MOSAIC for Multiple-Reward Environments

Norikazu Sugimoto, Masahiko Haruno, Kenji Doya, and Mitsuo Kawato

February 29, 2008

Abstract

In many situations of human motor control as well as robotics applications, movements are guided by rewards such as scores and salaries. Reinforcement learning (RL) has provided a useful framework to analyze and develop a computational algorithm for reward-based control. However, an essential difficulty in real-world RL control tasks is the variability of the external world, i.e., the RL agent has to consider both different kinds of rewards and multiple dynamics. This dual variability in the environment remains unexplored in previous RL studies. We previously proposed a modular control architecture known as MOSAIC that identified and controlled multiple external dynamics by constructing and switching multiple modules. The key computation in MOSAIC is the competition between these modules, for which winners are determined only by how precisely each module predicts the dynamics of the controlled object. Here, we extend this framework to RL and propose MOSAIC for multiple reward (MOSAIC-MR) architecture to handle the dual variability of reward and dynamics. In sharp contrast to the original MOSAIC, each RL controller (module) in MOSAIC-MR considers the predictability of not only the dynamics but also the reward, and multiple RL controllers learn and switch based on Bellman's optimality of reward (temporal difference error). We evaluated the new features of MOSAIC-MR architecture with a simple pendulum swinging-up task, where two different reward functions exist under the influence of the variable wind. The RL agent successfully learned how to switch and control, allowing it to outperform conventional RL methods. Then, to test the scalability of MOSAIC-MR, we examined the bi-pedal walking of a simulated robot that learned to walk smoothly by switching RL modules, thereby demonstrating the applicability and generality of the MOSAIC-MR architecture in motor control.

1 Introduction

The majority of our movements are goal-directed because motor function is the only output channel that can act on the external world. To learn such movements, we have to consider two different variabilities of the external world: one in relation to reward function and the other in relation to external dynamics. That is, we must learn under the uncertainty of how our movement might be rewarded as well as the variety of dynamics we have to interact with. A challenge confronting reinforcement learning (RL) [1] in motor control is that both of these variabilities simultaneously influence the selection of actions.

Considering the dual variability of reward and dynamics, if a control agent is implemented with standard RL, there is no hope for it to behave appropriately due to the interference between

multiple rewards and dynamics. Therefore, an RL agent must be able to divide a complex environment into simpler sub-environments automatically and to adapt multiple controllers that are appropriate for each sub-environment. Modular/hierarchical architecture is a general and efficient framework to meet this requirement, and various algorithms have been proposed [2, 3, 4, 5, 6, 7, 8, 9, 10, 6, 11, 12, 13, 14, 15, 16, 17]; see also [18] for neuroscience connection. However, previous modular/hierarchical learning algorithms can only handle either reward or dynamics, not both. That is, some algorithms have been proposed to handle the multiple reward functions [19, 20, 21], while other algorithms have been proposed to cope with the multiple environmental dynamics [22, 23, 24, 25, 26]. Among the latter approach to dynamics using modular architecture, MOSAIC (modular selection and identification for control) switches and learns multiple controllers based on each module's prediction error of forward dynamics. MOSAIC works reasonably well in several motor control tasks but has difficulty when faced with a multiple-reward environment. The key factor in MOSAIC is the "responsibility signal" [27], the posterior probability of selecting each module. The responsibility signal is computed by Bayes' rule, integrating the predictability of the forward dynamics (likelihood) and other prior probabilities such as visual and somatosensory inputs. This Bayesian framework devised for combining dynamics and other information may be applicable to integrating reward and dynamics in the context of reinforcement learning.

Therefore, in this paper, we extend the framework of MOSAIC to RL and propose MOSAIC-MR (MOSAIC for Multiple Reward) to cope with the dual variability of reward and dynamics in motor control. MOSAIC-MR divides the environment consisting of multiple rewards and dynamics into several sub-environments and controls each sub-environment by an RL control module. Each RL module is learned and selected based on Bellman's principle of optimality; the RL module with the smaller squared temporal difference (TD) error (i.e., deviation from Bellman's optimal condition) is assigned a higher likelihood. The TD based module selection in MOSAIC-MR contrasts with the original MOSAIC, where a controller is always paired with a forward predictor and the selection of the controller depends only on the preciseness of the forward prediction. The MOSAIC-MR architecture employs Bellman's principle of optimality (TD error) for selecting modules that behave optimally in such multiple-reward environments, since once multiple rewards are involved, it is impossible to select and learn an optimal RL controller based only on the dynamics [28].

This paper is constructed as follows. First, the MOSAIC-MR architecture is described in detail in Section 2. Then, in Sections 3, we evaluate the MOSAIC-MR architecture with a pendulum swinging-up task where two different reward functions are switched under the influence of the wind. When several pairs of reward function and external dynamics exist, the architecture learned how to switch and control the environment with little interference, and the behavior of the system was found to be indistinguishable from the analytical optimal solution. When multiple sub-environments are involved, it is important to identify what factor affects the differentiation or evolution of RL modules. In the pendulum simulation, we also discuss the general relationship between module switching and time constants of the value function. In section 4, we investigate bi-pedal walking of a simulated robot to test the scalability of MOSAIC-MR in a more realistic problem. The result shows that the robot learns to walk smoothly by switching autonomously constructed RL modules, demonstrating the applicability and generality of the proposed architecture. Finally, the paper concludes with discussions in Section 5.

