ARTICLE **Communicated by Andrew Barto**

MOSAIC for Multiple-Reward Environments

Norikazu Sugimoto
xsugi@nict.go.jp
Center for Information and Neural Networks, National Institute of Information and Communications Technology, Kyoto 619-0288, Japan, and Department of Brain Robot Interface, Brain Information Communication Research Laboratory Group, ATR, Kyoto 619-0288, Japan

Masahiko Haruno
mharuno@nict.go.jp
Center for Information and Neural Networks, National Institute of Information and Communications Technology, Kyoto 619-0288, Japan, PRESTO, Japan Science and Technology Agency, Saitama 3312-0012, Japan, and Tamagawa University Brain Science Institute, Tokyo 194-8610, Japan

Kenji Doya
doya@oist.jp
Neural Computation Unit, Okinawa Institute of Science and Technology, Okinawa 904-0495, Japan

Mitsuo Kawato
kawato@atr.jp
Brain Information Communication Research Laboratory Group, ATR, Kyoto 619-0288, Japan

Reinforcement learning (RL) can provide a basic framework for autonomous robots to learn to control and maximize future cumulative rewards in complex environments. To achieve high performance, RL controllers must consider the complex external dynamics for movements and task (reward function) and optimize control commands. For example, a robot playing tennis and squash needs to cope with the different dynamics of a tennis or squash racket and such dynamic environmental factors as the wind. In addition, this robot has to tailor its tactics simultaneously under the rules of either game. This double complexity of the external dynamics and reward function sometimes becomes more complex when both the multiple dynamics and multiple reward functions switch implicitly, as in the situation of a real (multi-agent) game of tennis where one player cannot observe the intention of her opponents or her partner. The robot must consider its opponent's and its partner's unobservable behavioral goals (reward function). In this article, we address

how an RL agent should be designed to handle such double complexity of dynamics and reward. We have previously proposed modular selection and identification for control (MOSAIC) to cope with nonstationary dynamics where appropriate controllers are selected and learned among many candidates based on the error of its paired dynamics predictor: the forward model. Here we extend this framework for RL and propose MOSAIC-MR architecture. It resembles MOSAIC in spirit and selects and learns an appropriate RL controller based on the RL controller's TD error using the errors of the dynamics (the forward model) and the reward predictors. Furthermore, unlike other MOSAIC variants for RL, RL controllers are not a priori paired with the fixed predictors of dynamics and rewards. The simulation results demonstrate that MOSAIC-MR outperforms other counterparts because of this flexible association ability among RL controllers, forward models, and reward predictors.

1 Introduction

Robotics researchers have tried to apply reinforcement learning (RL) algorithms to acquire an appropriate control policy that obtains high rewards in complex and variable environments. To achieve this goal, an RL robot has to cope not only with nonstationary external dynamics but also nonstationary tasks (the reward function). Consider a robot that can play both tennis and squash. It needs to deal with the different dynamics of tennis and squash rackets and may also have to compensate for such other dynamic factors as the wind. These dynamic factors can change unexpectedly and implicitly, and the agent has to detect them by itself. In addition to the dynamics, the robot simultaneously has to optimize its tactics under the regulations of tennis or squash. This problem becomes more difficult when both the multiple dynamics and multiple reward functions switch nonstationary, as in real games of tennis where the robot cannot explicitly observe the intention of its opponent and its partner and has to estimate them. In general, multi-agent settings hinder other agents' objectives; one agent behaves according to its own states and its estimates of those of the other agent. In addition, the problem of the nonstationarity of multiple reward function arises when an inexperienced agent, such as a newborn baby, has to explore and subcategorize its environments from scratch to maximize rewards. In such situations, the agent probably cannot link the change in the state variables with the change in the reward function. Thus, an ideal RL robot must be able to learn accurate control to achieve high rewards under nonstationary dynamics and reward functions.

What kind of RL architecture is suitable for robots to learn under nonstationary dynamics and reward functions? We assume that nonstationary dynamics and reward functions are produced due to the unobservable discrete variables of the true system. In other words, a true system has several

sets of dynamics and reward functions and switches them implicitly. A simple and natural way for the RL robot to handle such nonstationarity is to use an estimation method for partially observable Markov decision processes (POMDPs). In fact, the POMDP approach has been successfully applied to robot tasks without complex dynamics in uncertain environments (Simmons & Koenig, 1995; Cassandra, Kaelbling, & Kurien, 1996; Thrun, Fox, Burgard, & Dellaert, 2000; Theocharous, Murphy, & Kaelbling, 2004). To solve these tasks, such off-line methods as the Baum-Welch algorithm for hidden Markov models estimates hidden states sufficiently well. However, in more dynamic control problems, even without considering reward functions at the moment, the robot has to consider a high-dimensional state space and estimate a hidden variable online. In such situations, the complex external dynamics need to be reduced using sophisticated methods (Stephens, 2007; Morimoto & Atkeson, 2009; Kajita, Kanehiro, Kaneko, Fujiwara, & Yokoi, 2003). To apply these techniques developed in robotics to the RL dynamics, it is convenient for RL robots to have a modular architecture in which dynamics and reward can be addressed independently. This independence of dynamics and reward is also critical when we think of the transfer of skills because the association of task goal and elemental actions is altered. For instance, should the robot learn everything from the very beginning again to play tennis using a squash racket?

MOSAIC is a modular control algorithm for nonstationary and nonlinear external dynamics (Wolpert & Kawato, 1998; Haruno, Wolpert, & Kawato, 1999, 2001; Sugimoto, Morimoto, Hyon, & Kawato, 2010). Although MOSAIC itself does not handle RL settings, several studies have extended it to RL. Model-based multiple-module RL (MMRL) defines a pair of a forward model, which is a local predictor of external dynamics, and an RL controller. MMRL evaluates each module based on only the accuracy of forward prediction without considering the nonstationarity of reward when switching controllers. Therefore, MMRL cannot escape from the interference of learning among RL controllers, even if the change of reward function is observable. Context detection reinforcement learning (CD-RL; da Silva, Basso, Bazzan, & Engel, 2006) defines a module as a tuple of an RL controller, a forward predictor, and a reward predictor. CD-RL can control both the nonstationary external dynamics and the reward function by evaluating each module based on the sum of the errors in forward and reward predictions. However, since modules are constructed for all possible combinations of forward and reward predictors, inefficient learning and unnecessarily huge memory requirements are inherent in this architecture. Moreover, the performance of the CD-RL architecture often falls into suboptima because the granularity of the forward models and the RL controllers is not necessarily the same. For example, more forward models might be needed for good performance than the number of RL controllers, as exemplified in our simulation below.

Here we propose a novel modular RL architecture, MOSAIC for multiple reward environments (MOSAIC-MR), which models multiple external dynamics and multiple reward functions independent of RL controllers. MOSAIC-MR provides a simple way to deal with both nonstationary external dynamics and the reward function, regardless of whether the change of the reward function is observable. MOSAIC-MR learns multiple RL controllers based on TD error and also uses the history of selected RL controllers as a prior probability by which the architecture can avoid combinatorial explosion in the parameter space.

The rest of this article is structured as follows. First, the MOSAIC-MR architecture is described in detail in section 2. In section 3, we test its performance in two types of simulations. In section 3.1, we evaluate the MOSAIC-MR architecture by a pendulum swing-up task where two different reward functions are used alternately under the influence of wind. The MOSAIC-MR architecture learns how to switch and control its environment with little interference. MOSAIC-MR's behavior is indistinguishable from the optimal solution. In section 3.2, we examine a situation where two agents collaboratively attempt to swing a pendulum up while considering the other agent's goal. This is a simple example of social interaction in which both multiple rewards and multiple dynamics play a crucial role (Ghavamzadeh, Mahadevan, & Makar, 2006; Uchibe, Asada, & Hosoda, 1996). These two simulations demonstrate the superiority of MOSAIC-MR over flat RL, MMRL, and CD-RL. Finally, in section 4, we conclude with a discussion.

2 MOSAIC-MR

MOSAIC-MR architecture, which assumes that the complex and nonstationary external world can be represented as a mixture of multiple external dynamics and multiple reward functions, decomposes the external environment into subenvironments that can be characterized by simpler dynamics and reward functions. MOSAIC-MR calculates the local control policy for all subenvironments and appropriately combines them for the current situation. The architecture consists of three sets of modules: reward, forward, and RL (see Figure 1). Here we provide a brief overview of the general functions of each module before explaining the algorithmic details (provided below). Each reward module predicts a local reward function, and each forward module predicts the local environmental dynamics. Both modules are applied to environmental state $\mathbf{x}(t) \in X \subset \Re^N$ and action $\mathbf{u}(t) \in U \subset \Re^D$. Note that the reward module predicts an immediate reward but not a future one. Therefore, the reward module differs from RL modules that compute a value function. One MOSAIC-MR characteristic is that the reward and forward modules divide a complex environment into several subenvironments that are controlled by the RL modules. The decomposition of the external environment by the reward and forward

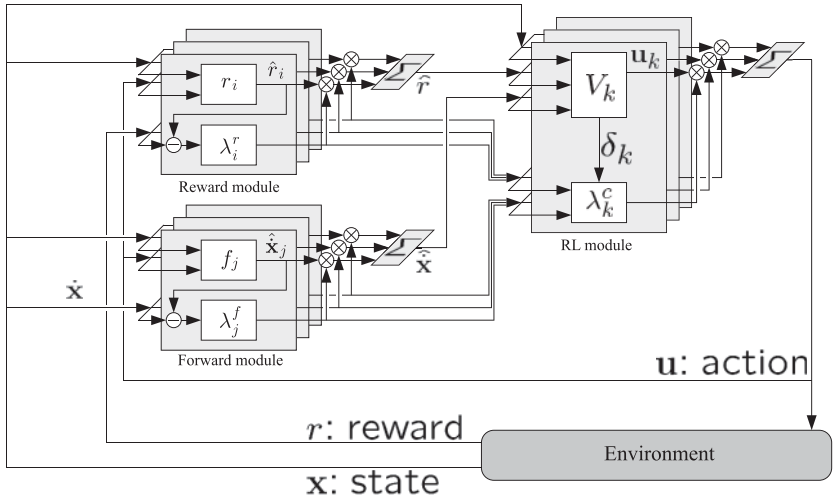

Figure 1: Schematic diagram of MOSAIC-MR. Reward modules (top left) predict the immediate reward ($\hat{r}_i$), and forward modules (bottom left) predict the local dynamics of environment ($\hat{\dot{\mathbf{x}}}_j$). Reward and forward modules decompose a complex environment into subenvironments on the basis of prediction errors ($r - \hat{r}_i$) and ($\dot{\mathbf{x}} - \hat{\dot{\mathbf{x}}}_j$), respectively. RL modules (displayed at right) output action ($\mathbf{u}_k$) appropriate for each subenvironment. The learning and control of each RL module are weighted based on TD error (δ_k), that is, on how well each RL controller predicts the expected discounted future rewards.

modules is performed on the basis of the prediction errors of the reward and the dynamics, respectively. The RL module, which receives the reward and dynamics predictions ($\hat{r}(t)$, $\hat{\dot{\mathbf{x}}}(t)$) and the weights of each module in the decomposition ($\lambda_i^r(t)$, $\lambda_j^f(t)$ defined later) from the reward and forward modules, approximates a local value function and selects an action that maximizes the local value function. Finally, MOSAIC-MR determines an action appropriate for the current status of the environment on the basis of each RL module's TD error, as described below.

