necessary is a prediction starting from the current state that moves to the reward acquisition state.

In this study's ping-pong game, the necessary factors for reward prediction include such physical laws of the ball as moving straight ahead or deflection off the wall, laws of environmental state change by agent action, for example, a paddle moving rule by actions that control left or right, and knowledge about the state under which rewards are given. As long as we manipulate the ping-pong game and its variations, we can obtain prediction ability rather easily by combining such knowledge.

After learning the standard game shown in Fig. 2, we learn its variations more quickly than the first

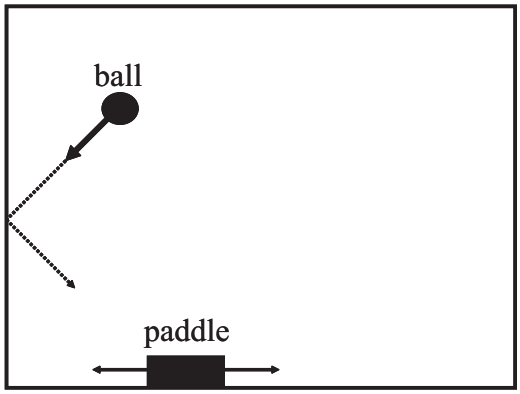

Figure 2: The ping-pong game used for this study. There are a ball and a paddle in play panel and a player gets a reward by returning the ball by controlling the paddle's motion.

game. This phenomenon can be modeled as the effect of reusing local causality knowledge, which, as the ball or paddle motion or reward conditions, is acquired by learning the first standard game. The knowledge is reused in variation games to construct a prediction model without learning.

But knowledge from the first task by a conventional RL method may not be applied to the next task. Humans may reuse it when extracting the reusable parts of the knowledge from the first one and adapting them to the new task. But the work is not easy even for humans; the knowledge differs from task to task, and the adaptation process requires task analysis and is time-consuming. After all, current learning machines are inferior to humans in the ability to quickly adapt to new but similar tasks by applying past experiences.

In this study, we focus on the human abilities of task analysis, prediction and learning procedure setting, and then try modeling. We also evaluate the model by computer simulations. The ability corresponds to the quick and autonomous construction of environmental models in MBRL. We call the task predicting environmental scheme a "task model." In the next section, we give a more detailed explanation of a task model in MBRL.

2.3 What is the task model?

The task model we propose is for task state prediction from the current to the rewarded state that functions as a task simulator. It is constructed online by a combination of local causality representing submodules selected from a pool of submodules and knowledge by observing the actual phenomena of the target world.

Figure 3 shows the relation between the usual RL part and the proposed task model. The task model receives the current state of the task and the agent action, predicts the next state of the task by dynamically

selecting local causality from a set of knowledge, and realizes state prediction until reward acquisition. Then the task model outputs the predicted reward and the corresponding predicted state of the task to the RL part. Then the RL part learns the proper action by TD learning with those predicted rewards and states.

The local causality relations included in the ping-pong game (Fig. 2) are: (1) such physical laws of the ball's motion as straight moving and wall deflection; (2) paddle motions following agent action; and (3) conditions for rewards. The actual form of these causality relations are represented as regularities between a subset of variables that represent the task state. We assume the relation can be learned and represented as a Predictor between related variables. First, the task model is given state [Ball_x_position, Ball_y_position, Ball_moving_direction, Ball_moving_speed, Paddle_position], and agent action [Left, Right, Stay] at that moment. Next, the task model predicts the game state in one time step by selecting one Predictor using the Selectors. The predictor selection of the moment and one step prediction are iterated on the current predicted state until the predicted state reaches the rewarded state. Finally, the task model outputs the predicted reward and the ball's state to the RL part.

In the RL part, the agent learns its action for task solving using the predicted results from the task model. It learns the paddle's action using an actor-critic learning method in [Current_ball_state, Current_action, Predicted_reward, Predicted_ball_state] state space. Its learning is different from usual actor-critic methods because it uses a multiple steps forward predicted state instead of a single step future state. Previously, we showed that using the multiple steps forward predicted state realized drastic acceleration of action learning in the ping-pong game (Ohigashi et al. 2003).

A large effect on learning performance by combining task model and RL has been shown in many past MBRL studies. However, in most of them, the task model is realized by analysis of human engineers and embedded in RL by hand. As far as we continue this strategy of solving problems by hand, we will eternally have to continue the setting work of the learning procedure every time we encounter a new problem. If we achieve a learning system that can autonomously construct the task model, we may take a large step toward an actual intelligent system that resembles humans. What kind of framework and function are necessary for such achievement? We explain our model in the next section.

2.4 Construction of task models

For the autonomous construction of a task model, we must find and represent knowledge about the task environment. For that purpose, we assume the following two constraints on the real world: (a) There are

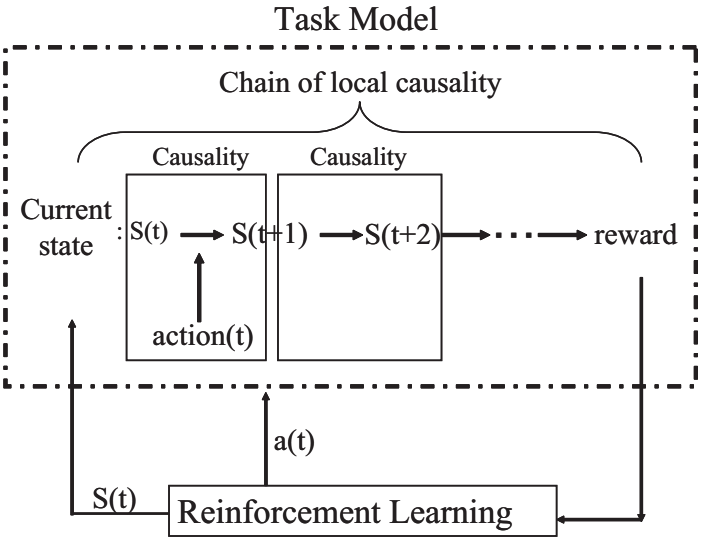

Figure 3: Task model that represents causal relations by a set of local predictors. Given a task model, we can predict acquisition of reward starting from a current task state and an action sequence.

local regularities between the task representing variables posed by physical laws and causalities. (b) These local regularities are shared by similar but different tasks. Now the task environment can be represented by a set of local physical laws, the causalities for reward acquisition, and the relations between the agent and the environment, and each task is composed by a combination of these regularities(Fig. 4).

To realize the autonomous setting of learning procedure and quick, immediate adaptation to various tasks in this local regularity dominated world, we introduce the following two functions: (i) Finding the local causal relations between variables and learning predictions between them. (ii) Construction of the entire task prediction model, the task model, by the dynamic combination of predictors.

As the concrete form of knowledge representation for regularities, we used the local prediction of variables by other variables within the task representing variable set, the Predictor. Since the agent only observes the world by perceived value, causality or the physical laws representing regularity must be constructed through perception. A Predictor that receives variables to predict other causally related values is a possible candidate for knowledge representation. Multiple Predictors correspond to various regularities in the world. To realize the task model, we prepared three types of Predictors: those that predict the automatic state changes of the environment, those that predict rewards, and those that predict the state changes caused by the actions of the agent.

Each of these Predictors was paired with a Selector that predicts the occurrence of causality from the

environmental state. We detected the occurrence of causalities by speculatively predicting the next state with Predictors and comparing the prediction results with the actual perceived data. If the causality represented by a Predictor happened, the Predictor's prediction matched actual data; but it didn't match if the causality did not happen. This process is the competitive selection of best Predictor from a pool of Predictors based on the actual experience of the phenomenon. Then we could use the detected causality occurrence to train the corresponding Selector, which learned the relationship between the sensory input and Predictor selection. If no Predictor correctly predicted the actual data in one moment, a new causality must be happening, and a new Predictor and Selector were created to learn the current causal relation. The incremental acquisition of Predictor and Selector pairs corresponded to the accumulation of knowledge about the environment.

After the learning of Predictors and Selectors, the proper Predictor is selected and applied to the environmental state of each moment. Given the perceptual input of a moment, the Selectors competed for its activation and the Predictor that corresponded to the highest activated Selector was used to predict the moment. The selected Predictor changed time by time depending on the state of the task, which represented the regularities of the world. By iteratively applying the Predictor selection process to the predicted state, we realized longer term prediction of the environment from the current state without specific modeling. Then, we realized the autonomous construction of task models by dynamically choosing Predictors in a perceptually driven way.

The same Predictors acquired in previous tasks could be used for new but similar tasks. The application of Predictor to new tasks with a different state space is possible because each Predictor only uses a subset, not all the variables for the task, to represent the local relation. Even when the new task has a partially different state space, Predictors and Selectors could be applied if the subset of the variables was contained in the new state space. The application range of Predictor and Selector was wider when the subset size was smaller. So, after additional learning of newly required Predictors and Selectors in the new task, they could be combined with previous learned ones and used to construct the new task model by observing the environmental state. By experiencing new tasks one by one, we expect the system to start covering a range of possible variation tasks in an incremental way.