2 MOSAIC-MR

The MOSAIC-MR architecture recognizes that the complex external world consists of multiple reward functions and external dynamics. Therefore, its objective is to decompose the external environment into sub-environments that can be characterized with different reward functions and dynamics and then output the optimal control policy for each sub-environment. More specifically, to achieve the goal, the architecture consists of three sets of modules: reward, forward, and RL (Figure 1). Here, we briefly overview the general functions of each module before going into the details of the algorithm below. Each reward module approximates a local reward function, and each forward module predicts a local environmental dynamics. Both modules work on environmental state $\mathbf{x}(t) \in \Re^{N}$ [1] and action $\mathbf{u}(t) \in \Re^{D}$ in local space. Note that the reward module approximates an immediate reward but not a future reward. In other words, the reward module differs from a value function. The key computation of MOSAIC-MR is that reward and forward modules divide a complex environment into several sub-environments controlled by RL modules. The decomposition of the external environment by the reward and forward modules is performed based on approximation errors of reward and dynamics, respectively. The RL module, which receives the outputs ($\hat{r}(t)$, $\hat{\mathbf{x}}(t)$) and results of decomposition ($\lambda_j^r(t)$, $\lambda_i^f(t)$) from the reward and forward modules, approximates a local value function and selects a local greedy action. Finally, MOSAIC-MR determines an action appropriate for the current status of the environment based on the temporal difference (TD) error as described later.

[Figure 1 about here.]

In the following, we explain how selection and learning are performed in each module in the order of reward, forward, and RL modules. Hereafter, for simplicity, the reward, forward, and RL modules are annotated by f, r, and c, respectively, and they are indexed by $i \in M^f$, $j \in M^r$, and $k \in M^c$.

2.1 Reward module

We assume that the reward function can be approximated by several local reward functions. The reward signal at each time step, $r(t)$, is given as a mixture of local reward functions. Responsibility signal λ_j^r, which represents the relative preciseness of multiple reward modules, is given by Bayes' rule:

$$\lambda_j^r(t) = \frac{P(j)p(r(t) \mid j)}{\sum_{j' \in M^r} P(j')p(r(t) \mid j')}, \tag{1}$$

where $P(j)$ is the prior probability of selecting reward module j and $p(r(t) \mid j)$ is the likelihood of reward module j given reward $r(t)$.

The likelihood of reward module $p(r(t) \mid j)$ is given by the approximation error of reward $r(t) - \hat{r}_j(t)$. Here, $\hat{r}_j(t)$ is the approximated reward of reward module j conditioned on current

[1] $\Re^{\bullet}$ represents $\bullet$-dimension column vector.

state $\mathbf{x}(t)$ and current action $\mathbf{u}(t)$. By assuming that approximation error is Gaussian with variance $(\sigma_j^r)^2$, likelihood $p(r(t) \mid j)$ is defined as follows:

$$p(r(t) \mid j) = \frac{1}{\sqrt{2\pi}\sigma_j^r} \exp\left[-\frac{1}{2(\sigma_j^r)^2}(r(t) - \hat{r}_j(t))^2\right], \tag{2}$$

$$\hat{r}_j(t) = r_j(\mathbf{x}(t), \mathbf{u}(t)). \tag{3}$$

When any specific switching feature is known *a priori*, we include this information in the formulation of the responsibility signal as a prior probability [25, 29]. In the MOSAIC family, two kinds of prior knowledge have been used for module selection: temporal continuity and spatial locality. These two types of prior knowledge are not task-dependent but general features. Temporal continuity reduces excessively frequent switching due to noise. The prior for this constraint is given by $P(j)$:

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

$E_j^r(t)$ represents a low-passed square error of reward approximation at time t,

$$\begin{aligned} E_j^r(t) &= \sum_{h=0}^{t/\Delta t} (\alpha^r)^{h\Delta t}(r(t - h\Delta t) - \hat{r}_j(t - h\Delta t))^2 \\ &= (r(t) - \hat{r}_j(t))^2 + (\alpha^r)^{\Delta t} E_j^r(t - \Delta t), \end{aligned} \tag{5}$$

where $0 < \alpha^r < 1$ is a parameter that controls the strength of the temporal continuity and Δt is a time step of observation and control. To implement spatial locality, since it uses a Gaussian spatial prior, prior probability $P(j)$ is formulated as follows:

$$P(j) \propto \frac{1}{(2\pi)^{N/2}|\Sigma_j^r|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}(t) - \mathbf{x}_j^r)^{\mathrm{T}}(\Sigma_j^r)^{-1}(\mathbf{x}(t) - \mathbf{x}_j^r)\right], \tag{6}$$

where $\mathbf{x}_j^r$ is the center of the area of localization and Σ_j^r is a covariance matrix that specifies the area in the jth reward module. If both temporal continuity and spatial locality need to be met, we implement the prior as the product of equations (4) and (6). We pursue this approach in the simulations below.