In the following sections, we explain how selection and learning are performed in the reward, forward, and RL modules in that order. For simplicity, we denote them by r, f, and c and index them by $i = 1, \ldots, M^r$, and $j = 1, \ldots, M^f$ and $k = 1, \ldots, M^c$.

2.1 Reward Module. Each reward module consists of a reward model that predicts local reward function r and responsibility signal estimator $\lambda_i^r(t)$. We assumed that the reward signal at each time step can be predicted by a mixture of local reward functions. The responsibility signal, which

represents the relative preciseness of the multiple reward modules, is given by Bayes' rule:

$$\lambda_i^r(t) = \frac{P(i)q(r(t)|\, i)}{\sum_{i'=1}^{M^r} P(i')q(r(t)|\, i')}, \tag{2.1}$$

where $P(i)$ and $q(r(t)|\, i)$ are the prior probability and the likelihood.

When any specific switching feature is known a priori, we include this information in the formulation of the responsibility signal as a prior probability (Wolpert & Kawato, 1998; Haruno et al., 1999, 2001). For MOSAIC-MR, two kinds of prior knowledge are used for module selection: temporal continuity and spatial locality. Temporal continuity reduces excessively frequent switching due to noise. To express spatial locality, we applied a gaussian spatial prior. The details of these two priors are provided in appendix A.

The likelihood of reward module $q(r(t)|\, i)$ is given by the prediction error of reward $r(t) - \hat{r}_i(t)$. Here, $\hat{r}_i(t)$ is the predicted reward of reward module i conditioned on current state $\mathbf{x}(t)$ and current action $\mathbf{u}(t)$. Each reward model is represented by the quadratic functions of $\mathbf{x}(t)$ and $\mathbf{u}(t)$. When the prediction error is assumed to be gaussian with variance $(\sigma_i^r)^2$, the likelihood $q(r(t)|\, i)$ is defined as

$$q(r(t)|\, i) = \frac{1}{\sqrt{2\pi}\sigma_i^r} \exp\left[-\frac{1}{2(\sigma_i^r)^2}(r(t) - \hat{r}_i(t))^2\right], \tag{2.2}$$

$$\hat{r}_i(t) = r_i(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.3}$$

where $r_i(\mathbf{x}(t), \mathbf{u}(t))$ is the ith reward model. The learning rate of each reward module is proportional to the responsibility signal. Details regarding the prior probability and learning rule are described in appendix A.

2.2 Forward Module. The forward modules divide the environment into subenvironments based on the prediction error of the dynamics. MOSAIC-MR compares the prediction errors of each forward module and computes responsibility signal $\lambda_j^f(t)$ by Bayes' rule:

$$\lambda_j^f(t) = \frac{P(j)q(\dot{\mathbf{x}}(t)|\, j)}{\sum_{j'=1}^{M^f} P(j')q(\dot{\mathbf{x}}(t)|\, j')}. \tag{2.4}$$

Here, $q(\dot{\mathbf{x}}(t)|\, j)$ is the likelihood based on the prediction error of module j, and $P(j)$ is the prior probability of module j. By assuming that the prediction

error is gaussian with variance $(\sigma_j^f)^2$, the likelihood $q(\dot{\mathbf{x}}(t)|\, j)$ is given as

$$q(\dot{\mathbf{x}}(t)|\, j) = \frac{1}{\sqrt{2\pi}^N \left(\sigma_j^f\right)^N} \exp\left[-\frac{1}{2\left(\sigma_j^f\right)^2} \|\dot{\mathbf{x}}(t) - \hat{\dot{\mathbf{x}}}_j(t)\|^2\right], \tag{2.5}$$

$$\hat{\dot{\mathbf{x}}}_j(t) = f_i(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.6}$$

where N is the number of the dimensions of state $\mathbf{x}(t)$ and $f_i(\mathbf{x}(t), \mathbf{u}(t))$ is the jth forward model. Details regarding prior probability and the learning rule are provided in appendix A.

2.3 RL Module. RL modules receive reward prediction, forward state prediction, and responsibility signals from the reward and forward modules and learn the local value functions. Local greedy policies are computed using these value functions, and the final action is selected from among them based on the TD error. This section describes these processes in detail.

We denote the responsibility signal of an RL module as $\lambda_k^c(t)$, which is given by Bayes' rule:

$$\lambda_k^c(t) = \frac{P(k)q(\mathbf{x}(t), r(t)|\, k)}{\sum_{k'=1}^{M^c} P(k')q(\mathbf{x}(t), r(t)|\, k')}. \tag{2.7}$$

Here, $q(\mathbf{x}(t), r(t)|\, k)$ is the likelihood and $P(k)$ is the prior probability of the RL module. Likelihood $q(\mathbf{x}(t), r(t)|\, k)$ evaluates the goodness of each RL module based on TD error $\delta_k(t)$ of its local value function $V_k(t)$, which approximates an expected future reward (see appendix B). An RL module with a small squared TD error contributes considerably to the control. This is the rationale behind the definition of the responsibility signal of the RL module. By assuming that the TD error is gaussian with variance $(\sigma_k^c)^2$, likelihood $q(\mathbf{x}(t), r(t)\,|\, k)$ is given by

$$q(\mathbf{x}(t), r(t)|\, k) = \frac{1}{\sqrt{2\pi}\sigma_k^c} \exp\left[-\frac{1}{2(\sigma_k^c)^2}\delta_k(t)^2\right], \tag{2.8}$$

where $(\sigma_k^c)^2$ is the variance of the TD error. Details regarding the TD error and prior probability are described in appendixes B and C, respectively.

The control policy of each RL module is derived from the optimal control theory. Greedy action $\mathbf{u}_k(t)$ with respect to current local value function $V_k(t)$

of the kth RL module is given from the Hamilton-Jacobi-Bellman equation,[1]

$$\mathbf{u}_k(t+\Delta t) = \underset{\mathbf{u}_k}{\operatorname{argsup}} \left[\hat{r}(t) + \frac{\partial V_k(\mathbf{x}(t))}{\partial \mathbf{x}} \hat{\dot{\mathbf{x}}}(t) \right], \tag{2.9}$$

where $\hat{r}(t) = \sum_{i=1}^{M^r} \lambda_i^r(t)\hat{r}_i(t)$ and $\hat{\dot{\mathbf{x}}}(t) = \sum_{j=1}^{M^f} \lambda_j^f(t)\hat{\dot{\mathbf{x}}}_j(t)$ are the predicted reward and predicted dynamics, respectively. Here we assume that reward $r(\mathbf{x}(t), \mathbf{u}(t))$ consists of two terms: reward $r(t)$ for state $\mathbf{x}(t)$ and cost $S(t)$ for action $\mathbf{u}(t)$. We rewrite equation 2.3 as

$$r_i(\mathbf{x}(t), \mathbf{u}(t)) = r_i(\mathbf{x}(t)) - S(\mathbf{u}(t)). \tag{2.10}$$

In this case, the greedy action of the kth RL module is given based on its value function $V_k(\mathbf{x}(t))$. The following equation is derived by substituting equation 2.10 into equation 2.9 and taking a partial derivative of the right-hand side of equation 2.9 by $\mathbf{u}$:

$$\begin{aligned} 0 &= \sum_{i=1}^{M^r} \lambda_i^r \frac{\partial}{\partial \mathbf{u}} \left\{ r_i(\mathbf{x}(t)) - S(\mathbf{u}_k(t+\Delta t)) \right\} + \frac{\partial \hat{\dot{\mathbf{x}}}(t)}{\partial \mathbf{u}}^{\mathrm{T}} \frac{\partial V_k(\mathbf{x}(t))}{\partial \mathbf{x}}^{\mathrm{T}} \\ &= -\frac{\partial S(\mathbf{u}_k(t+\Delta t))}{\partial \mathbf{u}} + \frac{\partial \hat{\dot{\mathbf{x}}}(t)}{\partial \mathbf{u}}^{\mathrm{T}} \frac{\partial V_k(\mathbf{x}(t))}{\partial \mathbf{x}}^{\mathrm{T}}. \end{aligned} \tag{2.11}$$

Finally, the greedy action of the kth RL module is given by

$$\mathbf{u}_k(t+\Delta t) = S'^{-1} \left(\frac{\partial \hat{\dot{\mathbf{x}}}(t)}{\partial \mathbf{u}}^{\mathrm{T}} \frac{\partial V_k(\mathbf{x}(t))}{\partial \mathbf{x}}^{\mathrm{T}} \right), \tag{2.12}$$

where S' represents the partial differential of cost function S by action $\mathbf{u}$ (Doya, 2000). Then MOSAIC-MR weights each action $\mathbf{u}_k(t)$ with responsibility signal $\lambda_k^c(t)$ and finally executes it as its motor command:

$$\mathbf{u}(t+\Delta t) = \sum_{k=1}^{M^c} \lambda_k^c(t)\mathbf{u}_k(t+\Delta t). \tag{2.13}$$

[1] The partial derivative of a scalar by a column vector becomes a raw vector defined as $\frac{\partial V}{\partial [x_1, x_2, x_3, \ldots]^{\mathrm{T}}} = \left[\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \frac{\partial V}{\partial x_3}, \ldots \right]$. The partial derivative of a column vector by a column vector becomes a matrix: $\frac{\partial \dot{\mathbf{x}}}{\partial [u_1, u_2, u_3, \ldots]^{\mathrm{T}}} = \left[\frac{\partial \dot{x}_1}{\partial \mathbf{u}}, \frac{\partial \dot{x}_2}{\partial \mathbf{u}}, \frac{\partial \dot{x}_3}{\partial \mathbf{u}} \right]^{\mathrm{T}}$.