Then after the task model was constructed, a prediction based RL method could be applied to realize quick action learning based on the current state and the predicted rewarding state (Ohigashi et al. 2003). This is the entire image of our dynamic task model construction model. In the next section, we show the

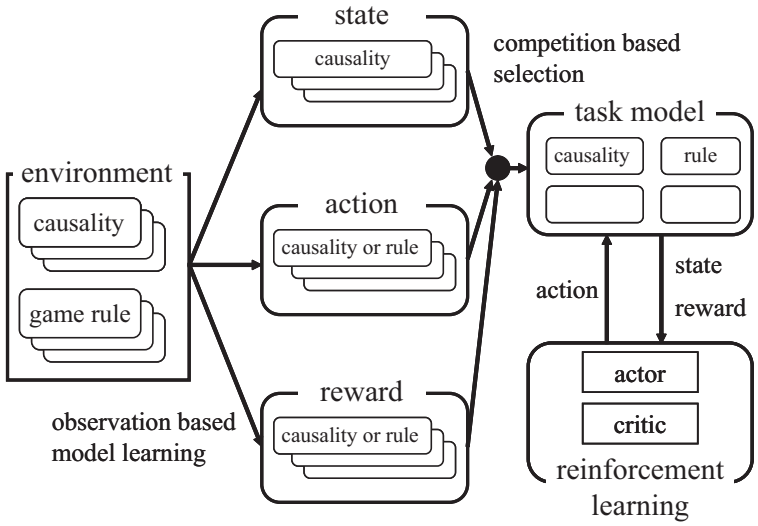

Figure 4: Construction of task model by dynamic combination of local predictors of environment

results of the ping-pong case.

3 Construction of task model by combination of local predictors

3.1 Acquisition of local predictors

At the beginning of learning, our model had three different kinds of local prediction models (LPM): one for the state transition of the environment, one for expected rewards, and one for the state transition by selected action. Each LPM consisted of a Predictor and a Selector corresponding to the Predictor.

The LPM predicted states_{t+1} of an event at next time step $t+1$ from states_t at current time t. Input s_t for each LPM was different subset of state variables concerning the event. Each of the LPM may not predict all events but just a part of events concerning the object. A model may learn to predict the ball's movement by the causality of its straight movement by the law of inertia, for example, and another model may learn to predict the ball's movement by another causality of its deflection off the wall. Each LPM is specialized to local laws, and the state transition of the target is predicted by a set of LPMs. For example, in the ping-pong game, ball movement is described with two LPMs, one for straight movement and one for deflection off the wall.

We define notation here. P^i and S^i represent the Predictor and Selector for each LPM i. Predictor and Selector are learned according to the following procedures.

1. Predictors of acquired all LPMs predict next state $s_t + 1$ from current state s_t. Here i is an index of LPM.

   $P^i(s_t) = s^i_{t+1}$

2. Calculate a prediction error $error^i_t$ of each predictor P^i using state s_{t+1} at the next time t+1.

   $error^i_t = s_{t+1} - s^i_{t+1}$

3. Select the predictor that has minimum prediction error.

   $winner = arg\min_i error^i_t$

4. If the prediction error of the winner predictor didn't reduce even after learning, a new LPM is added to learn the state transition. If not, Predictor P^winner learns by using $s_t + 1$ as teacher signal and Selector S^winner learns by using 1 as a teacher signal and Selectors other than the winner Selector learn by using 0 as a teacher signal.

   $if \quad error^{winner}_t - error^{winner}_{t-1} > threshold \quad then$ add new LPM_i

   $else$ Predictor and Selector learns.

5. Return to 1.

By iterating the addition and learning of LPM, we got a group of LPMs specialized to relations between local variables that existed in the given task. When more than one LPM exists, all Predictors predict the current input and a LPM with minimum prediction error is chosen as the winner of the competition and learns. When the above conditional applied to the prediction error of the winner, a new LPM is added.

The number of local predictor modules (LPM) changes according to value of threshold. If the value is raised, coverage of each LPM extends and its reusability decreases. If the value is lowered, the coverage of LPM narrows and its reusability increases. We set the threshold by experience so that the reusability might increase and the performance might improve.

Selector learned whether its corresponding Predictor correctly predicted the state transition of the input state. Specifically, Selector learned to output 1 if the Predictor won the competition, and Selector learned to output 0 if the Predictor lost the competition. The input variables for Selector were not necessarily the same as the inputs variables for Predictor. The input variables to Selector must be selected from all the task's variables to facilitate correct prediction. The selection of input variables strongly influenced the reusability of Predictor and Selector. If input variables that depend on specific tasks were included in the selected variables, LPM reusability decreased. If input variables that didn't depend on specific tasks were

selected, reusability increased. In later simulations, we selected input variables for Predictors and Selectors by hand to maintain high reusability. However, the autonomous selection of input variables is an important problem for the discovery of regularity from sensory information, a problem for future work.

3.2 Dynamic construction of task model by selection of local prediction models

We define notation here. Symbols e, r, and a represent LPMs that predict state transition, the expected reward, and the state transition by selected action, respectively. For example, the Predictor that predicts expected rewards is described as P_r. The task model was constructed according to the following procedures.

1. Action a was decided randomly.
2. Select Predictor for action P_a^i by competition between corresponding Selectors; the winner predicted the state transition. i is the index of the local predictor.

   $win_a = arg\max_i S_a^i(s_t), \quad P_a^{win_a}(s_t, a) = s_{t+1}$
3. Select a Predictor for environmental change P_e^i by competition between corresponding Selectors; the winner predicted state transition.

   $win_e = arg\max_i S_e^i(s_t), \quad P_e^{win_e}(s_t) = s_{t+1}$
4. The predicted state was given to LPM to know expected reward. If the acquisition of a reward was predicted, predicted state s_{t+1} is considered goal state $\hat{s}$. If not, return to 3. If reward acquisition was not predicted within a predefined amount of prediction iteration, the predicted state was not outputted.

   $win_r = arg\max_i S_r^i(s_t)$

   $if \quad P_r^{win_r}(s_{t+1}) \neq 0 \quad then \quad s_{t+1} = \hat{s}$

   $else \quad return to 3.$

The constructed task model works as an environment simulator and predicts state transition of current task. In the simulation, the reward predictor predicts appearance of reward in each of predicted states. The predicted reward is used as *reward* term of TDerror equation of reinforcement learning (see sec 3.3). The output of reward predictor is used also as a condition which stops the prediction of state transition by LPM. So, the reward predictor exists outside of reinforcement learning and its output is used as similar as actual reward.

Our proposed method didn't keep a combination of LPMs corresponding to a task. So, it must repeat the Selectors' competition process whenever it is used, and such calculation cost is not small. This problem can be solved by keeping a combination of LPMs for each task. However, the solution requires a model of higher level task understanding, which includes task identification and top-down task model construction based on the evaluation of similarity between tasks and reusing a combination of LPMs. It remains as future work.

However, the dynamic combination of LPM realized fast adaptation for similar tasks. This is a distinctive feature of our model, and the effect is large enough even if the above disadvantage is ignored.

3.3 Reinforcement learning based on task model

The constructed task model was used to predict state transition in a task. Predicted reward state s_{t+1} is considered as goal state $\hat{s}$. A suitable action for a pair of current and goal states is learned by a TD-learning method based on predicted rewards. The update equation is described as follows:

$$TDerror = reward + \gamma * V(\hat{s}) - V(s_t)$$

The state value and selection probability of actions were updated based on the value of TD error. When the task model couldn't give prediction output, next time step state s_{t+1} was used, as in TD learning.

This learning method is identical to the PRL model (Ohigashi et al. 2003) that we proposed before. It realized faster learning than the actor-critic RL model because it decided action based on the current and predicted states expected to acquire rewards.

3.4 Applications to ping-pong game

Our model was applied to a ping-pong game in which an agent operated a paddle with left, right, and stay actions and returned the ball to acquire rewards. The behavior of our model in the ping-pong game was as follows.

- Acquisition of LPM

  Predictor $P_e(s_t)$ for environmental state changes predicted the ball's position at the next time step from the current position. $P_e^1(s_t)$, prepared before the learning, usually concentrated on predictions of the ball's straight movement, which occupied a majority of the ball's movements. Since $P_e^1(s_t)$ couldn't predict the ball's deflection from the surrounding walls, new predictors $P_e^i(s_t)$ were added

that concentrated on predicting ball rebounds. Corresponding selector $S_e^i(s_t)$ learned whether $P_e^i(s_t)$ correctly predicted the current state using the distance between the ball and the wall and the direction of its movement.

Reward predictor $P_r(s_t)$ learned to output value 1 when the paddle returned the ball. In this game, only one $P_r(s_t)$ was acquired because the conditions for reward acquisition were only one. If there were multiple reward conditions and the first $P_r^1(s_t)$ couldn't predict its occurrence, new $P_r^i(s_t)$ were added and learned.

The state change predictor by action $P_a^i(s_t)$ predicted the paddle position of the next time step from the current paddle position. As paddle operations were limited and simple, one predictor was enough in this case.

- Dynamic construction of task model

  The task model was dynamically constructed by combining the acquired LPMs. First, the task model predicted paddle position at time $t+1$ using $P_a(s_t)$ from the current paddle position and selected action a. Second, the task model predicted the ball's trajectory using $P_e(s_t)$ from the current ball position. Finally, $P_r(s_t)$ predicts reward acquisition when the predicted paddle position was identical to the predicted ball position. We obtained game state s_{t+1} from the current state s_t by applying related LPMs, each of which represented regularities of the environment. Then, iterating single time step prediction from time t to $t+1$, we realized longer time prediction as long as each local prediction worked correctly. In short, we could realize the task model that predict whether the ball fell at the paddle position in the near future of the game.

- Reinforcement learning based on the task model

  The action learning for the current state and the state predicted by the task model enabled the agent to learn an action strategy that predicted the ball's falling position from its current position and to move the paddle into the predicted ball position. This efficient strategy is often adopted by humans when playing the game.

4 Computer simulation

4.1 Game variations

We performed computer simulations using the ping-pong game (Fig. 2: Normal task) and variants (Fig. 5: Two-ball task, and Fig. 6: Wall-change task) to confirm that reusing knowledge leads to quick construction of task models for similar tasks. On the game panel, players saw the paddle and the ball in a two-dimensional field. From the panel, a set of state variables was extracted for learning. The x/y-coordinates of the space were defined within a range of $[0, 1]$. The direction of the ball's movement was represented by an angle between $-\pi$ and π, and a zero designated a down direction. At each time step, the paddle could perform one of three actions: staying, moving left, or moving right.