Next, we briefly explain how reward approximation modules are learned. The objective of learning in each reward module is to minimize the weighted approximation error. If each reward approximator is a polynomial formula, $\hat{r}_j(t) = W_j^r \mathbf{z}(t)$, the expected values for the jth reward module's localization center $\mathbf{x}_j^r$ and regression parameters W_j^r are as follows:

$$\mathbf{x}_j^r = \langle \mathbf{x} \rangle_j^r(T) / \langle 1 \rangle_j^r(T), \tag{7}$$

$$W_j^r = \langle r\mathbf{z} \rangle_j^r(T) / \langle \mathbf{z}\mathbf{z}^{\mathrm{T}} \rangle_j^r(T), \tag{8}$$

where $\mathbf{z}(t)$ is an input vector to each model and $\langle \bullet \rangle_j^r(T)$ denotes a weighted mean with respect to responsibility signal λ_j^r:

$$\langle \bullet \rangle_j^r(T) = \sum_{t=0}^{T/\Delta t} \bullet(t)\lambda_j^r(t). \tag{9}$$

By iterating the responsibility signal calculation and the parameter update, the likelihood will increase to sub-optimal. When the prior of the temporal continuity is not incorporated, the update rule is identical to the EM algorithm [30]. An on-line update can also be realized by modifying equation (9) as follows:

$$\langle \bullet \rangle_j^r(t) = (1-\eta)\langle \bullet \rangle_j^r(t-\Delta t) + \eta \bullet (t)\lambda_j^r(t), \tag{10}$$

where η is the learning rate.

2.2 Forward module

Selection and learning in the forward module are almost identical to those in the reward module. Each forward module predicts the environmental dynamics. MOSAIC-MR compares the prediction errors of each forward module and computes the responsibility signal of forward module $\lambda_i^f(t)$ by Bayes' rule:

$$\lambda_i^f(t) = \frac{P(i)p(\dot{\mathbf{x}}(t) \mid i)}{\sum_{i' \in M^f} P(i')p(\dot{\mathbf{x}}(t) \mid i')}, \tag{11}$$

where $\dot{\mathbf{x}}(t)$ is the time derivative of the state, $p(\dot{\mathbf{x}}(t) \mid i)$ is the likelihood of module i, and $P(i)$ is the prior probability of module i.

By assuming again that the prediction error is Gaussian with variance $(\sigma_i^f)^2$, likelihood $p(\dot{\mathbf{x}}(t) \mid i)$ is given by

$$p(\dot{\mathbf{x}}(t) \mid i) = \frac{1}{\sqrt{2\pi}\sigma_i^f} \exp\left[-\frac{1}{2(\sigma_i^f)^2}\|\hat{\dot{\mathbf{x}}}_i(t) - \dot{\mathbf{x}}(t)\|^2\right], \tag{12}$$

where $\hat{\dot{\mathbf{x}}}_i(t)$ is the prediction of the ith forward module f_i. Each forward module has the following form:

$$\hat{\dot{\mathbf{x}}}_i(t) = f_i(\mathbf{x}(t), \mathbf{u}(t)). \tag{13}$$

Here, in exactly the same manner as the reward model, if we incorporate the temporal continuity and spatial locality into prior probability $P(i)$, the prior ends up with equations (14) and (16):

$$P(i) \propto \frac{1}{\sqrt{2\pi}\sigma_i^f} \exp\left[-\frac{1}{2(\sigma_i^f)^2}E_i^f(t-\Delta t)\right], \tag{14}$$

$$\begin{aligned} E_i^f(t) &= \sum_{h=0}^{t/\Delta t} (\alpha^f)^{h\Delta t}\|\dot{\mathbf{x}}(t-h\Delta t) - \hat{\dot{\mathbf{x}}}_i(t-h\Delta t)\|^2 \\ &= \|\dot{\mathbf{x}}(t) - \hat{\dot{\mathbf{x}}}_i(t)\|^2 + (\alpha^f)^{\Delta t}E_i^f(t-\Delta t), \end{aligned} \tag{15}$$

where $0 < \alpha^f < 1$ is a parameter that controls the strength of the temporal continuity. $E_i^f(t-\Delta t)$ represents a low-passed square error of prediction before the observation at time t.

$$P(i) \propto \frac{1}{(2\pi)^{N/2}|\Sigma_i^f|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}(t)-\mathbf{x}_i^f)^{\mathrm{T}}(\Sigma_i^f)^{-1}(\mathbf{x}(t)-\mathbf{x}_i^f)\right], \tag{16}$$

where $\mathbf{x}_i^f$ is the center of the area of specialization and Σ_i^f is a covariance matrix that specifies the shape. The learning of forward modules can be done by the EM algorithm, similar to the learning of reward modules.

2.3 RL module

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

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

$$\lambda_k^c(t) = \frac{P(k)p(\mathbf{x}(t), r(t) \mid k)}{\sum_{k' \in M^c} P(k')p(\mathbf{x}(t), r(t) \mid k')}, \tag{17}$$

where $p(\mathbf{x}(t), r(t) \mid k)$ is the likelihood and $P(k)$ is the prior probability of the RL module. Likelihood $p(\mathbf{x}(t), r(t) \mid k)$ evaluates the goodness of each RL module based on the TD error of its local value function, which approximates an expected future reward, and TD error represents the error between expected and actual reward. For continuous state and time, TD error $\delta(t)$ is defined as follows with observed state $\mathbf{x}(t)$ and reward $r(t)$ at time t [31]:

$$\delta(t) = r(t) - \frac{1}{\tau}V(\mathbf{x}(t)) + \frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}}\dot{\mathbf{x}}(t). \tag{18}$$

Here, τ is a very important parameter that determines how far the agent looks ahead (time scale) to its future rewards.