3 Results

We tested the learning performance of the MOSAIC-MR algorithm with a pendulum swing-up task in which multiple reward functions and multiple dynamics play a critical role. In this simulation, we examined the following specific features: decomposition of an environment into subenvironments by the reward and forward modules and adaptation of each RL module to subenvironments. We compared the performance of MOSAIC-MR with three other RL algorithms. To see the effect of the modular architecture in RL, we tested continuous-time and continuous-state-space RL (flat RL) and MMRL (multiple model-based reinforcement learning) (Doya, Samejima, Katagiri, & Kawato, 2002), the modular RL algorithm based on forward dynamics. One MMRL module consists of a forward model and an RL controller. MMRL selects a module based on the accuracy of the forward prediction, not the the TD error. Then we examined the effect of the TD error on module switching by testing context detect reinforcement learning (CD-RL) (da Silva et al., 2006), the modular RL algorithm that defines the module as a tuple of a reward module, a forward module, and an RL controller. We can also study how a flexible combination of reward and forward modules affects the RL controllers by contrasting CD-RL and MOSAIC-MR.

We used two different types of simulations: single-agent and multi-agent pendulum swing-up tasks. In both cases, the environment has two different external dynamics and reward functions. In the single-agent case, two reward functions alternate periodically at fixed intervals. In the multi-agent case, two reward function switch depending on the other agent's internal state, which is unobservable to the other. Therefore, the reward function has a more complex structure than the single-agent case.

Figure 2a shows the simulation setting. The pendulum angle is represented as θ rad, where $\theta = 0$ corresponds to the upright position. The torque is represented as T Nm. The state variable is $\mathbf{x}(t) = [\theta(t), \dot{\theta}(t)]^{\mathrm{T}}$, and the action variable is $\mathbf{u}(t) = T(t)$. The physical systems were simulated by the fourth-order Runge-Kutta method, and the learning dynamics were simulated by the Euler method. For both methods, time steps of 0.05 sec were used.

3.1 Pendulum Swing-Up by Single Agent

3.1.1 Task Setting. The MOSAIC-MR agent must learn two reward functions, R_{left} and R_{right}, which give maximum reward to the agent when the pendulum leans left and right, respectively:

$$r(\mathbf{x}(t), \mathbf{u}(t)) = \cos\left(\theta(t) - \theta_0\right) - S(\mathbf{u}(t)) \begin{cases} R_{\mathrm{left}}\colon \theta_0 = -\dfrac{45\pi}{180}\ \mathrm{rad} \\ R_{\mathrm{right}}\colon \theta_0 = \dfrac{45\pi}{180}\ \mathrm{rad} \end{cases}, \quad (3.1)$$

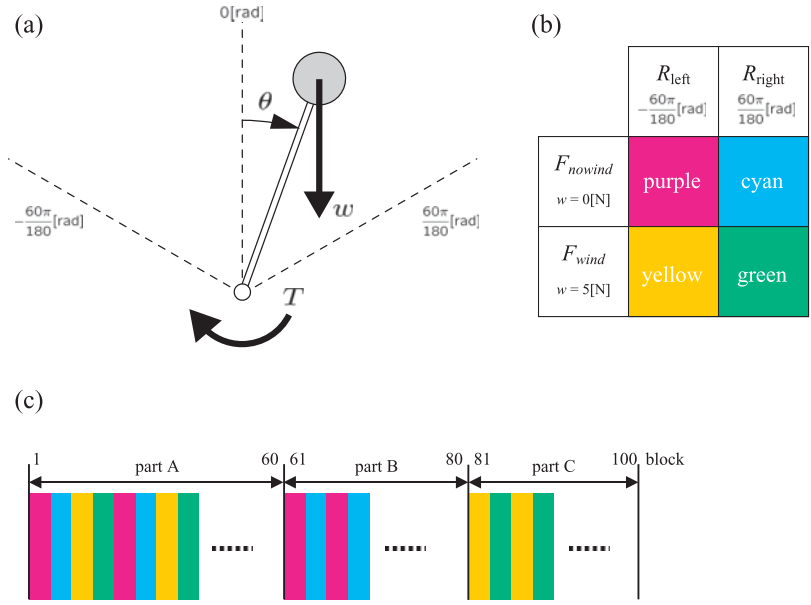

Figure 2: Simulation of swing-up pendulum. (a) The top-most position corresponds to angle 0 rad. Mass m is 1 kg; length l, 1 m; friction coefficient μ, 0.1; and acceleration of gravity g, 9.81 m/sec^2. Torque T is limited ($|T| < 15$ Nm). Simulation time step Δt is 0.05 sec, and the time derivative of state $\dot{\mathbf{x}}(t)$ is given by the Euler method. (b) There were two dynamics (F_{wind} in the presence of wind and F_{nowind} without wind) and two reward distributions (R_{left} maximum at $-\frac{60}{180}\pi$ and R_{right} maximum at $\frac{60}{180}\pi$). Thus, there are four possible conditions: purple, cyan, yellow, and green. (c) Simulation was done in three parts. Part A (1–60 blocks) comprises the alternations of all four conditions. Parts B and C (61–80 and 81–100 blocks, respectively) comprise the alternations of cyan and purple, and green and yellow, respectively.

where $S(\cdot)$ is the cost function for the action defined as

$$S(\mathbf{u}(t)) = \frac{1}{2} 0.005 T(t)^2. \tag{3.2}$$

There are two external dynamics: F_{nowind}, F_{wind}; the wind blows downward in F_{wind}, but there is no wind in F_{nowind}. The equation for both dynamics is given by

$$\ddot{\theta}(t) = \left(\frac{g}{l} + \frac{w}{ml}\right) \sin\theta(t) - \frac{\mu}{ml^2}\dot{\theta}(t) + \frac{1}{ml^2} T(t) + \mathcal{N}(0, 0.01)$$
$$\begin{cases} F_{\text{nowind}}\text{: } w = 0\,\text{N} \\ F_{\text{wind}}\text{: } w = 5\,\text{N} \end{cases}, \tag{3.3}$$

where $\mathcal{N}(0, 0.01)$ represents the system noise with mean 0 and standard deviation 0.01. m, l, μ, and g are the mass, length, friction coefficient, and gravity factors. The task consists of four subenvironments because there are two reward functions and two environmental dynamics. In this simulation, for convenience we indexed subenvironments by colors: purple, cyan, yellow, and green (see Figure 2b).

Simulation was performed in three parts (see Figure 2c) of lengths 60, 20, and 20 blocks (1 block consists of 100 trials). In part A, we alternated all four subenvironments to examine how each RL module adapts to them. In part B, only two subenvironments (purple and cyan) were alternated; the yellow and green subenvironments do not appear during this period. In part C, we alternated the yellow and green subenvironments after exposure to the purple and cyan subenvironments in part B to examine whether the agent can maintain the motor skills, learned in part A of the simulation, for the yellow and green subenvironments.

The agent can separately learn the reward and forward modules from the RL modules because supervised learning can be applied; since feedback is instantly available, the agent can learn the reward and forward modules without a large amount of trial and error. Additionally, once the learning of the reward and forward modules is completed, the agent can identify any combination of the two. Therefore, we trained the agent so that it learned the reward and forward modules with the help of randomly selected actions before starting the RL module learning. However, note that parallel learning is also possible by controlling the learning rates of the RL modules depending on the performance of the reward and forward modules.

We assumed the following quadratic and linear forms as the reward and forward modules:

$$r_i(\mathbf{x}(t), \mathbf{u}(t)) = -\frac{1}{2}Q_i(\theta(t) - \theta_i^r)^2 + q_i - \frac{1}{2}0.005T(t)^2, \tag{3.4}$$

$$f_j(\mathbf{x}(t), \mathbf{u}(t)) = A_j\mathbf{x}(t) + B_jT(t) + c_j. \tag{3.5}$$

Here, Q_i, θ_i^r, and q_i are the parameters of the ith reward module, and A_j, B_j, and c_j are the parameters of the jth forward module. We prepared four reward and forward modules and trained them using random action outputs. After the learning was completed, two quadratic functions, out of four each, covered the two peaks in the two reward functions R_{left} and R_{right}, and four linear forward models predicted dynamics F_{nowind} and F_{wind} around $\theta = 0$ and $\theta = \pi$.