The ball complied with the physical laws of moving straight at a constant speed and rebounding off the wall. A sensor perceived the collision between the ball and the wall or the paddle in following order [Up, Left, Right, Down, Paddle]. For example, we encoded the sensor input with a bit sequence like $[1, 0, 0, 0, 0]$ when the ball collided with the upper wall or $[0, 0, 0, 0, 1]$ when the ball collided with the paddle. The game's goal for the agent was returning the ball by operating the paddle. By hitting the ball, the agent got a positive reward. A negative reward was given when the agent missed the ball because of incorrect paddle operation. We defined an epoch of game play as a "ball traveling time" in which the ball moves up from the bottom line until it reaches the bottom line again through deflection at the top line. The agent decided an action at every step of the ball movement simulation according to its velocity. One epoch corresponds to about 20 to 40 steps depending on the direction of the ball's movement.

In two-ball and wall-change tasks, the conditions of reward acquisition and the rules of ball or paddle movement were identical to the normal task. In the two-ball task, the initial movement directions of the two balls were different, and collisions between them were ignored.

4.2 Setting of local prediction models and RL parts

While the agent was playing or observing the game, he received information on the ball's x/y-coordinates, b_x, b_y, the ball's velocity, v_x, v_y, the paddle's x-coordinate, d_x, and the sensor value around the ball. The agent generated and learned the local prediction models using these inputs.

We used a three-layered neural network with backpropagation to realize Predictor and Selector. P_e predicted the ball's velocity at the next step $(v_x, v_y)_{t+1}$ from the ball's current velocity $(v_x, v_y)_t$. P_r

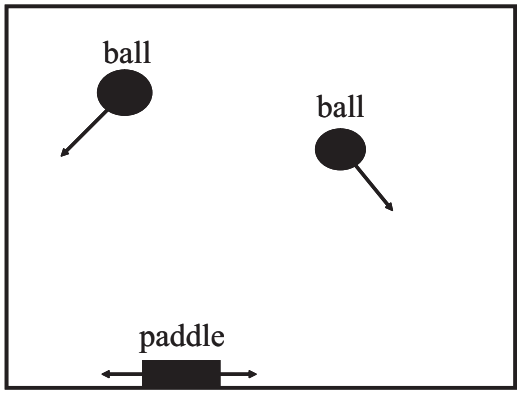

Figure 5: Two balls task. An additional ball was added to normal task. The balls moved around without collision and paddle chased the lower ball.

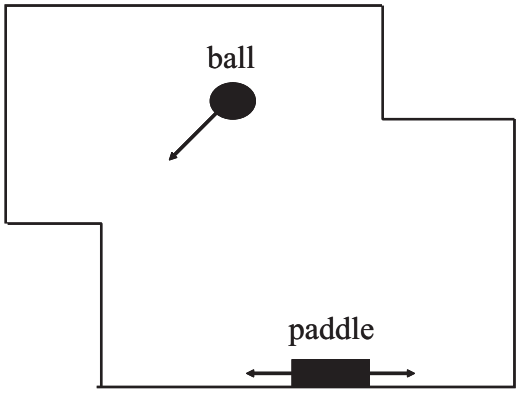

Figure 6: Wall change task. Shape of wall was changed from normal task.

predicted the reward (1 or 0) from sensor values. P_a predicted the paddle's position at the next step from its current position and selected action. S_e and S_r received the sensor values and predicted whether the corresponding predictor could predict the state of the next step from the current state. S_a received the selected action and predicted whether the corresponding predictor could predict the state of the next step from the current state. The number of prediction steps was limited to 100. If reward acquisition was not expected from the predicted state until the maximum number of prediction steps was reached, the agent performed a soft max action selection rule with current state $s_(t)$, which is commonly used as an action selection rule for conventional actor-critic based RLs.

In the RL part that actually learned the actions using the task model, we prepared a lookup table for the value function by dividing all current paddle positions and the goal paddle positions predicted into ten equal parts; thus a total of $10 * 10 = 100$ grid entries were prepared in the state space. The agent learned

proper actions for each entry.

After the agent acquired the necessary LPMs for the current game, it constructed a task model by combining LPMs, thus becoming able to predict the entire game's transition using the task model. In this prediction phase, the agent updated b_x, b_y, v_x, v_y, P_x to the value of the next step using Predictors for the current step.

However, the current version of our model still has some parts in which we have to ask the intervention of human engineers. The agent can't update the sensor value input because the LPM for sensor values is not prepared. S_e and S_r need sensor values as input. In this simulation, we assumed that the agent can observe the game panel in the prediction phase and receive the sensor value corresponding to the ball's predicted position. In two-ball tasks, we gave prior knowledge that the laws of ball movement are the same for both balls and designed the agent to select and predict the ball that has a lower b_y value because we knew that the ball would reach the bottom line faster than the other one. In fact, the normal and two bal tasks are essentially the same.

4.3 Procedure of computer simulations

In the simulations, the learning agent initially played the normal task for 10,000 epochs and acquired LPMs. Next, it learned two-ball tasks for 10,000 epochs. Finally, the learning agent learned the wall-change task for 10,000 epochs. At every change of tasks, we flushed the action table of agent to compare the improvements of the learning performance. Here, we expect drastic improvement of the action learning performance at the beginning of the second and third tasks. We evaluated the performance of our model by monitoring its success rate for every 100 learning epochs and observed task model reconstruction process after the change of tasks.

4.4 Results of task model construction for similar tasks

Figure 7 shows the changes of the success rate in every 100 learning epochs and the total prediction error of acquired LPMs. The learning agent didn't learn efficiently with the task model during the initial phase of the normal task learning because it had to acquire LPMs. After LPMs were acquired, learning was immediately accelerated. When tasks were changed after 10,000 and 20,000 epochs, the success rate fell to initial state levels because the action table was initialized. However, the success rate rose very rapidly because the LPMs acquired in the initial normal task learning were reused and the task models were quickly

constructed leading to the realization of quick action learning. The upper limit of the success rate in the two-ball task was lower than the other tasks, caused by the paddle's slow moving speed. The learning agent couldn't return the ball with the paddle when two balls came down simultaneously.

Figure 8 shows initial 3000 trials in figure 7. X axis of figure 8 corresponds to x axis of figure 7 from 0 to 30. Figure 8 shows the changes of the success rate in all 100 learning epochs and the prediction error of acquired LPMs respectively. In the learning stage "A", the agent learned P_e which predicted such physical laws of the the ball's motion as straight moving and wall deflection. In the learning stage "B", the agent learned P_r mainly. It took longer time to learn P_r than the learning of P_e. Because P_r needed too many trials than P_e to have an amount of experience of returning the ball by controlling paddle's motion and acquiring the reward for its learning. When the learning of P_e and P_r converged, the agent started to predict by task model. In the learning stage "C", the agent updated the action table by MBRL based on task model.

Figures 9 and 10 show the process of task model reconstruction after the task changed to two-ball and wall-change tasks, respectively. The horizontal axis of the figures shows the number of simulation steps after task changes. The vertical axis shows the number of selected prediction module used for the new task after task change. The value is average of five trials.

In both Fig.9 and 10, most LPMs clearly began to be used in initial tens of steps, that is, a few epochs. Average of 31.2 epochs and 15.6 epochs were required until the last LPMs for task model construction were selected in Fig. 9 and in Fig. 10 respectively. This speed of task model construction was so fast that the time didn't reach even to the one notch of horizontal axis in Fig. 7. These results show that task models were constructed quite rapidly compared to normal task learning.

5 Discussion

5.1 What has been realized for autonomous task learning?

In the previous section, we showed that we could construct a task prediction model for the ping-pong game by a dynamic combination of LPMs and realize quick action learning in combination with a prediction based RL. In this study we sought model of human processes for analyzing tasks and realizing a task processing procedure that leads to autonomous task learning. We revealed the importance of local knowledge accumulation in daily experiences and knowledge selection in new tasks. Though the problem is important,

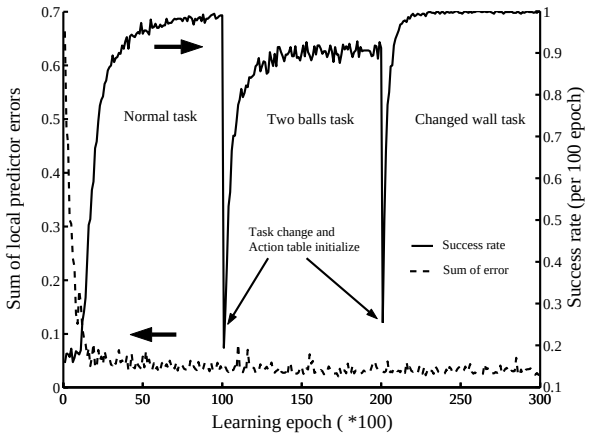

Figure 7: Success rate transition for every 100 learning epochs and total prediction error of local predictors

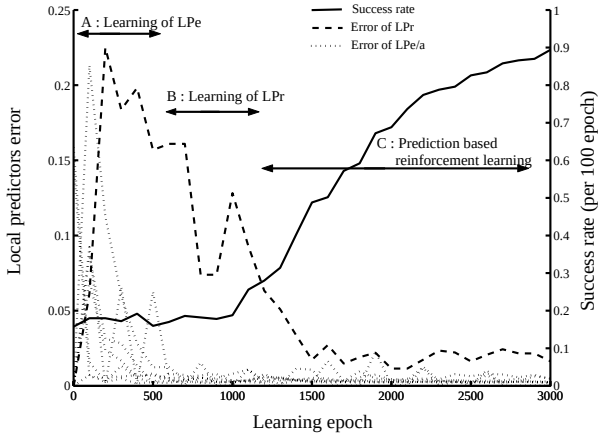

Figure 8: Success rate transition and prediction error of each local predictors

in this paper we only realized a part of our case study on specific tasks.