An RL module with smaller squared TD error should contribute more to the control. This is the rationale behind the definition of the RL module's responsibility signal. By assuming that the TD error is Gaussian with variance $(\sigma_k^c)^2$, likelihood $p(\mathbf{x}(t), r(t) \mid k)$ is given by

$$p(\mathbf{x}(t), r(t) \mid k) = \frac{1}{\sqrt{2\pi}\sigma_k^c} \exp\left[-\frac{1}{2(\sigma_k^c)^2}\delta(t)^2\right], \tag{19}$$

$$\delta_k(t) = \hat{r}(t) - \frac{1}{\tau}V_k(t) + \frac{\partial V_k(\mathbf{x}(t))}{\partial \mathbf{x}}\hat{\dot{\mathbf{x}}}(t), \tag{20}$$

where $\hat{r}(t)$ and $\hat{\dot{\mathbf{x}}}(t)$ are approximated reward and predicted dynamics, respectively:

$$\hat{r}(t) = \sum_{j \in M^r} \lambda_j^r(t)\hat{r}_j(t), \tag{21}$$

$$\hat{\dot{\mathbf{x}}}(t) = \sum_{i \in M^f} \lambda_i^f(t)\hat{\dot{\mathbf{x}}}_i(t). \tag{22}$$

Here, $\hat{r}(t)$ and $\hat{\dot{\mathbf{x}}}_i(t)$ are the approximated values of $r(t)$ and $\dot{\mathbf{x}}(t)$ used to reduce the effect of the observation noise.

Next, we explain the prior probability in the responsibility signal of the RL module. Generally, TD error has large variance because it is very sensitive to both the performance of the reward approximator and the noise. In addition, if the state space has a high dimension or the number of RL modules is large, MOSAIC-MR has difficulty obtaining enough information from the TD error to select an appropriate module. Therefore, we additionally introduce two prior probabilities for module selection. First, we incorporate "selection frequency" representing how many times the RL module is selected. For each sub-environment, MOSAIC-MR counts the frequency of the selected RL modules as a probability distribution conditioned on the responsibility signals of the reward and forward modules, $P(k \mid \lambda_j^r(t), \lambda_i^f(t))$. For the condition of such prior probability, we introduce the responsibility signals of the reward module and the forward module instead of the raw data. In this way, the agent can learn the prior probability quickly because the responsibility signal can be regarded as abstracted information about the switching of the sub-environment. As a second probability, we incorporate "temporal continuity" in a similar way to the reward and forward modules. Thus, responsibility signal $P(k)$ results in the following equation:

$$P(k) \propto P(k \mid \lambda_j^r(t), \lambda_i^f(t)) \frac{1}{\sqrt{2\pi}\sigma_k^c} \exp\left[-\frac{1}{2(\sigma_k^c)^2} E_k^c(t-\Delta t)\right], \tag{23}$$

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

$$\begin{aligned} E_k^c(t) &= \sum_{h=0}^{t/\Delta t} (\alpha^c)^{h\Delta t} \delta(t-h\Delta t)^2 \\ &= \delta(t)^2 + (\alpha^c)^{\Delta t} E_k^c(t-\Delta t), \end{aligned} \tag{24}$$

where $0 < \alpha^c < 1$ is a parameter that controls the strength of the temporal continuity. Although this formulation of the prior is a heuristics, we found that it performs well, particularly when the reward information in the sub-environments is contaminated by much noise.

The learning rule for each RL module resembles standard TD learning. By using a weighted TD error with responsibility signals, each RL module is expected to be localized in each sub-environment. When the TD error of the kth RL module is given by equation (17) at time t, we minimize the following objective function:

$$D_k(t) = \frac{1}{2} \lambda_k^c(t) \delta_k(t)^2. \tag{25}$$

Each RL module updates the parameter by a gradient descent method:

$$\dot{w}_k^c = -\eta^c \frac{\partial D_k}{\partial w_k^c}, \tag{26}$$

$$= -\eta^c \lambda_k^c(t) \delta_k(t) \left[-\frac{1}{\tau} \frac{\partial V_k(\mathbf{x}(t))}{\partial w_k^c} + \frac{\partial}{\partial w_k^c} \frac{\partial V_k(\mathbf{x}(t))}{\partial \mathbf{x}} \dot{\mathbf{x}}(t)\right], \tag{27}$$

where w_k^c is the parameter of the RL module and η^c is the learning rate.

Here, we assume that reward $r(\mathbf{x}(t), \mathbf{u}(t))$ consists of two terms: the reward for state $\mathbf{x}(t)$ and the cost for action $\mathbf{u}(t)$. In this case, the greedy action in the kth RL module based on its

value function $V_k(\mathbf{x}(t))$ is given by

$$\mathbf{u}_k(t) = S'^{-1}\left(\frac{\partial\hat{\mathbf{x}}(t)}{\partial\mathbf{u}}^{\mathrm{T}}\frac{\partial V_k(\mathbf{x}(t))}{\partial\mathbf{x}}^{\mathrm{T}}\right), \tag{28}$$

where $S(\bullet)$ is a cost function for action and $S'(\bullet)$ is a partial differential by action $\mathbf{u}$[31]. Each RL module outputs the local greedy action, denoted as $\mathbf{u}_k(t)$, and based on the gradient of local value function $\frac{\partial V_k(\mathbf{x}(t))}{\partial\mathbf{x}}$. Then the MOSAIC-MR weights each action $\mathbf{u}_k(t)$ by responsibility signal $\lambda_k^c(t)$ and finally executes it as its motor command:

$$\mathbf{u}(t) = \sum_{k\in M^c}\lambda_k^c(t)\mathbf{u}_k(t). \tag{29}$$

3 Pendulum Swing-up with Multiple Goals and Dynamics

We tested the learning performance of the MOSAIC-MR algorithm with a pendulum swing-up task in which multiple reward functions and multiple dynamics comprise an external environment. This task is suitable for evaluating the basic properties of the algorithm because optimal behavior can be derived analytically. In this simulation, we examined the following specific properties: (1) how reward and forward modules decompose an environment into sub-environments; (2) how each RL module adapts to sub-environments; and (3) as stated in the introduction, whether different sets of behaviors are learned depending on how far in advance the MOSAIC-MR agent anticipates future rewards.