3.1.2 Learning Results. After learning the reward and forward modules, the four RL modules, indexed as A, B, C, and D, also learned to control the pendulum. Each trial started at $t = 0$ sec and lasted 10 sec unless the pendulum was overrotated ($|\dot{\theta}(t)| > 4\pi$). Its initial state, $\mathbf{x}(0)$, was chosen

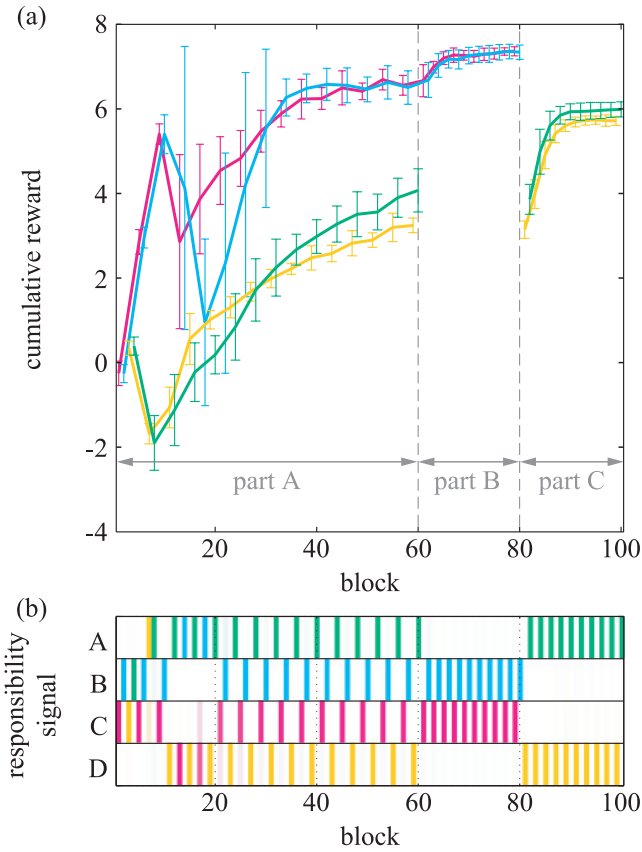

Figure 3: Learning performances of MOSAIC-MR. (a) Learning curve in terms of cumulative reward per block. Mean and standard deviation (error bar) over 10 repeated simulation runs are displayed for each block. (b) A typical example of the responsibility signal of RL modules $\lambda_k^c(t)$. In each block, the mean of $\lambda_k^c(t)$ is plotted. The color of each block represents a subenvironment, and high intensity indicates the selected frequency module.

randomly from uniform distribution ($\theta(0) \in \left[-\frac{45\pi}{180}, \frac{45\pi}{180}\right]$, $\dot{\theta}(0) = 0$]). One block consisted of a sequence of 100 trials, and the subenvironment was switched every time a new block started. The learning parameters were $\tau = 1$, $(\sigma_{A\sim D}^c)^2 = 0.01$, and $\alpha^c = 0.9$. We used a normalized gaussian network (NGNet) (Moody & Darken, 1989) to implement the value functions (see D).

Figure 3a shows the learning curve in terms of the cumulative reward per block. The mean and the standard deviation of 10 simulation runs were plotted. From part A to part B, the cumulative rewards of the purple

and cyan subenvironments gradually increased, except for the fluctuation seen in some initial blocks; the maximum values were 7.365 ± 0.107 and 7.343 ± 0.17. Due to the presence of wind, the cumulative reward of the yellow and green subenvironments in part A was significantly lower than that of purple and cyan. Note that in the initial stage of part C, the performances of the yellow and green subenvironments were maintained at the same level as the last stage of part A (yellow: $3.245 \pm 0.174 \rightarrow 3.145 \pm 0.206$, green: $4.073 \pm 0.512 \rightarrow 3.863 \pm 0.354$). The t-test showed that the performances of the yellow and green subenvironments at the end of part A and the beginning of part C were not significantly different ($p = 0.2$). In other words, the MOSAIC-MR agent successfully avoided interference between the subenvironments. Then, at the end of part C, the cumulative rewards of the yellow and green subenvironments reached maximum values of 5.719 ± 0.108 and 5.989 ± 0.178, respectively.

Figure 3b shows the result of the module switching of the RL modules. The mean of the responsibility signal of RL controllers $\lambda_k^c(t)$ is represented by the intensity of the color. The fluctuating learning process continued until the 18th block, and the mapping of the RL module and the subenvironment were fixed after the 19th block. In every block after the 19th, the MOSAIC-MR agent switched the RL module perfectly, and the mean of the responsibility signal became nearly 1 or 0. The average of the responsibility signal of the module with the maximum value was 0.949 ± 0.083, indicating that after learning, only one module was being selected almost always. After this learning, a single RL module controlled a single subenvironment; RL modules A, B, C, and D controlled the green, cyan, purple, and yellow subenvironments, respectively. Note that after learning, the RL modules aslo successfully switched within a few (one or two) trials when the subenvironment changed even if the temporal continuity prior was used.

Since TD error is the driving force of RL module selection in MOSAIC-MR, we must ensure that the selected RL controller exhibits the minimum TD error among all RL controllers. Figure 4 shows that the squared TD error of the selected RL modules in Figure 3b had minimum values and decreased consistently in each subenvironment. At the beginning of part A, the TD error of modules D (yellow) and A (green) decreased in the purple and cyan subenvironments (top panels in Figure 4), because these subenvironments are similar except for the presence of wind; the corresponding optimal control policies are also similar. In other words, the experience of similar subenvironments was efficiently used to accelerate learning, particularly in its early phase.

3.1.3 Comparison. For comparison, we simulated the same pendulum swing-up task with three other RL algorithms. First, we tested a flat RL (for details, see section 4.2 of Doya, 2000) to demonstrate the advantage of the modular RL approach. The flat RL learned the value function with a function

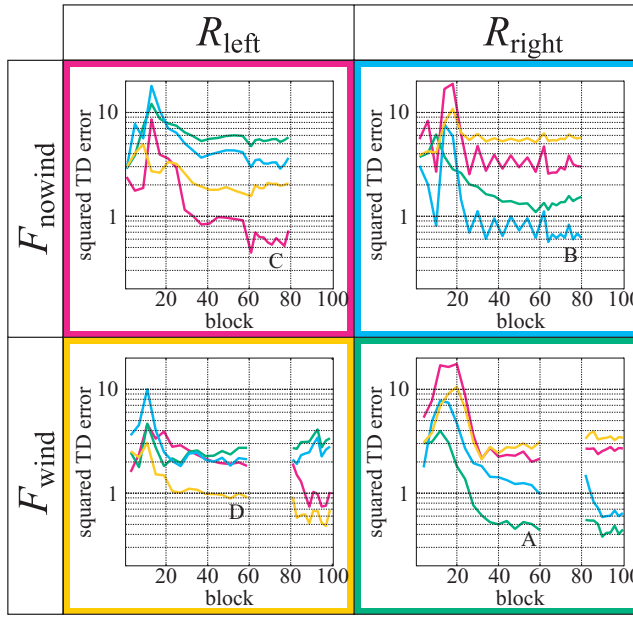

Figure 4: Learning curves in terms of TD error. TD errors of RL modules A, B, C, and D are plotted in purple, green, cyan, and yellow. The vertical axis of each graph is in log scale, and its range is 0.2 to 20.

approximator, which has the same structure and number of parameters as used by MOSAIC-RL, and its output action was in accordance with equation 2.12 (the partial differential of dynamics, $\frac{\partial \dot{\mathbf{x}}(t)}{\partial \mathbf{u}(t)}$, is derived analytically).

Figure 5a shows the learning curves in terms of the cumulative rewards. In the early stage of part A, the learning curve rises rapidly in each subenvironment because the flat RL learns only one module. However, the performances saturated at very low values because one RL controller had to control all four subenvironments and could not adapt well to each environment. In part B, the performance in the purple subenvironment exceeded the cyan subenvironment because the solution of the flat RL adapted to the left goal (R_{left}). In the early stage of part C, performance in the yellow subenvironment is relatively higher than the performance in the green subenvironment. The goals of the purple and yellow subenvironments are the same (R_{left}). Therefore, this happened because the learned parameter was transferred from the purple to the yellow subenvironment. When we analyze the performances in each purple, cyan, yellow, and green subenvironment, the final cumulative rewards (purple and cyan in part B and yellow and green in part C) were 2.657 ± 1.157, 1.133 ± 0.524, -0.564 ± 0.297, and -0.801 ± 0.384, respectively, and they were all significantly worse compared to the values obtained for MOSAIC-MR (t-test: $p < 0.01$).

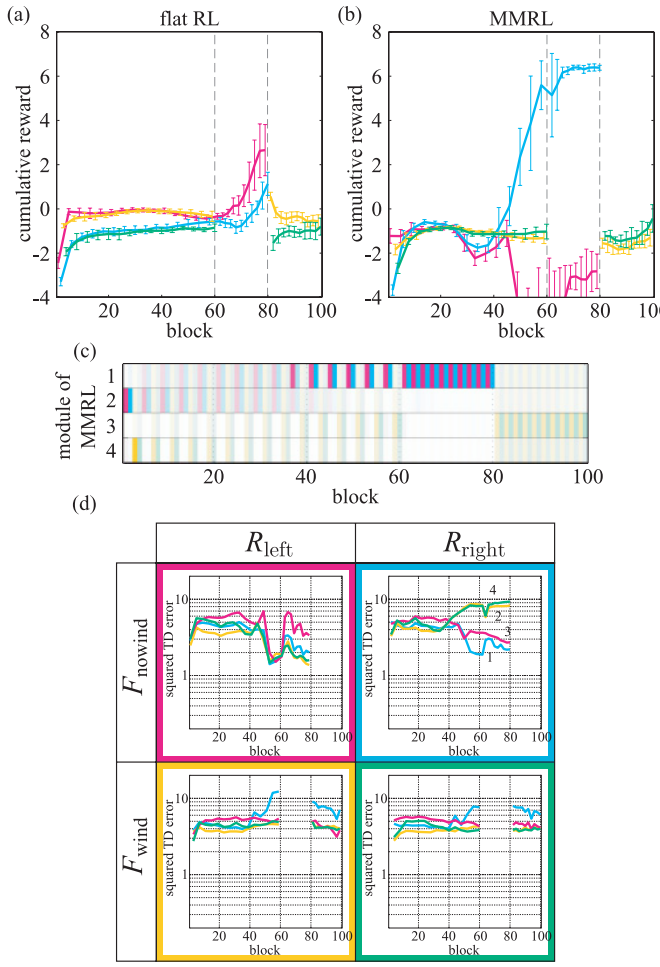

Figure 5: Learning performances of flat RL and MMRL. (a, b) Results in terms of cumulative reward in the same format as Figure 3a. (c, d) Typical examples of the responsibility signal and TD error of MMRL in the same format as Figures 3b and 4.