A MOSAIC model has a similar structure and function to our proposed model [D.M. Wolpert and M.Kawato 1998]. MOSAIC model features pairs of a forward model and an inverse model for each control target, and a model pair is chosen and used at each moment to control the target motion. Though the detailed function and learning method differ from our model, the concept of model pair is similar to the LPM that we used as parts of the task model. But in this study, we use model pairs as parts for the macroscopic function of dynamic task model construction that is essential in multiple tasks environment. What is important in our model is the dynamic nature of the task model formation; we don't insist on its implementation method. When we focus on the modeling of a target with complicated operation modes, such forward and inverse type architecture as MOSAIC and ours appear to have wider application ranges that are not limited to motion control.

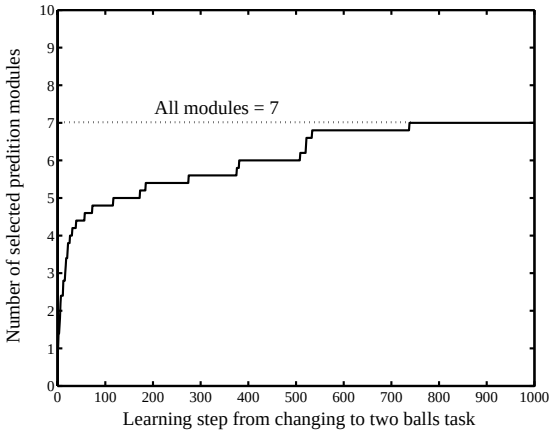

Figure 9: Accumulated number of selected prediction module after task change from normal to two balls task. Average of five trials.

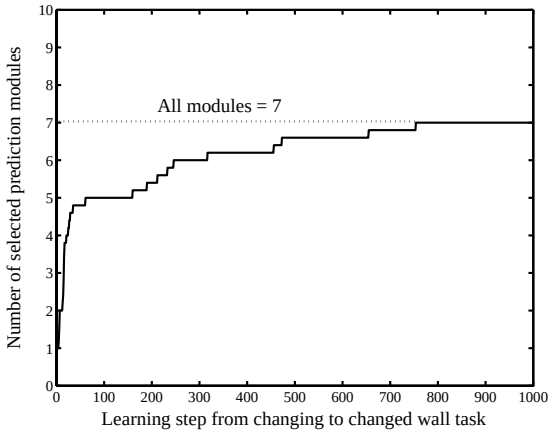

Figure 10: Accumulated number of selected prediction module after task change from two balls task to wall change task. Average of five trials.

5.2 Autonomous problem solving based on knowledge reuse

The ping-pong game and its variations in this study were just toy problems and too simple as tasks in multiple tasks environment. But the performance of the proposed method is clear, and we could observe the utility of knowledge reuse acquired from past experience. What mechanism of the proposed method enabled such performance?

First is the high reusability of LPM. Predictors and Selectors functions were defined in the subspaces of large task state spaces by selecting the input and output variables. They learned the local relation between those limited variables. Prediction could be applied to the new tasks that have different state variables when the new task space included the subspace variables. The learning of Predictor and Selector also became

easier because the range of input-output relation was limited.

Second is the quick construction of task models. By choosing previously learned regularities by observing target tasks, rapid construction of task models is realized that enabled quick action learning within the domain of task variations. Solving the long learning time problem in MBRL will increase new applications for machine learning.

However, we must be careful about the proposed method's generality. Though we tried not to introduce task specific constraints in our model, we don't deny the possibility because the application samples were simple and limited. We have to evaluate our model's generality by applying it to diverse and more realistic tasks.

5.3 Reuse of action tables and task models

In our current model, we reused LPMs in new tasks, but didn't reuse the acquired action tables that were used to guide the task state from the current one to the desired one. Actually, we may be able to reuse the action table acquired in the normal task in the two-ball and wall-change tasks to realize quicker, actually immediate, learning. However, at least now, we don't know the general mechanism of the action table selection that evaluates the possibility of reuse without depending on human intuition.

Another unsolved problem is reusing the task model and the combination of LPMs. Since our current model didn't preserve the task model, it has to consume time to construct a new task model even when it encounters previously experienced tasks and its variations. If the model could extract knowledge about the combination of LPMs and modify it depending on new task performance, we might have realized quicker adaptation to the task. For its realization, we need a method to extract the features of a task and calculate similarity between them. Then we might be able to find a task model similar to the current task situation and apply it in a top-down manner in addition to our current bottom-up manner.

5.4 Variable selection problem and real environment complexity

In the case study of this paper, we designed a subset of variables given to all Predictors and Selectors by hand. For their reusability, input variables must be limited to necessary ones. And the inclusion of unnecessary variables for the representation of the input-output relation disturbs learning speed and precision. The proposed method actually presupposes a solution to the variable selection problem in the LPM construction phase. But in this paper, we focused on the concept of the entire system and avoided the variable selection

problem. Variable selection will be a key problem that decides the success of our model.

5.5 Choice of action learning method

The aim of this paper is the proposal of task model construction method. We used conventional MBRL as an action learning method with a task model. But action learning methods are not limited to RL. We may use other methods, including a symbolic tree search or Bayesian probabilistic inference, when exploiting the task model as an environmental prediction method. In such cases, the information representation of the task model must be changed depending on the action search method. Still, the essence of the task model is its construction method. To discuss the possibility of task model use in different action search methods, we may need to consider a more generalized mathematical description of the task model construction process.

5.6 Correspondence in the brain

Does the knowledge reuse mechanism in our model exist in actual brains? Evidence that the many cortical areas in the brain are activated depending on the task is consistent with our concept of knowledge and its reuse. But it remains unclear whether the activated areas are used for event prediction, as we suppose, or whether they are used for different purposes. Many neurophysiologic researches are studying brains that are trained on a specific task by training animals, and it will be difficult to reveal brain activity in multiple tasks environment. We need neurophysiological evidence about a brain that learned many similar but different tasks. Though the classical learning set paradigm matches this requirement well [Warren 1965], there are few studies in the field.

What knowledge activation control mechanism corresponds to Selectors? A Selector's function is sensory signal driven neural circuit activation, which is identical to bottom-up attention. In contrast, Ogawa and Omori proposed a neural network model for problem solving that searches for a combination of many neural circuits in a top-down way [Ogawa A and Omori T]. The method can be regarded as top-down attention. It is natural to think that task model construction and problem solving neural circuit construction in the brain are realized by a combination of bottom-up and top-down attention systems.

6 Conclusion

In this paper, we proposed a method of task model construction by a dynamic combination of LPMs. To evaluate the method, we conducted a computer simulation on a series of task variations using the task model and MBRL. The results showed the autonomous construction of task processing procedures and quick action learning. We used RL as an action learning method, but we should not limit it to RL. The dynamic construction of the task model may be a method that could be applied to a wider range of problems including insight.

The specific feature of our proposed method exists in the online task model construction by the reuse of knowledge acquired from past experiences. This is also a specific feature of human problem solving behavior. However, to apply the method to real world, we will have to solve various problems including hidden variable discovery, avoidance of conflict between LPMs and variable selection for predictors and selectors from plenty of possible variables. These problems seem to have solved in human. We expect to bridge human cognitive behavior and brain computational processing by developing an experimental paradigm that can be used to evaluate human knowledge reuse by improving the proposed model through experimental studies.

References

D. M. Wolpert and M. Kawato. (1998) Multiple paired forward and inverse models for motor control. Neural Networks, 11, 1317-1329.

K. C. Tan, Y. J. Chen, K. K. Tan, and T. H. Lee. (2005). Task-Oriented Developmental Learning for Humanoid Robots. IEEE TRANSACTIONS ONINDUSTRIAL ELECTRONICS, VOL. 52, NO.3.

K.Doya, K.Samejima, K.Katagiri, M.Kawato. (2002) Multiple Model-Based Reinforcement Learning. Neural computation, 14, 1347-1369.

J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. (2000). Autonomous mental development by robots and animals. Science, vol. 291, no. 5504, 599-600.

J. Weng, W. S. Hwang, Y. Zhang, C. Yang, and R. Smith. (2000). Developmental humanoids: Humanoids that develop skills automatically. in Proc. First IEEE-RAS Int. Conf. Humanoid Robots, Cambridge, MA, Sep.7-8.

M.Haruno, D.M.Wolpert, M.Kawato. (2001) MOSAIC Model for Sensorimotor Learning and Control. Neural Computation, 13, 2201-2220.

Ogawa A. and Omori T. (in press). Acquisition of learning processing in navigation task using Functional Combination Model. Systems and Computers in Japan, WILEY.

R.A.Jacobs, M.I.Jordan, S.J.Nowlan, G.E.Hinton. (1991) Adaptive Mixtures of Local Experts Neural Computation, 3, 79-87

S. Thrun and J. O'Sullivan. (1996). Discovering Structure in Multiple Learning Tasks. The TC Algorithm. Proceedings of the Thirteenth International Conference on Machine Learning. L. Saitta (ed.), Morgan Kaufmann, San Mateo, CA.

S. Thrun and L. Pratt. (1998). Clustering learning tasks and the selective cross-transfer of knowledge. in Learning to Learn, S. Thrun and L. Pratt, Eds. Norwell, MA: Kluwer.

Sutton, R.S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. Proceedings of the Seventh International Conference on Machine Learning, 216-224.