[Figure 2 about here.]

Figure 2a shows the simulation setting. The angle of the pendulum 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 was simulated by the Euler method, both with time steps of 0.05 sec.

The MOSAIC-MR agent must learn two reward functions, R_{l} and R_{r}. R_{l} and R_{r} give maximum reward to the agent when the pendulum lean left and right, respectively:

$$r(\mathbf{x}(t),\mathbf{u}(t)) = \cos\left(\theta(t)-\theta_0\right) - S(\mathbf{u}(t)) \quad \begin{cases} R_l\colon \theta_0 = -\frac{45\pi}{180}\,\mathrm{rad} \\ R_r\colon \theta_0 = \frac{45\pi}{180}\,\mathrm{rad} \end{cases}, \tag{30}$$

where S is the cost function for the action defined as

$$S(\mathbf{u}(t)) = \frac{1}{2}0.001T(t)^2. \tag{31}$$

There are two external dynamics (F_{nowind}, F_{wind}); the wind blows downward in F_{wind}, but there is no wind in F_{nowind}. Equations for both dynamics are given as follows:

$$\ddot{\theta}(t) = \left(\frac{g}{l}+\frac{w}{ml}\right)\sin\theta(t) - \frac{\mu}{ml^2}\dot{\theta}(t) + \frac{1}{ml^2}T(t) + \mathcal{N}(0,0.1^2) \quad \begin{cases} F_{\mathrm{nowind}}\colon w = 0\,\mathrm{N} \\ F_{\mathrm{wind}}\colon w = 5\,\mathrm{N} \end{cases}, \tag{32}$$

where $\mathcal{N}(0, 0.1^2)$ represents the system noise with mean 0 and variance 0.1^2. Thus, the task consists of four sub-environments because there are two reward functions and two environmental dynamics. In this simulation, for convenience we indexed sub-environments by four colors: purple, cyan, blue, and green (Figure 2b).

The main structure of this simulation consists of three parts (Figure 2c). In the first part, we alternate all four sub-environments to examine how each RL module adapts to them. In the second part, only two sub-environments (purple and cyan) are alternated; the blue and green environments never appear. In the third part, we alternate the blue and green environments after exposure to the purple and cyan environments in the second part to examine whether the agent can maintain motor skills for the blue and green environments.

3.1 Results of reward and forward modules

[Figure 3 about here.]

Generally, the reward and forward modules can be learned separately from the RL module because learning the former can be performed by a supervised learning method. In other words, the reward and forward modules can be learned without a huge number of trial and error because feedback is instantly available during learning. Additionally, once the reward and forward modules are complete, the agent can identify any combination of the two kinds of modules. Therefore, we trained the reward and forward modules by using randomly selected actions before starting the RL module learning. More specifically, we prepared four reward modules, indexed as I, II, III, and IV. Each reward module approximates the reward function in a quadratic form:

$$r_j(\mathbf{x}(t), \mathbf{u}(t)) = -\frac{1}{2}Q_j(\theta(t) - \mathbf{x}_j^r)^2 + q_j - \frac{1}{2}0.001T(t)^2. \tag{33}$$

Figure 3a shows the results of reward approximation after learning. Two reward modules, r_{I} and r_{II}, approximated reward function R_{r}, and the other two reward modules, r_{III} and r_{IV}, approximated reward function R_{l} around the maximum and minimum points, respectively. Figure 3b shows the decomposition results of the reward functions. The top and bottom rows show the mean of the responsibility signals for reward functions R_{l} and R_{r}, respectively. Black and white represent 1 and 0. r_{III} and r_{IV} are selected for reward function R_{l}, and r_{I} and r_{II} are selected for reward function R_{r}. Thus, the MOSAIC-MR agent can learn the reward models perfectly.

The forward modules are learned similarly to the reward modules. We prepared four forward modules, indexed as 1, 2, 3, and 4 that are implemented in linear form:

$$f_i(\mathbf{x}(t), \mathbf{u}(t)) = A_i\mathbf{x}(t) + B_i\mathbf{u}(t) + c_i. \tag{34}$$

Figure 3c shows the results of forward prediction. Forward modules f_1 and f_2 predict environmental dynamics F_{nowind}, and forward modules f_3 and f_4 predict environmental dynamics F_{wind}. Despite the difficulty arising from the linear approximation overlap, the forward modules successfully decomposed two environmental dynamics (Figure 3d).

3.2 Results of RL modules

After the learning of the reward and forward modules, the four RL modules, indexed as A, B, C, and D, learned to control the pendulum. Each trial started at $t = 0$ sec and lasted for 10 sec unless the pendulum was over-rotated ($|\dot{\theta}(t)| > 3\pi$). The initial state of the pendulum, $\mathbf{x}(0)$, was chosen randomly from uniform distribution ($\theta(0) \in [-\pi, \pi]$, $\dot{\theta}(0) \in [-2\pi, 2\pi]$). A sequence of 100 trials makes one block, and the sub-environment was switched everytime a new block started. The learning parameters used were $\tau = 1$, $(\sigma^c_{A\sim D})^2 = 0.15$, and $\alpha^c = 0.9$. We used NGNet [32] to implement a value function.

[Figure 4 about here.]