Next, we tested MMRL. In this simulation, the MMRL agent utilized four modules, as in the case of MOSAIC-MR, and each forward model and the RL controller had the same architecture and the same number of parameters as MOSAIC-RL.

From the last stage of part A to part B, the performance in the cyan subenvironment improved rapidly. In contrast, the performance in the purple subenvironment decreased because MMRL cannot distinguish the cyan

and purple subenvironments that have the same external dynamics but different reward functions. With wind (yellow and green subenvironments), the performance improved only slightly for two reasons: MMRL could not distinguish two goals (R_{left} and R_{right}), and control is difficult because of the influence of the wind. In the initial stage of part C, the performances of the yellow and green subenvironments were maintained at the same level as in the last stage of part A. This also shows that the MMRL architecture can distinguish only the wind dynamics. The final cumulative rewards (purple and cyan in part B and yellow and green in part C) were -2.823 ± 0.780, 6.384 ± 0.111, -1.303 ± 0.377, and -0.431 ± 0.625. We compared the postlearning performance of MMRL with that of MOSAIC-RL, and the *t*-test demonstrated that MOSAIC-RL outperformed MMRL in all subenvironments (*t*-test: $p < 0.01$).

Figure 5c shows typical examples of the responsibility signal in MMRL. Here, module 1 is mainly responsible for the left and right rewards under the no-wind dynamics after learning. Module 3 is mainly responsible for the left and right rewards under the wind dynamics. In the no-wind dynamics (purple and cyan) and the wind dynamics (yellow and green), modules 1 and 3 were mainly selected, but this separation is not perfect, and small interferences are visible. This also demonstrates that MMRL can distinguish only the change of the dynamics. In the cyan subenvironment, module 1 has the smallest TD error, which is consistent with the observation that the cyan subenvironment has the highest performance (see Figure 5b).

Finally, we tested CD-RL proposed by da Silva et al. (2006). Each reward model, forward model, and RL controller have the same architecture and the same number of parameters as in MOSAIC-RL. The coefficients in the sum of the reward and forward prediction errors were determined to balance the amplitudes of the two errors.

Figure 6a shows the results when 16 modules were used. The module is defined for all combinations of the four reward modules and the four forward modules. In part B, the performances of the purple and cyan subenvironments increased similarly. In the last stage of part A and the initial stage of part C, the performance of the yellow and green subenvironments remained at the same level. These results indicate that CD-RL can distinguish the four subenvironments. However, the following reasons seem to cause low performance in comparison with MOSAIC-MR. Around $\theta = 0$ and π (standing and hanging positions), different dynamics models are used due to the strong nonlinearity of the dynamics and the complexity of the reward function. Because the RL controllers of CD-RL are paired with their environmental models, multiple RL controllers were unnecessarily switched within a trial to approximate the cumulative reward. With this switch, RL controllers cannot accurately evaluate future cumulative reward. Figure 6b represents the results when only four RL controllers are available (the same number as the MOSAIC-MR simulation). Compared to the case of 16 modules, the standard deviation and the performance differences

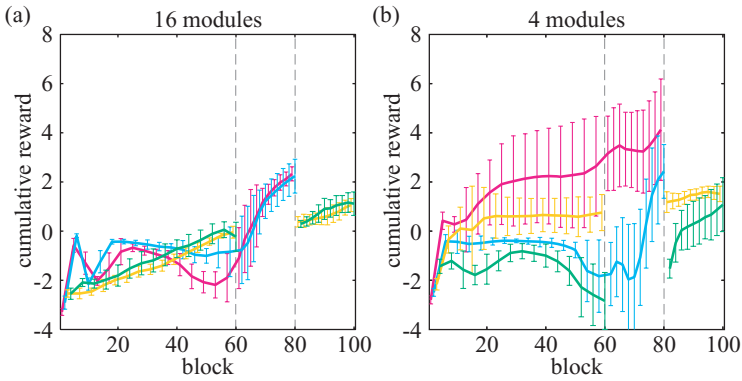

Figure 6: Learning performances of CD-RL in pendulum swing-up task. Mean and standard deviation of the cumulative reward in each block over 10 simulation runs are plotted. (a, b) The results when there were 16 and 4 modules.

between the subenvironments with identical external dynamics but different goals (purple and cyan, yellow and green) are very large. Since CD-RL has less flexibility than MOSAIC-MR, CD-RL needs more than four controllers. These results demonstrate the efficiency of a flexible combination of the reward and forward models used in MOSAIC-MR.

3.2 Collaborative Swing-Up by Two Agents

3.2.1 Task Setting. In the previous sections, MOSAIC-MR outperformed the conventional methods when the reward distribution was changed by some external mechanism. In this section, we discuss the situation where such a shift of reward distribution occurs naturally during interactions between two agents. Agents I and II collaboratively swing a single pendulum up. More specifically, agent II cannot swing the pendulum up by itself. The task for agent I is to estimate the goal of agent II and maximize the amount of reward by supporting it.

Figure 7a delineates the simulation setting. Agent II has a simple feedback control policy $T_{\mathrm{II}}(t) = -K(\mathbf{x}(t) - \mathbf{x}_0)$, where K and $\mathbf{x}_0$ represent the feedback gain and target posture. When the absolute value of the angle exceeds $\frac{\pi}{2}$, K is set to $[-0.1, -0.1]$, $\mathbf{x}_0 = [\pi, 0]^{\mathrm{T}}$. This enables the pendulum to be pushed away from the downward posture ($\theta = \pi$ rad). Otherwise K is fixed as $K = [1, 1]$, where $\mathbf{x}_0$ is the target posture where the reward function of agent II takes maximum value. [2] Since the maximum torque of agent II is low ($|T_{\mathrm{II}}| \leq 5 <$ mg Nm), agent II cannot achieve the target by

[2] If agent II takes R_{left}, the posture $\mathbf{x}_0$ is $[-\frac{60}{180}\pi, 0]^{\mathrm{T}}$. Otherwise ($R_{\mathrm{right}}$), $\mathbf{x}_0$ is $[\frac{60}{180}\pi, 0]^{\mathrm{T}}$.

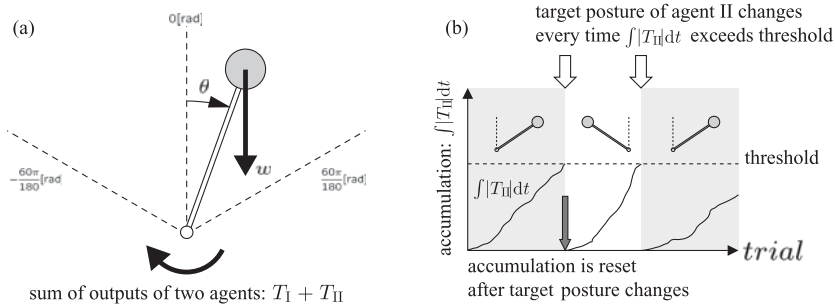

Figure 7: Pendulum swing-up by two agents. (a) Settings are the same as in Figure 2a. Summation of the torques applied by both agents I and II ($T_{\mathrm{I}} + T_{\mathrm{II}}$) drives the pendulum. (b) Goal of agent II alternated depending on the accumulation of the absolute value of its torque ($\int |T_{\mathrm{II}}| \mathrm{d}t$) (fatigue). Wind also changed every 100 blocks. Agent I had to detect these changes and adapt.

itself. Therefore, the assistance of agent I is indispensable. The torques from agents I and II are combined, and the output is applied to the pendulum ($\mathbf{u} = T_{\mathrm{I}} + T_{\mathrm{II}}$). However, agent I cannot directly observe the actual torque of agent II (T_{II}); therefore, it has to estimate the actual torque of agent II (T_{II}) using its forward models in an inverse manner. Agent I estimates the local output of agent II from the observed state and its derivative ($\mathbf{x}(t)$ and $\dot{\mathbf{x}}(t)$) and its own output T_{I} by using inverse transformation of linear forward model. First, each forward model j makes estimates $T_{\mathrm{II},j}(t - \Delta t)$) based on its dynamical equation:

$$\dot{\mathbf{x}}_j(t) = A_j \mathbf{x}(t - \Delta t) + B_j(T_{\mathrm{I}}(t - \Delta t) + T_{\mathrm{II}}(t - \Delta t)) + c_j. \tag{3.6}$$

In other words,

$$\hat{T}_{\mathrm{II},j}(t - \Delta t) = \left(B_j^{\mathrm{T}} B_j\right)^{-1} B_j^{\mathrm{T}} (\dot{\mathbf{x}}(t) - A_j \mathbf{x}(t - \Delta t) - c_j) - T_{\mathrm{I}}(t - \Delta t). \tag{3.7}$$

Then to obtain the appropriate estimated value of $T_{\mathrm{II}}(t - \Delta t)$, the estimates obtained from each forward module are weighted by the corresponding responsibility signal $\lambda_j^f(t - \Delta t)$:

$$\hat{T}_{\mathrm{II}}(t - \Delta t) = \sum_{j=1}^{M^f} \lambda_j^f(t - \Delta t) \hat{T}_{\mathrm{II},j}(t - \Delta t). \tag{3.8}$$

In parallel with the previous simulations, the goal of agent II is either R_{left} (swing up left) or R_{right} (swing up right), and the environmental dynamics

consist of F_{nowind} in the absence of wind and F_{wind} in the presence of wind. Agent II changed its goal (i.e., R_{left} or R_{right}) due to fatigue every time its cumulative output torque reached 5000[Nm · s]. The wind dynamics changed every 100 trials. Under these conditions, agent I had to maximize its reward. We ran 10,000 trials in this simulation, and all parameters were set identical to those in the single-pendulum simulations.