Sutton, R.S. (1991). Planning by incremental dynamic programming. In Proceedings of the Eighth International Workshop on Machine Learning, 353-357.

Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning. An Introduction. MIT Press.

Warren, J. M. (1965). Primate Learning in comparative perspective. In A. M. Schrier, H. F. Harlow & F. Stollnitz (Eds.),Behavior of nonhuman primates (Vol.1). New York, Academic Press.

Ohigashi, Takashi Omori, Koji Morikawa, and Natsuki Oka. (2003). Acceleration of Game Learning with Prediction-based Reinforcement Learning -Toward the emergence of planning behavior-. ICANN/ICONIP 2003, LNCS 2714, 786-793.

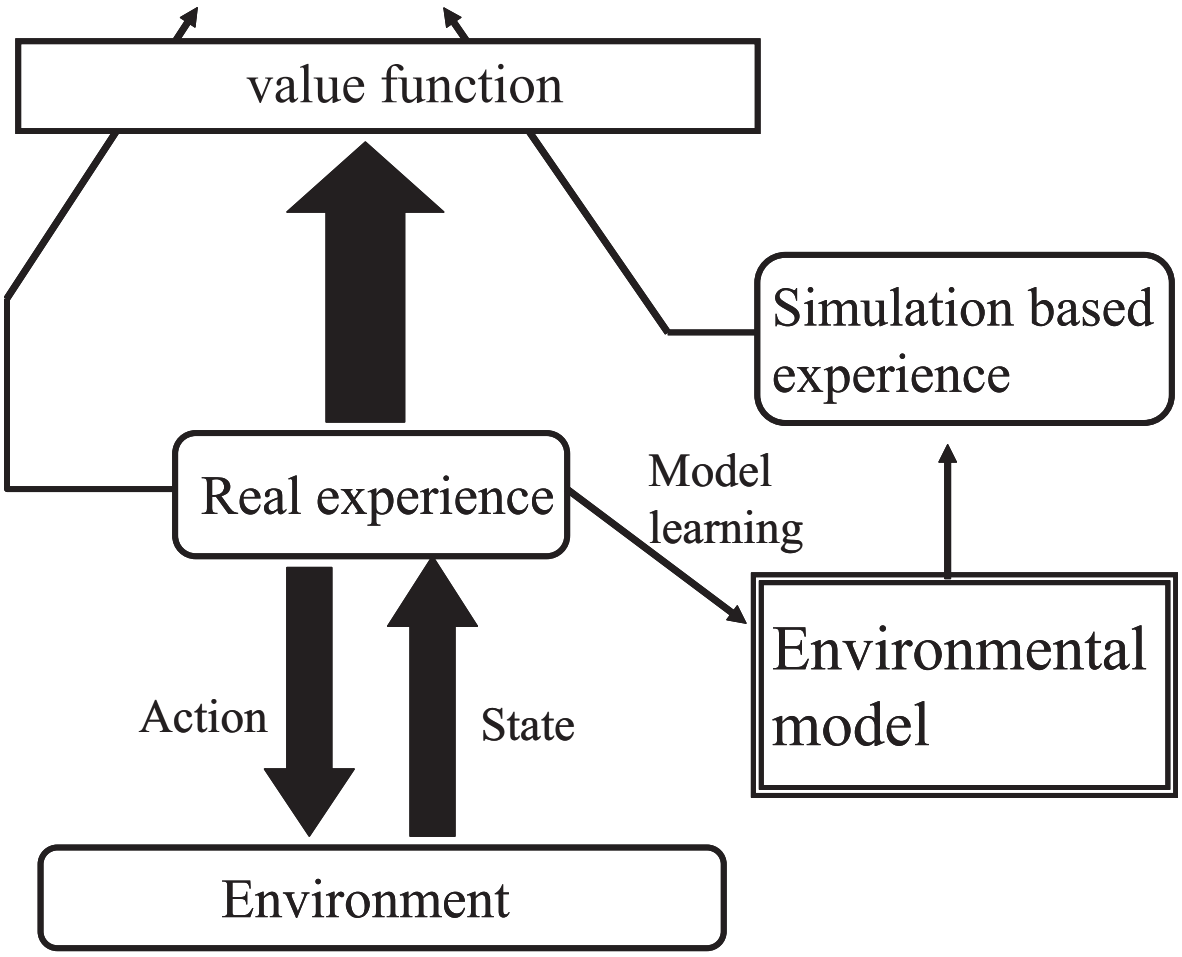

Figure 1: DYNA architecture: example of model-based reinforcement learning. Two RL methods are included in parallel. One is direct action and experience based RL, as shown on the left of Figure 1. Another is an RL using virtual action and simulated experience by environmental model on the right.

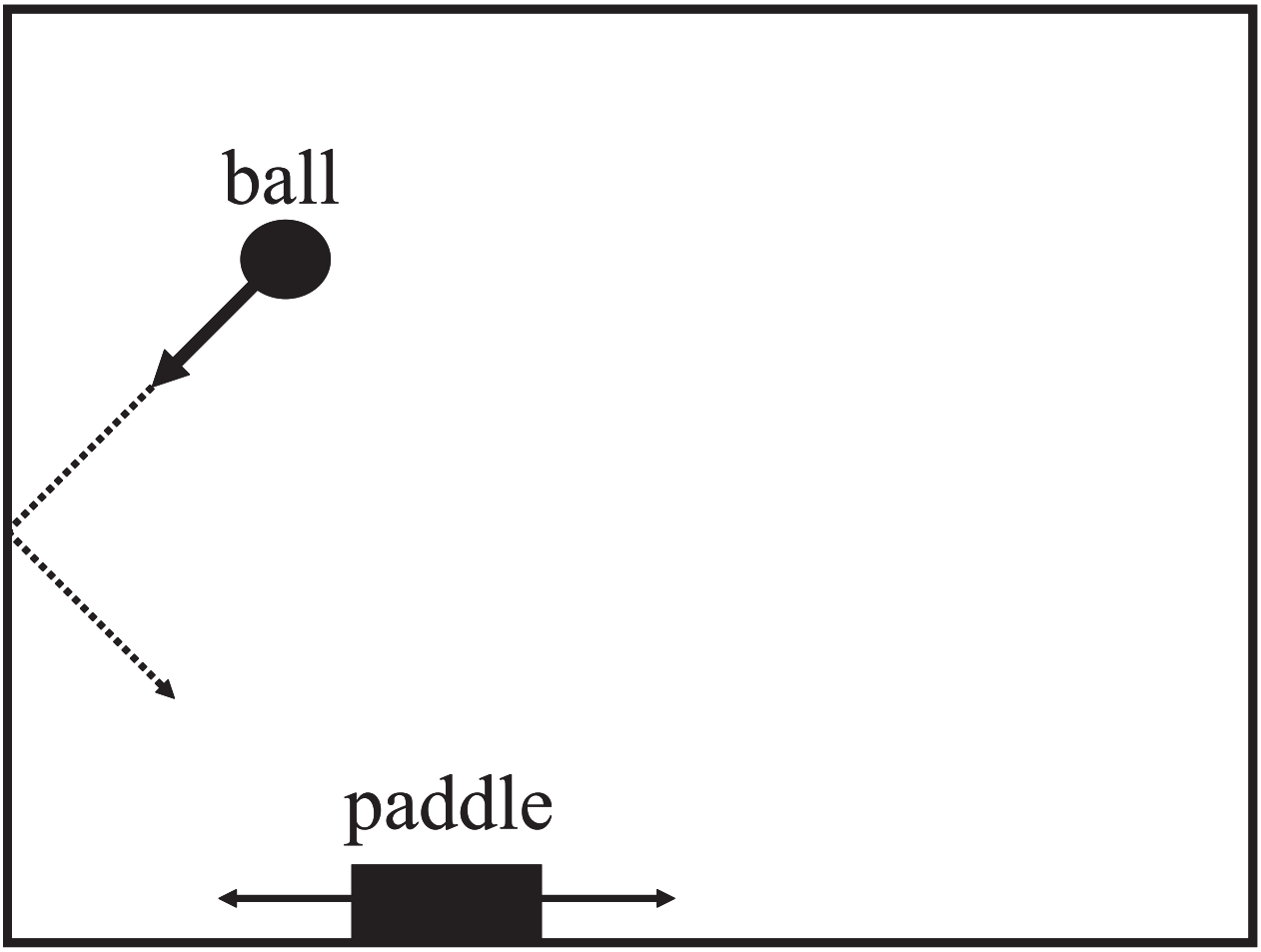

Figure 2: The ping-pong game used for this study. There are a ball and a paddle in play panel and a player gets a reward by returning the ball by controlling the paddle's motion.