Figure 4a shows the adaptation results of each RL module. The mean of the responsibility signal of RL module $\lambda^c_k(t)$ is represented by the darkness of color. The learning process continued until the 14th block, and the mapping of the RL module and the sub-environment were fixed after the 15th block. In every block after the 15th block, the MOSAIC-MR agent switched the RL module perfectly, and the mean of the responsibility signal became nearly 1 or 0. After this learning, a single RL module controlled a single sub-environment: RL modules A, B, C, and D controlled sub-environments cyan, purple, green, and blue, respectively.

[Figure 5 about here.]

Figure 4b shows the learning curve in terms of cumulative reward. The mean and the standard deviation of 10 simulation runs are plotted. From the first to the second parts, the cumulative rewards of purple and cyan sub-environments gradually increased except for the fluctuation seen in some initial blocks, and the maximum values were 8.49 ± 0.34 and 8.47 ± 0.23. Note that in the initial stage of the third part, the performances of the blue and green sub-environments were maintained at the same level as the last stage of the first part (blue: $5.99 \pm 1.14 \rightarrow 5.99 \pm 0.77$, green: $5.73 \pm 2 \rightarrow 6.24 \pm 1.39$). That is, the MOSAIC-MR agent successfully avoided interference between the sub-environments. Then, in the third part, the cumulative rewards of blue and green sub-environments reached maximum value, 6.92 ± 0.47 and 7.11 ± 0.42.

Since TD error is the driving force of RL module selection in MOSAIC-MR, the selected RL module must exhibit the least TD error among all RL controllers. Figure 5 confirms that in each sub-environment the squared TD error of the selected RL modules (shown in Figure 4a) took the minimum value and consistently decreased. It is also clear that the TD error of modules B (purple) and A (cyan) decreased while modules D (blue) and C (green) controlled the sub-environments (see also the bottom two panels in Figure 5). This is probably because these sub-environments are very similar except for the existence of the wind. This indicates the possibility that two modules (i.e., A and C, and B and D) are reduced to a single module if exposure to the same condition lasts for an infinite number of trials. This reduction of modules seems reasonable when no explicit top-down signal indicating module switching is available, and the erased module is unlikely to be useful in the future.

3.3 Comparison with a standard RL and a simple modular-RL

[Figure 6 about here.]

For comparison, we simulated the same task by two other RL algorithms. First, we tested a continuous-time and continuous-state-space RL (here called flat RL, and for details, see section 4.2 of [31]) to demonstrate the advantage of the modular architecture. The flat RL learned the value function with a function-approximator, which has the same structure and number of parameters as that used by MOSAIC-RL, and it output an action according to equation (28) (partial differential of dynamics, $\frac{\partial \dot{\mathbf{x}}(t)}{\partial \mathbf{u}(t)}$, is derived from motion equation analytically). Figure 6a shows the learning curves of the cumulative reward in each sub-environment. In each sub-environment (purple, cyan, blue and green), the maximum values were 5.13 ± 0.29, 6.57 ± 0.66, 3.73 ± 0.19 and 4.01 ± 0.28, and they were lower than MOSAIC-MR in all sub-environments. We used the t-test to compare the performance stabilities after learning of MOSAIC-RL and flat RL, and we found that MOSAIC-RL performed better in all sub-environments ($p < 0.01$) [2]. Thus, MOSAIC-MR had a statistically significant superiority over flat RL. It is likely that the lower performance of flat RL is due to the interference of learning between sub-environments. In the initial stage of the first part, the cumulative reward in the cyan sub-environment increased, but it decreased in the middle stage. At the same time, the cumulative reward in the purple increased. Therefore, there is a trade-off between the cyan and purple sub-environments, and one RL controller can learn the optimal policy of only one sub-environment.

Next, we tested MMRL [28], an example of the modular RL algorithm based on the dynamics. One module of MMRL consists of one forward model and one RL controller, and MMRL selects a module based only on the accuracy of forward prediction. In this simulation, the MMRL agent used four modules, and the forward model and the RL controller have the same architecture and number of parameters as MOSAIC-RL in the previous section. Figure 6b shows a learning curve in terms of cumulative reward in four sub-environments. We again compared performance stabilities after learning of MOSAIC-RL and MMRL the by t-test; consequently, MOSAIC-RL outperformed MMRL in all sub-environments ($p < 0.01$). In the initial stage of the first part, the cumulative reward in the cyan sub-environment increased, but it saturated at the $18th$ block. The maximum value was 6.4 ± 1.11, substantially lower than MOSAIC-MR's performance (8.47 ± 0.23). The maximum value of purple was 5.72 ± 0.78, which is also lower than that of MOSAIC-MR (8.49 ± 0.34). To examine the reason for MMRL's low performances, we analyzed the trajectory after learning in the cyan and purple. Despite the fact that the optimal solutions are the left and right tilted states, the learned final state of the pendulum in both sub-environments was the vertical standing position (around $\theta = 0$ rad). MMRL can detect the change of the wind but by definition cannot detect the change of the reward. Therefore, MMRL learned sub-optimal policy, which is intermediate between the left and right tilted states. In the blue and green sub-environments, the learning speed was very slow, and the maximum values of cumulative reward were 4.73 ± 1.46 and 5.03 ± 1.55, lower than MOSAIC-MR (6.92 ± 0.47, 7.11 ± 0.42). This result seems to arise from the difficulty of downward wind in addition to interference between the two objective states.

3.4 Results for short time constant of value function

[Figure 7 about here.]