3.2.2 Learning Results. Figure 8a shows learning curves in terms of the cumulative reward. The mean and standard deviations over 10 simulation runs are plotted. Learning in the purple and cyan subenvironments converged around the 3000th trial, and the maximum cumulative rewards for the purple, cyan, yellow, and green subenvironments were 7.325 ± 0.169, 7.324 ± 0.085, 4.69 ± 0.179, and 4.961 ± 0.156. The accumulated absolute values of agent II's torque gradually became saturated (see the top panel of Figure 8b). Its output torque became low probably because agent I learned how to collaborate with agent II. The bottom panel of Figure 8b shows the responsibility signal of agent I. The color and its intensity delineate a subenvironment and its amplitude of responsibility, respectively. RL modules A, B, C, and D successfully controlled the green, yellow, cyan, and purple subenvironments.

Figure 8c shows the learning curves in terms of TD error. In each subenvironment, the selected RL module exhibited the minimum TD error among all the RL modules. These results indicate that MOSAIC-MR can work efficiently in this cooperative multi-agent task and select an appropriate control policy on the basis of the estimated intention and the control signals of agent II.

3.2.3 Comparison. Figure 9 shows the learning results for the same simulation performed by MMRL in exactly the same format as shown in Figure 8. Figure 9a shows the learning curve in terms of the cumulative reward. MMRL produced almost flat learning curves and a large standard deviation, suggesting that this two-agent simulation is truly difficult for MMRL. The maximum value of the cumulative reward of each subenvironment, namely, purple, cyan, yellow, and green, reached 5.448 ± 1.479, 2.935 ± 2.576, 2.261 ± 2.912, and 1.103 ± 1.646. After comparing the performance of MMRL with that of MOSAIC-MR using the *t*-test, we found that MOSAIC-RL outperformed MMRL in all subenvironments ($p < 0.01$).

Figure 9b shows the responsibility signal of MMRL, the changes in the goal of agent II, and the switching of dynamics. Figure 9c shows the TD error. These figures are plotted in the same format as in Figures 8b and 8c. The results of this collaborative task involving two agents show that agent I could not clearly distinguish between two different dynamics due to the noisy sensation of the states. This was probably because MMRL is incapable of dissociating the two reward functions, which complicates the separation of the two dynamics. In other words, the MMRL trajectory is

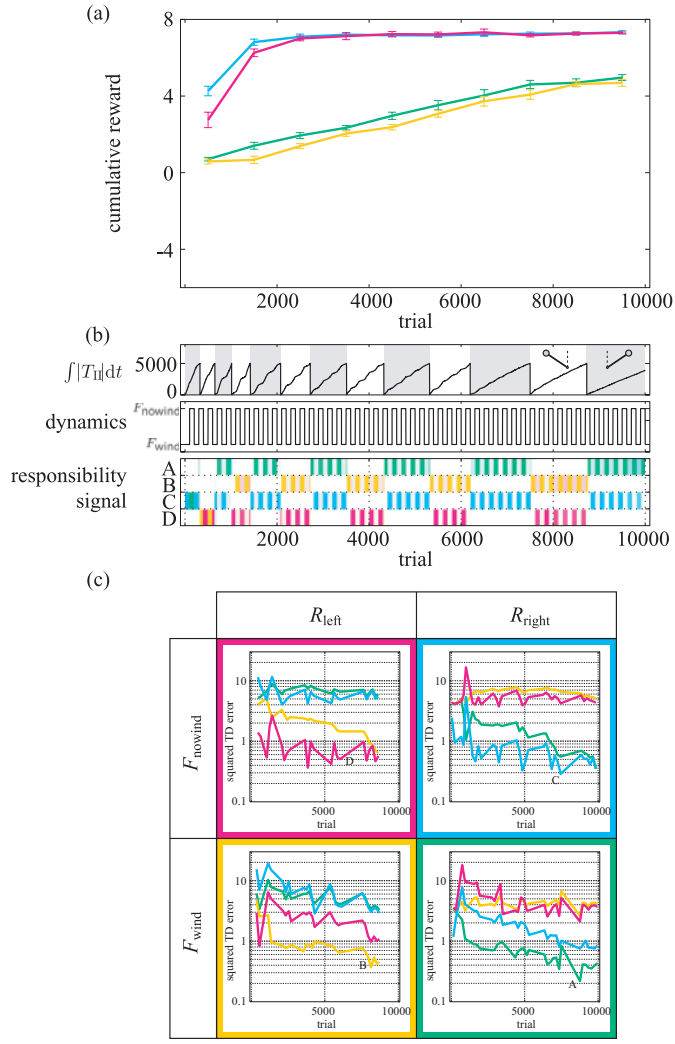

Figure 8: Results of collaborative pendulum swing-up task by two agents. (a) Average learning curve of MOSAIC-MR over 10 runs for agent I in terms of cumulative rewards showing how learning under each condition (purple, cyan, yellow, and green) proceeded. Mean and standard deviation are plotted over 1000 trials and 10 simulation runs. (b) Example of module switching obtained from one run. Top-most and middle panels exemplify the changes in the goal of agent II and the wind dynamics, respectively. The bottom panel displays the changes in four responsibility signals for agent I. (c) Learning curves in terms of TD errors are plotted in the same format as in Figure 4. The range of vertical axes is 0.1 to 30.

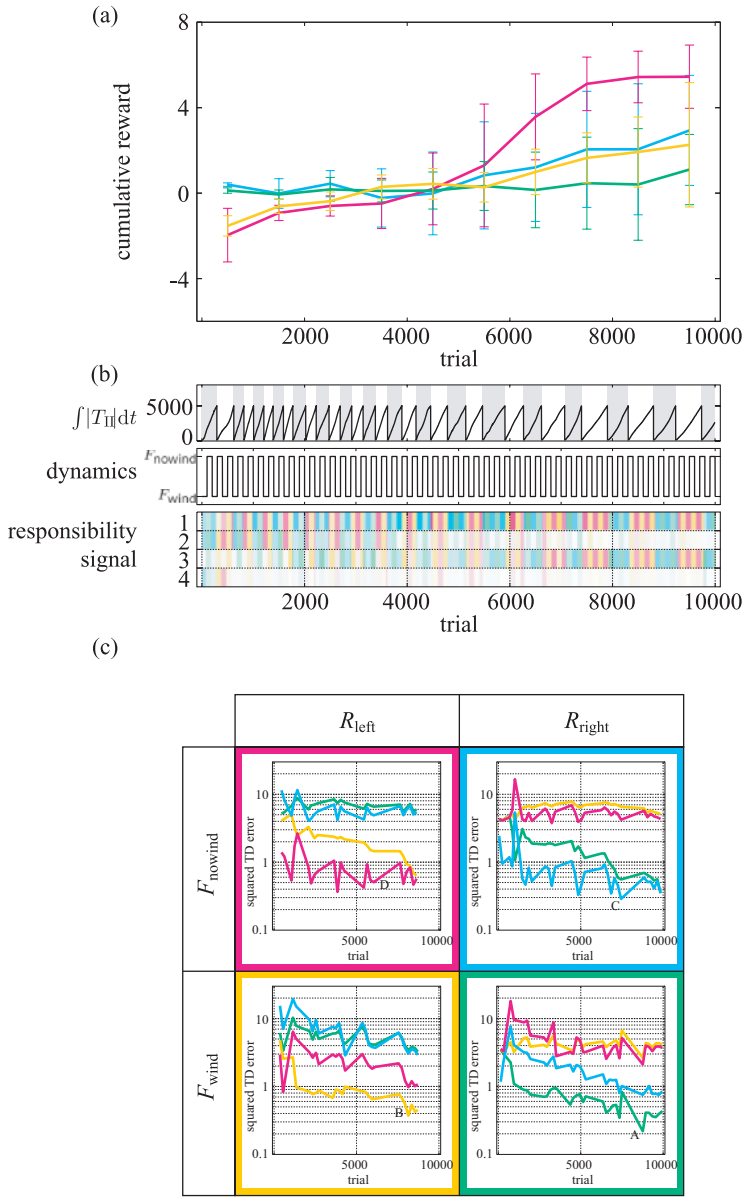

Figure 9: Results of collaborative task involving two agents in MMRL. (a) Average learning curve in terms of cumulative reward of MMRL for agent I. (b) Example of module switching obtained from one run. Top and middle panels exemplify changes in the goal of agent II and the wind dynamics. Bottom panel shows the responsibility signal of agent I. (c) Learning curves in terms of TD error. Each panel is displayed in same format as in Figure 8.

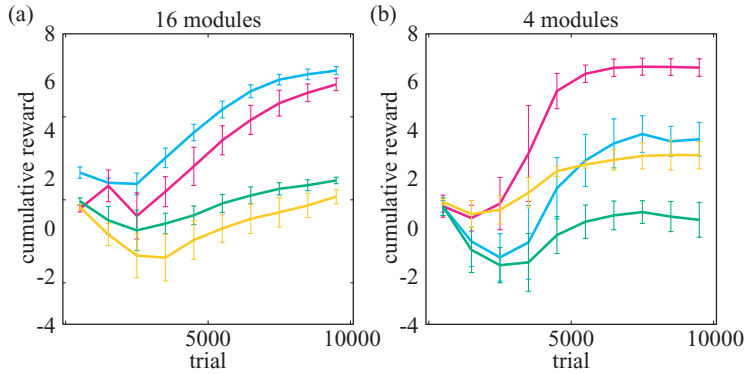

Figure 10: Learning performances of CD-RL in collaborative experiment. Mean and standard deviations of cumulative reward in 1000 trials over 10 repeated simulation runs are plotted. (a, b) Results when number of modules were 16 and 4, respectively.

close to the pendulum's upright position because the intermediate solution corresponding to the two reward (R_{left}, R_{right}) and forward models of the two dynamics, as learned by MMRL, coincides at $\theta = 0$ rad. Even if the difference in the strength of the wind is large, the difference in the dynamics around $\theta = 0$ rad becomes very small and difficult to detect. Correspondingly, Figure 9c shows that the TD error in module 3 always has the minimum value, and modular learning failed in this simulation.