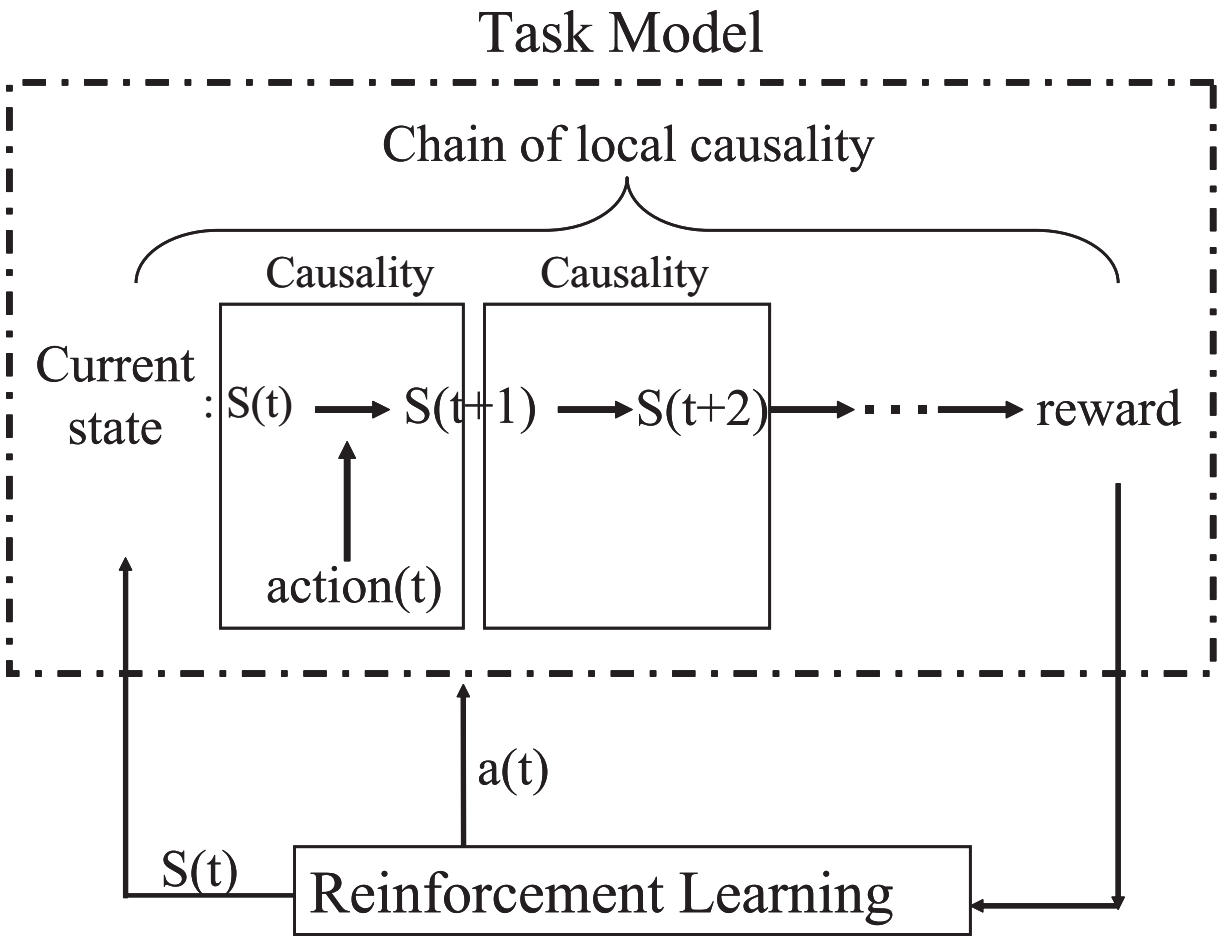

Figure 3: Task model that represents causal relations by a set of local predictors. Given a task model, we can predict acquisition of reward starting from a current task state and an action sequence.

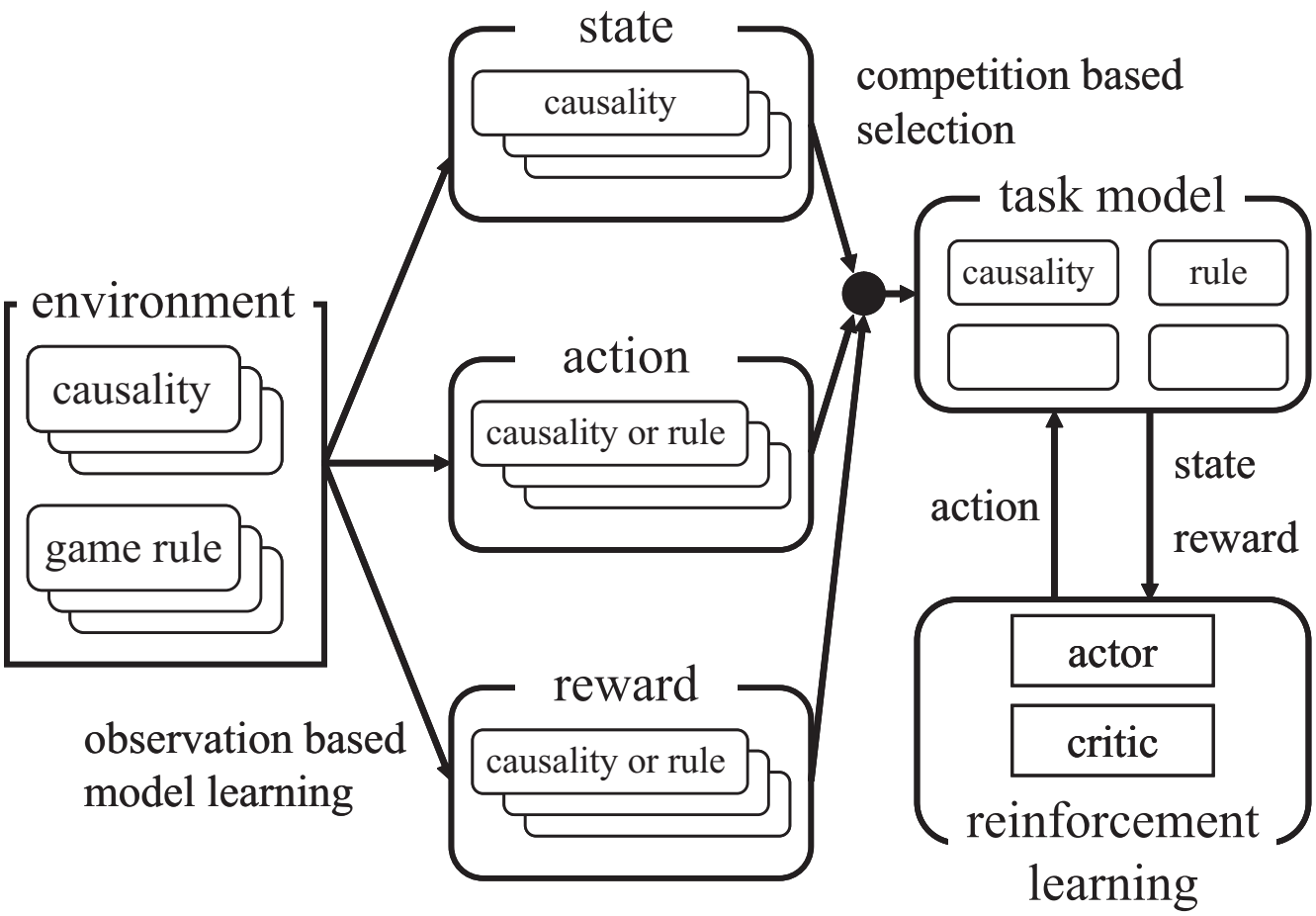

Figure 4: Construction of task model by dynamic combination of local predictors of environment

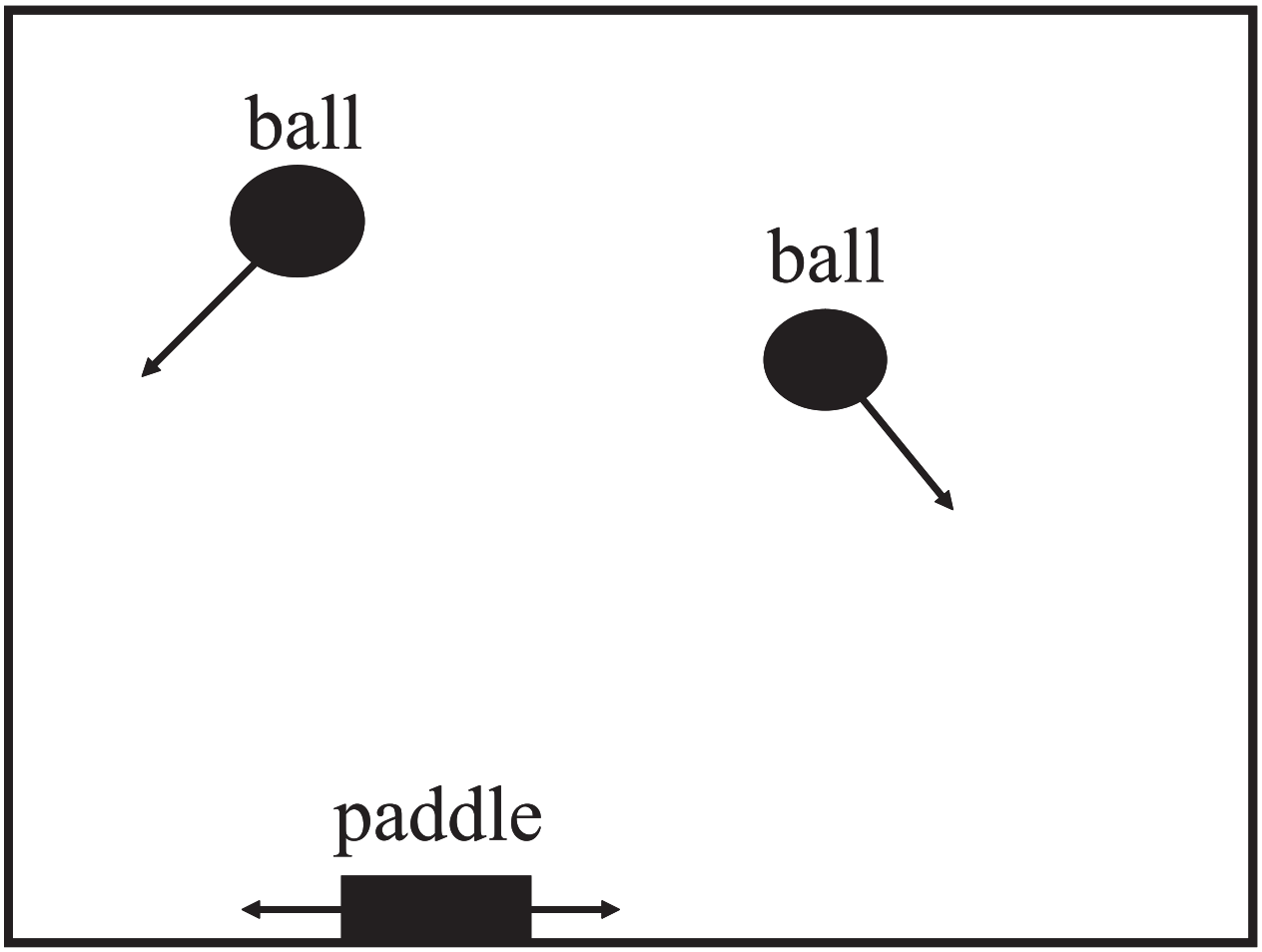

Figure 5: Two balls task. An additional ball was added to normal task. The balls moved around without collision and paddle chased the lower ball.