It is important to clarify which parameter plays a crucial role in module selection. The previous works on the MOSAIC architecture established that the competition of the module

[2] We compared algorithms at 59, 60, 79 and 80 blocks in purple, cyan, blue and green sub-environments.

selection depends on the scaling parameter of the responsibility signal [25, 29, 28]. In general, the larger scaling parameter gives softer competition and slower differentiation, and one module tends to control a larger space. The scaling parameter must surely have the same effect in MOSAIC-MR as well. Another important factor that potentially affects module selection in modular reinforcement learning is the time constant of the value function. Therefore, we examined how the time constant affects module selection by a simulation experiment.

The time constant determines how many events the value function reflects; the value function learns specialized shape for one environment if many observation data are taken into consideration. The similarity between each sub-environment becomes low if the time constant is large; the similarity becomes high if the time constant is small. When the time constant is large and the similarity is low, several RL modules are learned for control since RL modules differentiate easily. In contrast, when the time constant is small and the similarity is high, a relatively small number of RL modules are learned for control. To see whether these expectations are supported , we set the time constant small (τ: $1 \rightarrow 0.5$) and tested the same task as in the previous section.

Figures 7 show the results of short time-constant ($\tau = 0.5$). Figure 7a shows the responsibility signals of RL modules for each block. Four sub-environments were controlled by two RL modules, each responsible for two environments: RL module A controlled the purple and blue environments, RL module C controlled the cyan and green environments, and RL modules B and D were not used. In short, a single RL module covered the control of two tasks under two different dynamics (i.e., the environment is separated based on reward). Figure 7b shows cumulative reward for each sub-environment. In the mid-stage of the first part, the cumulative reward for purple decreased slightly while the cumulative reward for blue rapidly increased. This interference probably arose from the overgeneralization of RL modules. In the latter stage of this part, the parameter of RL module A seems to drift between sub-optimum for purple and sub-optimum for blue every time the control block is switched. In the second part, RL module A learned the optimal policy for the purple sub-environment. However, the cumulative reward of the blue sub-environment again decreased at the initial stage of the third part ($6.7 \pm 0.24 \rightarrow 5.22 \pm 0.36$) because RL module A was responsible for the purple sub-environment in the second part. For the same reason, the cumulative reward of the green sub-environment decreased from 6.22 ± 0.51 to 4.87 ± 0.4.

To compare the learning speed in cases of both large and short time constants, we fitted the learning curve by following an exponential function in the first part: $\text{cumulative reward} = e_1(1 - \exp(-e_2 \times \text{block})) + e_3$. Here, e_2 represents learning speed of the cumulative reward [3]. Comparing the learning speed in the purple, cyan, blue and green sub-environments, the values (e_2) were 0.12, 0.13, 0.0077 and 0.0087 in the case of a long time constant and 0.65, 0.57, 0.031 and 0.02 in the case of a short time constant. Thus, the learning speed of the short time constant is faster than that of the long time constant in all sub-environments.

[Figure 8 about here.]

The RL module's policy moved between the two sub-optimal policies when the time constant was small. To analyze the interference between the purple and blue sub-environments, we compared the sub-optimal policy with the optimal policy. Figures 8a and b show the value functions of the purple and blue sub-environments with overgeneralization ($\tau = 0.5$). The solid

[3] e_1 and $e_1 + e_2$ represent amount of increase and maximum value of cumulative reward

line and white dot represent the swing-up trajectory and endpoint of the trajectory, respectively. Figures 8c and d show the value functions of the purple and blue sub-environments without overgeneralization ($\tau = 1$). In the purple sub-environment, the optimal trajectory for the area surrounded by a green dashed rectangle is reduced angular velocity (Figure 8c). On the other hand, the optimal trajectory of the blue sub-environment is accelerated (Figure 8d). However, in both Figures 8a and b, the angular velocities were also accelerated due to the interference between the two environments. Notice that the endpoint (white dot) discrepancy and variance are smaller in the optimal cases (Figures 8c and d) than in the sub-optimal cases (Figures 8a and b), demonstrating the better learning performance of the optimal case.

In general, it is very difficult for modular reinforcement learning algorithms to determine the adequate number of modules to achieve learning and control. In a series of simulations of the pendulum swinging-up task, we showed that the basic MOSAIC-MR can detect the optimal number of modules and control the pendulum appropriately. Furthermore, we demonstrated that the time constant for value functions critically affects the number of RL modules employed. When the time constant is large, a larger number of RL modules are introduced during learning, each of which covers a smaller area of the input space. On the other hand, if small, an RL module learns and controls the multiple areas of the input space. In this case, the time required for learning decreases, while the learned control law is sub-optimal. Practical RL algorithms may have to adjust the number of modules according to the emergency. For example, when a lot of reward functions and dynamics exist in the task, it is sometimes difficult to adapt the RL module to all sub-environments due to the huge number of sub-environments and the limited time. In such a situation, the agent must be able to learn any sub-optimal policy necessary.

4 Controlling a Locomotion Robot

[Figure 9 about here.]

The control of a locomotion robot is difficult due to the strong non-linearity of the dynamics and the discontinuous switching of reward and dynamics caused by the contact between leg and ground. Some researchers have proposed that the finite state machine (FSM) is applicable to locomotion robots [33]. In the FSM framework, the designer divides the state space into several domains and manually configures the control law for each domain. This requires a huge amount of prior information on the robot and task. By contrast, MOSAIC-MR can divide the environment and configure the controller automatically. In other words, MOSAIC-MR can be regarded as one of the possible algorithms that might automatically learn FSM. Therefore, our algorithm has the potential to be applied to locomotion robots. In this section, we pursued this direction and applied MOSAIC-MR to the two-link locomotion robot shown in figure 9.