Finally, we tested CD-RL using this collaborative task. Figure 10a delineates the results when 16 modules are used. In all subenvironments, although the performances gradually improved, they still peaked at a lower value than MOSAIC-MR. In CD-RL, since the sequence of module switching completely depends on the forward and reward models fixed in each module, unnecessary switching of the RL controllers sometimes happened. That is why CD-RL cannot improve the performance as much as the MOSAIC-MR can. Moreover, the learning speed was relatively slow compared to MOSAIC-MR because the number of parameters was huge (quadruple as many as MOSAIC-MR). Figure 10b shows the results when only 4 modules were used. The learning speed was faster than the case of 16 modules. However, the performances of the cyan subenvironment compared to the purple subenvironment, and the green subenvironment compared to the yellow subenvironment were much lower. These results can be explained by the same reason as in the single-agent case: the number of RL modules is too few to achieve the goal. These results demonstrate that the flexibility of MOSAIC-MR also works efficiently in the multi-agent environment.

In summary, MOSAIC-MR outperformed conventional modular RL methods under multiple external dynamics and reward functions. In

comparison with MMRL, MOSAIC-MR's reward modeling plays a crucial role. In comparison with CD-RL, its module structure that flexibly links the reward module, forward module, and the RL controller is a clear merit.

4 Discussion

This article proposes MOSAIC-MR architecture to achieve efficient RL in a complex environment that can be modeled by multiple dynamics and reward functions. MOSAIC-MR automatically detects changes in the reward function and the environmental dynamics, decomposes the environment into subenvironments, and adapts multiple RL modules to those subenvironments. It flexibly combines multiple reward modules and forward modules without a combinatorial explosion of the parameters of RL modules. MOSAIC-MR was tested by simulating the pendulum swing-up problem. We also examined a situation where two agents collaboratively swung a pendulum up and compared the MOSAIC-MR performance with three RL algorithms: flat RL, MMRL, and CD-RL for the single-agent case and MMRL and CD-RL for the multi-agent case. MOSAIC-MR learned and controlled the subenvironments with little interference and outperformed the previously proposed methods because the reward model, the forward model, and the RL controller were constructed independently and combined flexibly.

Our simulations highlighted two key features of MOSAIC-MR. First, the comparison with MMRL clarified that module selection based on forward dynamics is insufficient for the RL controllers; switching based on TD error is also needed. The TD errors of MOSAIC-MR in our simulations well reflected the changes in both the rewards and dynamics (see Figures 4 and 8c). Second, flexible Bayesian integration of reward and dynamics improves performance. In contrast with MMRL and CD-RL, where the assignments of an environmental (reward and forward) model or models and a controller are fixed a priori, MOSAIC-MR is more flexible and easily adjusts to new applications. In other words, in MOSAIC-MR, the RL modules receive only an abstracted form of the prior probabilities from the reward and forward modules and do not need to access the detailed specifications of the task, such as the detailed trajectory of movements or the amount of the reward. When the RL controller is to be adapted for a new task, only the reward and forward modules have to be modified. Therefore, we can train the reward and forward models prior to the learning of the RL controllers, as in our simulations in section 3.1.1.

In general, flexibility and stability are trade-off problems. When RL controllers and environmental models were learned simultaneously from scratch, MOSAIC-MR's flexible architecture may have negative effects because the number of free parameters required for the learning exceeds those for MMRL and CD-RL. In such a situation, the sequential application of these methods might be a plus for practical implementation. That is, forward models are trained using MMRL, and then CD-RL learners reward

modules using the parameters learned by MMRL as its initial values. Then the RL module of CD-RL can provide initial parameters for MOSAIC-MR, and MOSAIC-MR finally adjusts the details of the RL controllers. For this scenario, future study must determine when and how to switch learning algorithms.

The need for multiple reward models arises when an action in one particular (observable) state can lead to a set of different rewards. As evident in the agent-interaction simulation, such a condition occurs when several unobservable goals exist. In general, there are two approaches to deal with this problem. One is hierarchical RL (Barto & Mahadevan, 2003; Haruno, Wolpert, & Kawato, 2003), where abstract higher (observable and unobservable) states are introduced to temporarily describe the long transitions of the elementary states and are associated with the reward. The other approach uses the multiple reward system, as described in this study. Hierarchical RL is particularly efficient when we can easily define subbehavior or subtasks (chunks of primitive actions) to achieve an overall goal. Multiple reward models are preferable when several overall goals are easily defined and implicitly affect the sequence of the actions of an agent. To our knowledge, MOSAIC-MR is the first to adopt a multiple reward approach to guide the sequential selection of action, which becomes possible by considering multiple dynamics.

In this article, we have focused on the importance of MOSAIC-MR in solving engineering problems. we have also proposed that in the neuroscience of decision making, the value function is processed in the basal ganglia and frontal cortex and regulated by the reward prediction error carried by the midbrain dopaminergic projection. Furthermore, it has recently been reported that the cerebellum, one of the key parts of the brain involved in dynamics learning, communicates with the basal ganglia (Hoshi, Tremblay, Féger, Carras, & Strick, 2005). These findings suggest that the computations of the three different components in MOSAIC-MR—reward, RL controller, and dynamics (Mehta & Schaal, 2002)—are implemented in different brain areas and that these neural substrates interact with each other. Another future direction for work is the computational mechanism of hierarchical and modular RLs in the brain (Kawato & Samejima, 2007; Haruno & Kawato, 2006).

Appendix A: Prior Knowledge of Selecting Reward and Forward Modules

When any specific switching feature is known a priori from experience, we can include this information in the formulation of the responsibility signal as a prior probability (Wolpert & Kawato, 1998; Haruno et al., 2001). In the MOSAIC algorithms, two kinds of prior knowledge have been used for module selection: temporal continuity and spatial locality. These two types of prior knowledge are general features and are not task dependent.

Temporal continuity reduces excessively frequent switching due to noise. The prior for this constraint is given by $P(i)$:

$$P(i) \propto \frac{1}{\sqrt{2\pi}\sigma_i^r} \exp\left[-\frac{1}{2(\sigma_i^r)^2} E_i^r(t - \Delta t)\right]. \tag{A.1}$$

$E_i^r(t)$ represents a low-passed square error of the ith reward module at time t,

$$\begin{aligned} E_i^r(t) &= \sum_{h=0}^{t/\Delta t} \alpha^{h\Delta t} (r(t - h\Delta t) - \hat{r}_i(t - h\Delta t))^2 \Delta t \\ &= (r(t - h\Delta t) - \hat{r}_i(t - h\Delta t))^2 \Delta t + \alpha^{\Delta t} E_i^r(t - \Delta t), \end{aligned} \tag{A.2}$$

where $0 < \alpha < 1$ is a parameter that controls the strength of the temporal continuity and Δt is a time step of observation and control.

Since a gaussian spatial prior is used to implement spatial locality, prior probability $P(i)$ is formulated as follows:

$$P(i) \propto \frac{1}{\sqrt{2\pi|\Sigma_i^r|}} \exp\left[-\frac{1}{2}(\mathbf{x}(t - \Delta t) - \mathbf{x}_i^r)^{\mathrm{T}} \left(\Sigma_i^r\right)^{-1} \left(\mathbf{x}(t - \Delta t) - \mathbf{x}_i^r\right)\right], \tag{A.3}$$

where $\mathbf{x}_i^r$ is the center of the area of localization and Σ_i^r is a covariance matrix that specifies the area in the ith reward module. If both temporal continuity and spatial locality need to be met, we implement the prior as the product of equations A.1 and A.3. We pursue this approach in the simulations.

Next, we briefly explain how the modules are learned. The objective of learning in each module is to minimize the weighted approximation error. If each approximator is represented in a linear form $\hat{r}_i(t) = W_i^r \mathbf{z}(t)$,[3] the expected values for localization center $\mathbf{x}_i^r$ of the ith reward module and regression parameters W_i^r can be updated using a standard EM algorithm for gaussian mixtures (Bilmes, 1998).

The prior probability of forward module selection $P(j)$ is also given on the basis of temporal continuity and spatial locality:

$$P(j) \propto \frac{1}{\sqrt{2\pi}^N (\sigma_j^f)^N} \exp\left[-\frac{1}{2(\sigma_j^f)^2} E_j^f(t - \Delta t),\right] \tag{A.4}$$

[3] $\mathbf{z}(t)$ is a vector with the features of the reward function. For example, $\mathbf{z}(t) = \left[\theta^2, 1\right]^{\mathrm{T}}$, and W_i^r is a linear coefficient vector in the pendulum simulations described in this article.

$$P(j) \propto \frac{1}{\sqrt{2\pi}^N \sqrt{|\Sigma_j^f|}} \exp \left[-\frac{1}{2} \left(\mathbf{x}(t-\Delta t) - \mathbf{x}_j^f \right)^{\mathrm{T}} \left(\Sigma_j^f \right)^{-1} \left(\mathbf{x}(t-\Delta t) - \mathbf{x}_j^f \right) \right], \tag{A.5}$$

where E_j^f is the low-passed signal of squared prediction error $\|\mathbf{x}(t) - \hat{\mathbf{x}}_j(t)\|^2$ and $(\sigma_j^f)^2$ is the variance of the prediction error. $\mathbf{x}_i^r$ is the center of the area of localization, and Σ_j^f is a covariance matrix that specifies the area in the jth forward module.