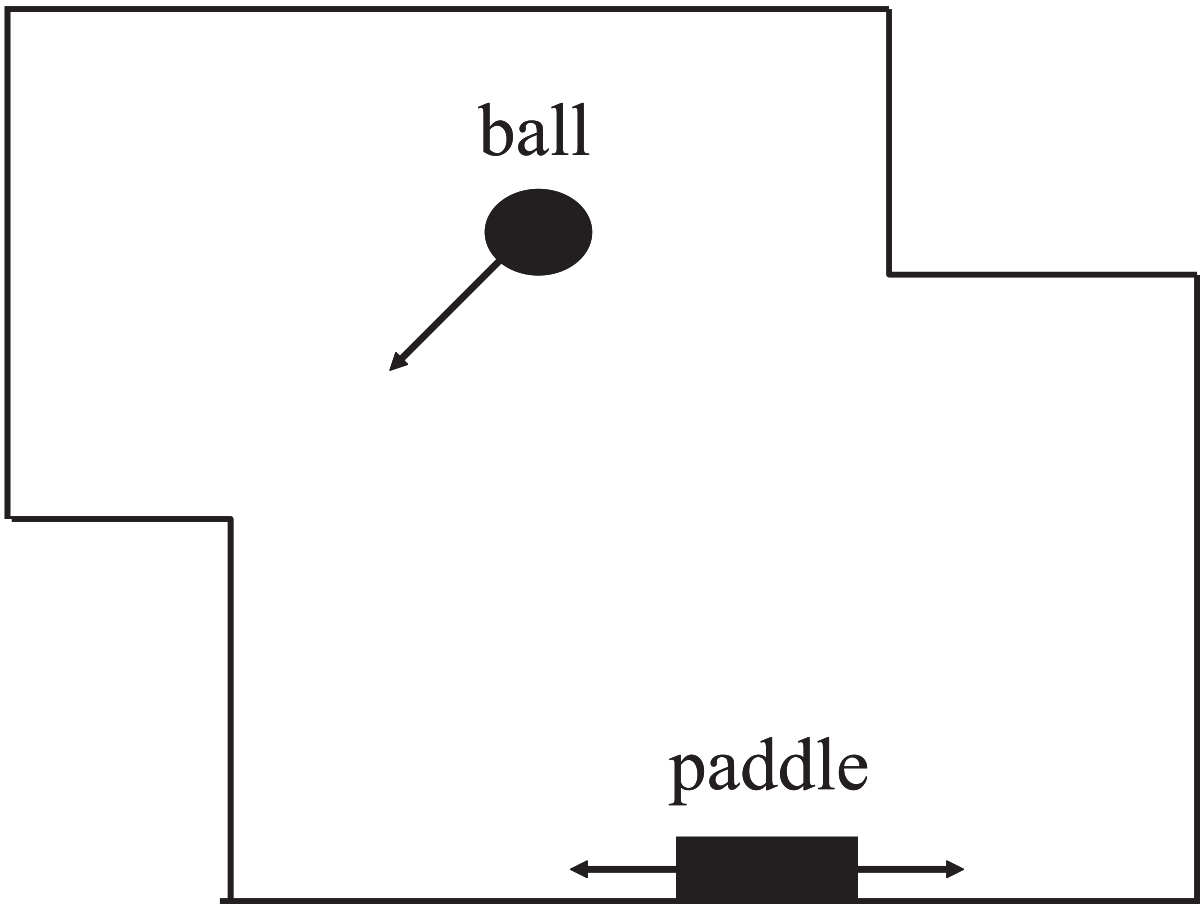

Figure 6: Wall change task. Shape of wall was changed from normal task.

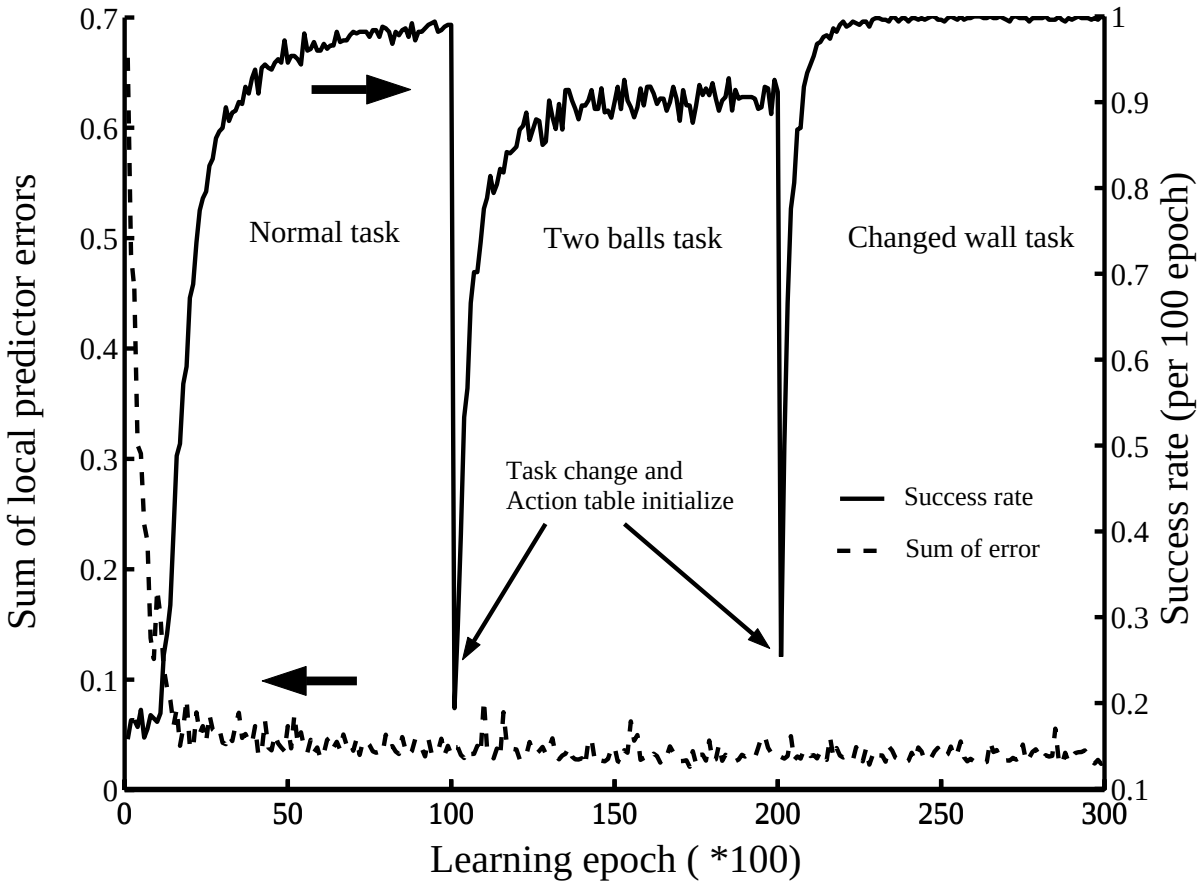

Figure 7: Success rate transition for every 100 learning epochs and total prediction error of local predictors

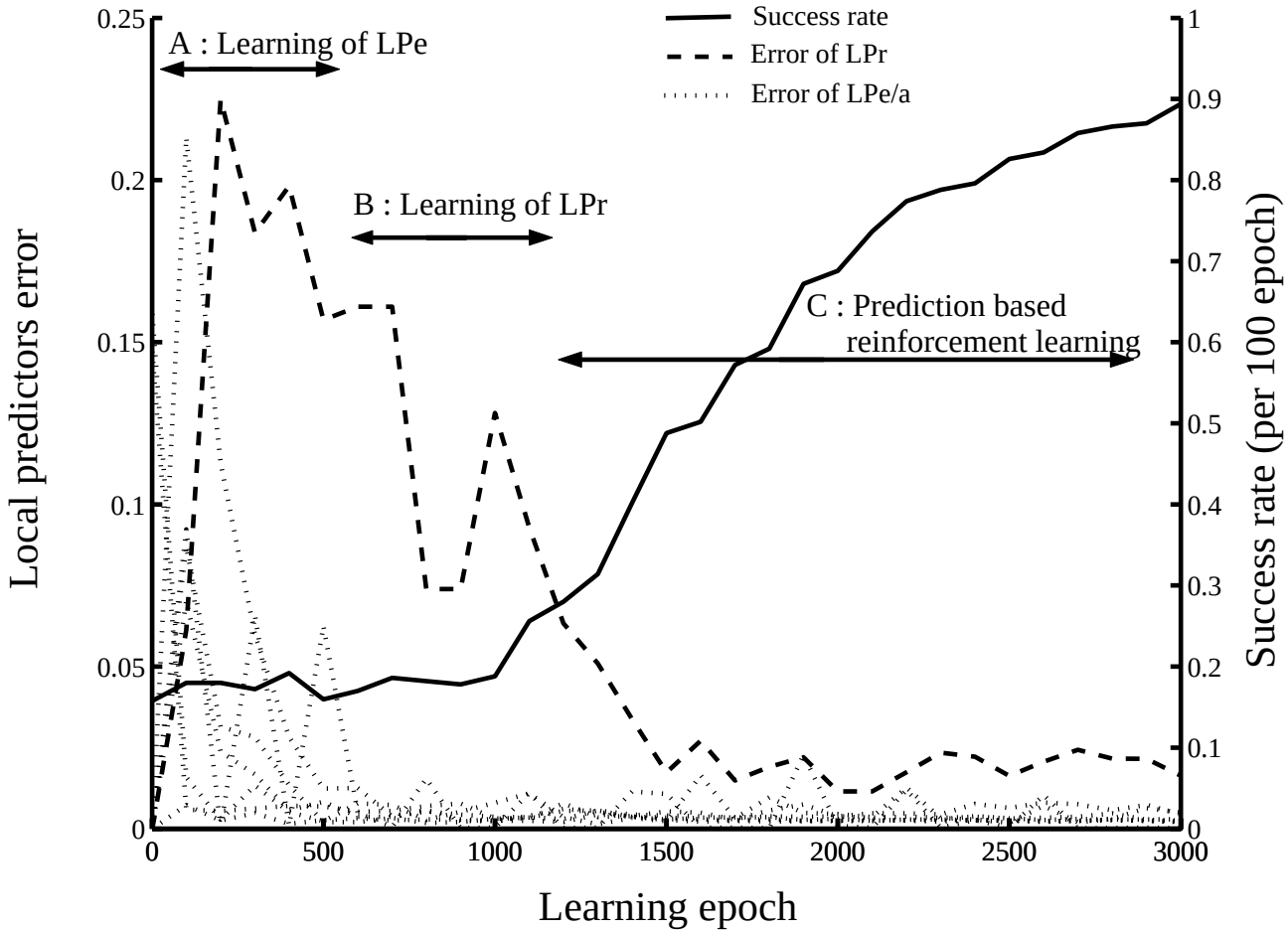

Figure 8: Success rate transition and prediction error of each local predictors

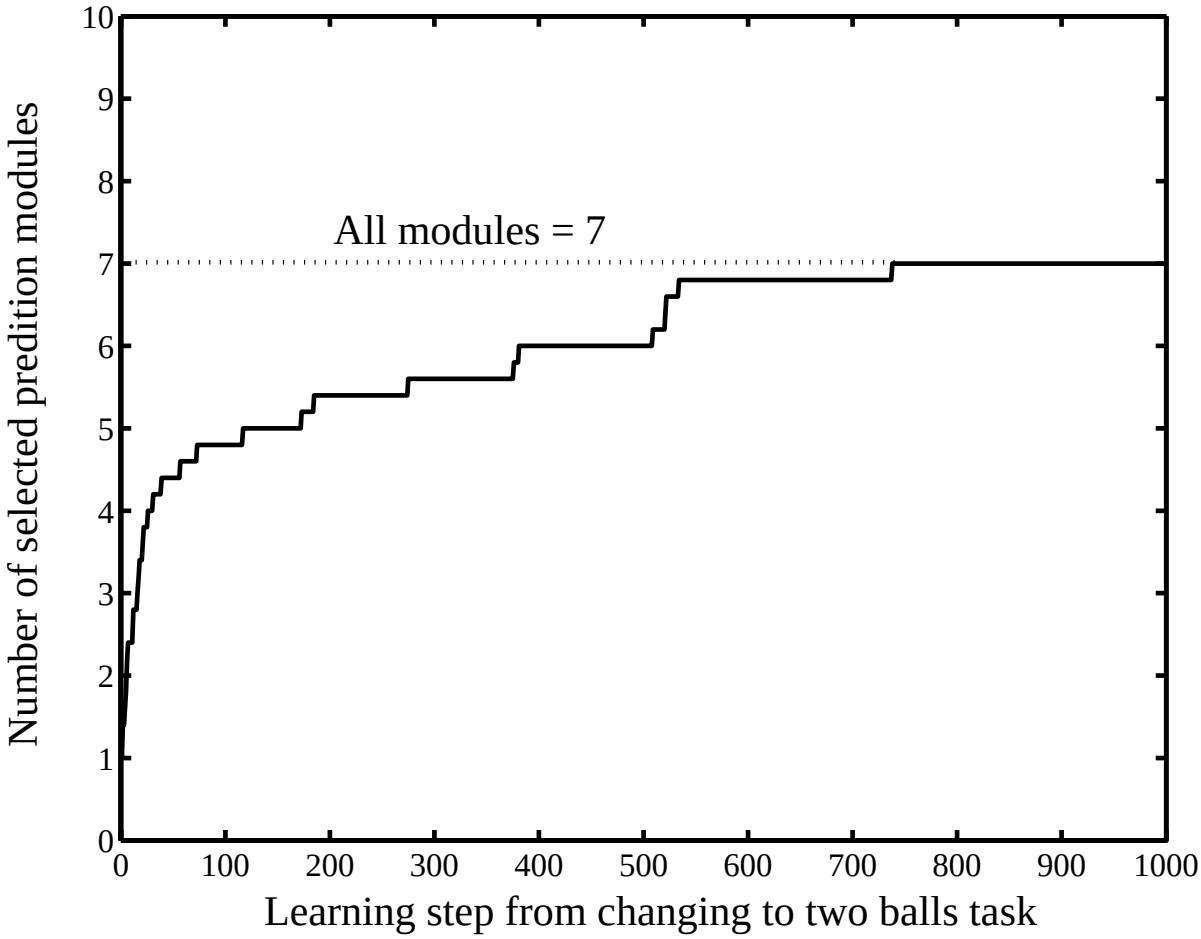

Figure 9: Accumulated number of selected prediction module after task change from normal to two balls task. Average of five trials.

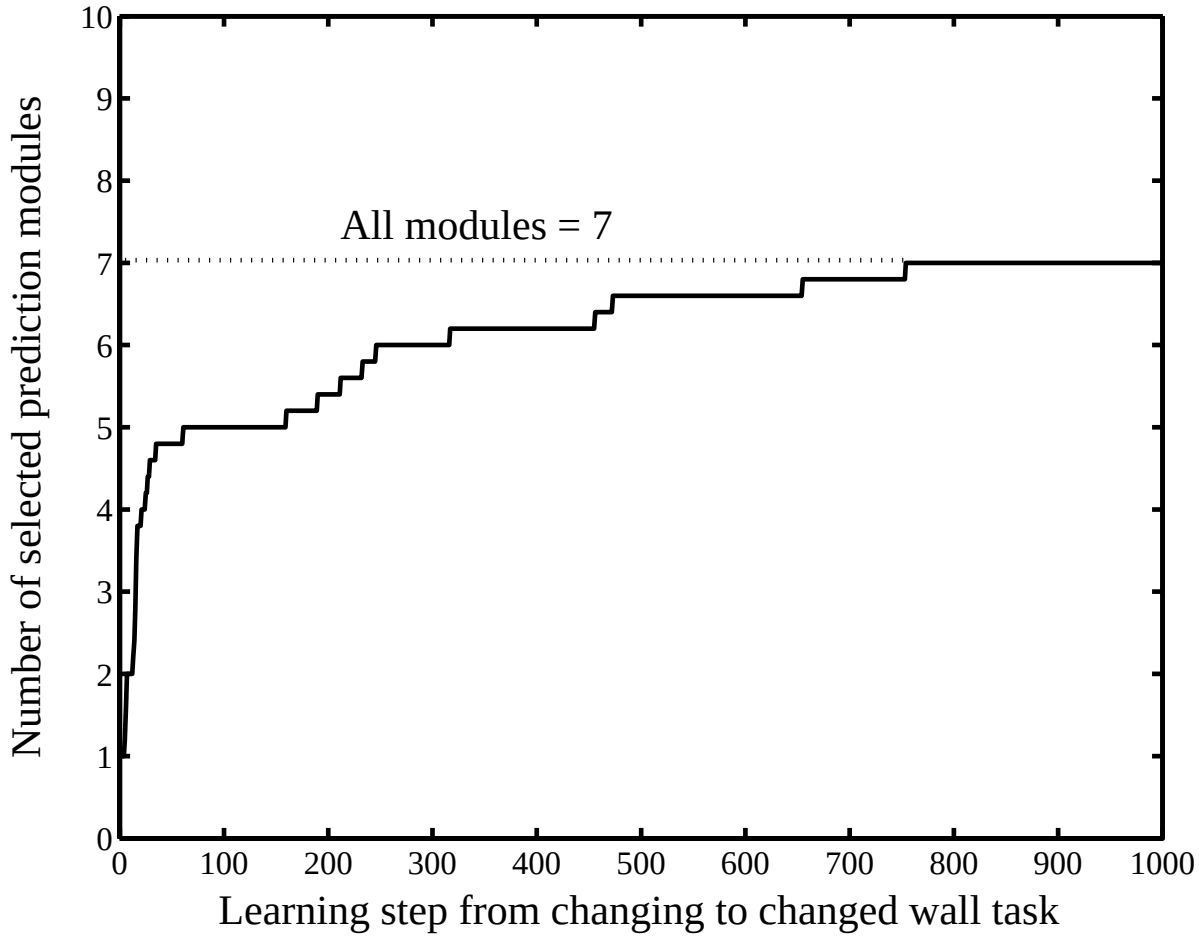

Figure 10: Accumulated number of selected prediction module after task change from two balls task to wall change task. Average of five trials.