Appendix B: TD Error

For continuous time, the value function is defined as

$$V(\mathbf{x}(t)) = \int_t^\infty e^{-\frac{s-t}{\tau}} r(\mathbf{x}(s), \mathbf{u}(s)) \mathrm{d}s. \tag{B.1}$$

Here, τ is a parameter that determines how far the agent looks ahead (timescale) to its future rewards. By dividing the integral into two parts, $[t, t+\Delta t]$ and $[t+\Delta t, \infty]$ and Taylor expanding, we have the following equation:

$$\frac{1}{\Delta t}(1 - e^{-\frac{\Delta t}{\tau}}) V(\mathbf{x}(t)) = r(\mathbf{x}(t), \mathbf{u}(t)) + e^{-\frac{\Delta t}{\tau}} \dot{V}(\mathbf{x}(t)). \tag{B.2}$$

Finally, we have the following equation by taking Δt to zero:

$$\frac{1}{\tau} V(\mathbf{x}(t)) = r(\mathbf{x}(t), \mathbf{u}(t)) + \frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}} \dot{\mathbf{x}}(t). \tag{B.3}$$

If the estimated value function is perfect, it should satisfy this consistency. If this condition is not satisfied, the prediction should be adjusted to decrease the inconsistency:

$$\delta(t) = r(t) - \frac{1}{\tau} V(\mathbf{x}(t)) + \frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}} \dot{\mathbf{x}}(t). \tag{B.4}$$

This is the TD error in the continuous-time case (Doya, 2000).

In MOSAIC-MR, the TD error of local value function $V_k(t)$ is obtained using predicted reward $\hat{r}(t)$ and predicted state $\hat{\dot{\mathbf{x}}}(t)$:

$$\delta_k(t) = \hat{r}(t) - \frac{1}{\tau} V_k(t) + \frac{\partial V_k(\mathbf{x}(t))}{\partial \mathbf{x}} \hat{\dot{\mathbf{x}}}(t), \tag{B.5}$$

where $\hat{r}(t)$ and $\hat{\mathbf{x}}(t)$ are the predicted reward and predicted dynamics. These predictions are given as weighted sums of all reward and forward models:

$$\hat{r}(t) = \sum_{i=1}^{M^r} \lambda_i^r(t)\hat{r}_i(t), \tag{B.6}$$

$$\hat{\mathbf{x}}(t) = \sum_{j=1}^{M^f} \lambda_j^f(t)\hat{\mathbf{x}}_j(t). \tag{B.7}$$

When these predictions are used, the effect of the observation noise is reduced.

Appendix C: Prior Knowledge of Selecting an RL Module

We explain the prior probability in the responsibility signal of the RL module. Generally TD error has a large variance because it is sensitive to both the performance of the reward predictor and the noise. Therefore, we introduce two prior probabilities for RL module selection. First, we incorporate temporal continuity in a manner similar to that for the reward and forward modules. From responsibility signal $P(k)$, we obtain

$$P(k) \propto \frac{1}{\sqrt{2\pi}\sigma_k^c} \exp\left[-\frac{1}{2(\sigma_k^c)^2} E_k^c(t - \Delta t)\right], \tag{C.1}$$

where $E_k^c(t)$ represents the low-passed square TD error:

$$\begin{aligned} E_k^c(t) &= \sum_{h=0}^{t/\Delta t} (\alpha^c)^{h\Delta t} \delta_k(t - h\Delta t)^2 \Delta t \\ &= \delta_k(t)^2 \Delta t + (\alpha^c)^{\Delta t} E_k^c(t - \Delta t), \end{aligned} \tag{C.2}$$

where $0 < \alpha^c < 1$ is a parameter that controls the strength of the temporal continuity. As a second prior, we incorporate a selection frequency that represents the number of times the RL module was selected.

For each combination of the three kinds of modules—the reward, forward, and RL modules—MOSAIC-MR counts the selection frequency of the combination:

$$\Lambda_{i,j,k}(t) = \int_0^t \lambda_i^r(s)\lambda_j^f(s)\lambda_k^c(s)\mathrm{d}s. \tag{C.3}$$

The initial values of $\Lambda_{i,j,k}$ are zero, and the combinations of reward, forward, and RL modules that are selected frequently have high $\Lambda_{i,j,k}$ values. We utilized the $\Lambda_{i,j,k}$ values marginalized by the current responsibility signals of the reward and forward modules as a prior probability of selection frequency:

$$P(k) \propto \sum_{i=1}^{M^r} \sum_{j=1}^{M^f} \Lambda_{i,j,k}(t) \lambda_i^r(t) \lambda_j^f(t). \tag{C.4}$$

Although this formulation of the prior is a heuristics, it can capture the abstract information of module switching and has performed well in several simulations.

Appendix D: Implementation of RL Modules

We implemented a normalized gaussian network (NGNet) (Moody & Darken, 1989) to approximate the value function of each RL module:

$$V_k(\mathbf{x}(t)) = \sum_{h=1}^{H} w_{k,h}^c \phi_h(\mathbf{x}), \tag{D.1}$$

where $h = 1, \ldots, H$, $w_{k,h}^c$, and ϕ_h are the index of the basis, the learning parameter, and the activation function. Activation function ϕ_h is given by gaussian distribution, and their sum is normalized ($\sum_{h=1}^{H} \phi_h(\mathbf{x}) = 1$ for any $\mathbf{x}$).

The learning rule for each value function resembles standard TD learning. By using a weighted TD error with responsibility signals, each value function is expected to be localized in each subenvironment. When the TD error of the kth value function is given by equation B.5 at time t, we minimize the following objective function:

$$D_k(t) = \frac{1}{2} \lambda_k^c(t) \delta_k(t)^2. \tag{D.2}$$

Each value function updates the parameter by a gradient descent method (Doya, 2000):

$$\dot{w}_{k,h}^c = \eta^c \frac{\partial D_k}{\partial w_{k,h}^c}, \tag{D.3}$$

$$= \eta^c \lambda_k^c(t) \delta_k(t) \left[-\frac{1}{\tau} \frac{\partial V_k(\mathbf{x}(t))}{\partial w_{k,h}^c} + \frac{\partial}{\partial w_{k,h}^c} \frac{\partial V_k(\mathbf{x}(t))}{\partial \mathbf{x}} \dot{\mathbf{x}}(t) \right], \tag{D.4}$$

where $w^c_{k,h}$ is the parameter of the RL module and η^c is the learning rate.

We can consider the exponentially weighted integral of the derivatives as the eligibility trace for updates of parameter $w^c_{k,h}$. Then the update rule is derived as

$$\dot{w}^c_{k,h} = \eta^c \lambda^c_k(t)\delta_k(t)e_{k,h}(t) \tag{D.5}$$

$$\dot{e}_{k,h}(t) = -\frac{1}{\kappa}e_{k,h}(t) + \frac{\partial V_k(\mathbf{x}(t))}{\partial w^c_{k,h}}, \tag{D.6}$$

where $0 < \kappa \leq \tau$ is the time constant parameter of an eligibility trace (Doya, 2000).

Acknowledgments

This study was supported by SRPBS project Development of Brain Machine Interface by Japan MEXT. M.H. was supported by PRESTO/JST and KAKENHI (22300139).

References

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*, 341–379.

Bilmes, J. A. (1998). *A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models*. (Tech. Rep.). Berkeley, CA: International Computer Science Institute.

Cassandra, A. R., Kaelbling, L. P., & Kurien, J. A. (1996). Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 963–972). Piscataway, NJ: IEEE.

da Silva, B. C., Basso, E. W., Bazzan, A.L.C., & Engel, P. M. (2006). Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 217–224). New York: ACM.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, *12*, 219–245.

Doya, K., Samejima, K., Katagiri, K., & Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Computation*, *14*, 1347–1369.

Ghavamzadeh, M., Mahadevan, S., & Makar, R. (2006). Hierarchical multi-agent reinforcement learning. *Journal of Autonomous Agents*, *13*(2), 197–229.

Haruno, M., & Kawato, M. (2006). Heterarchical reinforcement-learning model for integration of multiple cortico-striatal loops: fMRI examination in stimulus-action-reward association learning. *Neural Networks*, *19*, 1242–1254.

Haruno, M., Wolpert, D. M., & Kawato, M. (1999). *Multiple paired forward-inverse models for human motor learning and control*. Cambridge, MA: MIT Press.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Computation*, *13*, 2201–2220.

Haruno, M., Wolpert, D. M., & Kawato, M. (2003). *Hierarchical MOSAIC for movement generation*. International Congress Series, *1250*, 575–590.

Hoshi, E., Tremblay, L., Féger, J., Carras, P. L., & Strick, P. L. (2005). The cerebellum communicates with the basal ganglia. *Nature Neuroscience*, *8*(11), 1491–1493.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., & Yokoi, K.H.K. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1620–1626). Piscataway, NJ: IEEE.

Kawato, M., & Samejima, K. (2007). Efficient reinforcement learning: Computational theories, neuroscience and robotics. *Current Opinion in Neurobiology*, *17*, 205–212.

Mehta, B., & Schaal, S. (2002). Forward models in visuomotor control. *Journal of Neurophysiology*, *88*, 942–953.

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, *1*(2), 281–294.

Morimoto, J., & Atkeson, C. G. (2009). Nonparametric representation of an approximated Poincaré map for learning biped locomotion. *Autonomous Robots*, *27*(2), 131–144.

Simmons, R., & Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '95)* (pp. 1080–1087). San Francisco: Morgan Kaufmann.

Stephens, B. (2007). Humanoid push recovery. In *The IEEE-RAS 2007 International Conference on Humanoid Robots* (pp. 589–595). Piscataway, NJ: IEEE.

Sugimoto, N., Morimoto, J., Hyon, S., & Kawato, M. (2010). eMOSAIC model for humanoid robot control. In *Proceedings of From Animals to Animats 11: 11th International Conference on Simulation of Adaptive Behavior (SAB 2010)* (pp. 447–457). Berlin: Springer.

Theocharous, G., Murphy, K., & Kaelbling, L. P. (2004). Representing hierarchical POMDPs as DBNs for multi-scale robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1045–1051). Piscataway, NJ: IEEE.

Thrun, S., Fox, D., Burgard, W., & Dellaert, F. (2000). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, *128*(1–2), 99–141.

Uchibe, E., Asada, M., & Hosoda, K. (1996). Strategy classification in multi-agent environment: Applying reinforcement learning to soccer agents. In *ICMASS'96 Workshop 2: Robocup Workshop: Soccer as a Problem for Multi-Agent Systems*.

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

Received February 29, 2008; accepted September 29, 2011.