The state vector and action were $\mathbf{x}(t) = [\theta_L(t)\ \ \theta_R(t)\ \ \dot{\theta}_L(t)\ \ \dot{\theta}_R(t)]^{\mathrm{T}}$ and $\mathbf{u}(t) = T(t)$, respectively, where θ_L, θ_R, and T are the angle of the left leg, the angle of the right leg, and the torque of the hip. We used two reward functions, denoted as R_{L} and R_{R}. When the left leg swung, reward function R_{L} was applied, and when the right leg swung, reward function R_{R} was

applied:

$$R_{\mathrm{R}}(\mathbf{x}(t),\mathbf{u}(t)) = -\left(\theta_L(t) - \frac{15}{180}\pi\right)^2 - \left(\theta_R(t) + \frac{30}{180}\pi\right)^2 - S(\mathbf{u}(t)), \tag{35}$$

$$R_{\mathrm{L}}(\mathbf{x}(t),\mathbf{u}(t)) = -\left(\theta_L(t) + \frac{30}{180}\pi\right)^2 - \left(\theta_R(t) - \frac{15}{180}\pi\right)^2 - S(\mathbf{u}(t)). \tag{36}$$

If neither leg touched the ground, the reward signal was 0. Cost function $S(\mathbf{u}(t))$ was given by $\frac{1}{2}0.1T(t)^2$. We assume that the walking cycles consist of phases of single-leg support, whith means that only one leg touches the ground at each time step [34]. The phase of single-leg support transits to the other phase precisely when the swinging leg contacts the ground.

Reward and forward modules were learned in a similar way as the pendulum swing-up. The training data were sampled with random initial state and random action. The number of reward modules was two, and each reward module was formulated as a quadratic model; the number of forward modules was twenty, and each forward module was formulated as a linear model. Optimal control theory has established that an optimal RL controller can be derived analytically when the reward function and the environmental dynamics are in quadratic and linear forms, respectively. We thus derived RL modules for all combinations of each reward module and each forward module.

4.1 Results

In Figure 10a, the gray and black lines represent the angle position of the left and right legs, respectively. The angle only fluctuated at the very beginning (0-0.5 sec after movement onset), but it soon showed a stable and cyclic trajectory. The discontinuous change in these angles just before and after the landing probably arises from the collision of the leg with the ground. We also determined which RL modules contributed to the control of either leg. Figure 10b shows a time series of the responsibility signals of each module in which an RL module is associated with a pair of forward and reward modules (Figure 1). Gray and black dotted lines represent the timing of the landing for the left and right legs, respectively. Here, darker contrast means higher responsibility. Out of forty RL modules initially introduced, only four (3-I, 16-I, 17-I, and 1-II) were used for actual control. The fact that these RL modules were heavily switched around the dotted lines suggests that the control law of the RL agent critically depends on the time around landing. The abrupt change of dynamics around this point may be so complicated that it requires more than a single RL module to fully adapt to this environment. Notice also that three RL modules were used while the right leg was swinging, but one RL module was on the opposite side.

[Figure 10 about here.]

[Figure 11 about here.]

Next, we investigated why such an asymmetric recruitment of RL modules occurred. Figure 11 illustrates a pair of forward reward modules used during control along the trajectory on the $\theta_L - \theta_R$ plane. In this figure, forward modules are denoted by circles, diamonds, crosses, and rectangles and reward modules by gray and black markers. While the left leg is swinging, forward module f_1 is used, and forward modules $f_{3,16,17}$ are used when the right leg is swinging.

Thus, the switching between RL modules in this task is determined by forward dynamics. We speculate that more RL modules would be recruited if we increased the learning period of the forward models for the left leg.

Unlike the pendulum swinging-up task, the optimal number of modules is unknown in many real-world applications of RL. In such control of a locomotion robot, we cannot determine the number of forward, reward, and RL modules. Interestingly, the number of forward modules used for the left and right legs are, in fact, different as shown in Figure 11. Nevertheless, the robot learned to walk smoothly, and the angle positions for the left and right legs themselves are symmetric. Thus, this task exemplifies the utility of the MOSAIC-MR algorithm in a realistic environment in which we cannot know the structure of the problem *a priori*.

5 Discussion

This paper proposed the MOSAIC-MR architecture to achieve efficient reinforcement learning in a complex environment consisting of multiple reward functions and multiple environmental dynamics. MOSAIC-MR automatically detects the changes in both the reward function and the environmental dynamics and decomposes the environment into sub-environments. Then, multiple RL modules are adapted to the sub-environments. In the pendulum swinging-up simulation, we compared MOSAIC-MR with two RL algorithms, flat RL and MMRL. MOSAIC-MR perfectly learned and controlled sub-environments with little interference and outperformed the two conventional methods. We also examined how the time constant of the RL module's value function affects construction of the modules and switching among them. Briefly, when the time constant is large, the agent obtained an optimal policy; however, more learning trials were needed than with a smaller constant. By contrast, the module with a small time constant tended to cover multiple sub-environments but learned sub-optimal policy faster than with a larger constant. We also applied MOSAIC-MR to the control of a locomotion robot and demonstrated that the MOSAIC-MR agent could learn to walk smoothly.

These simulations highlighted two key features of MOSAIC-MR. First, the comparison with MMRL clarified that module selection based on forward dynamics is not sufficient in modular reinforcement learning and taht a TD-based (Bellman's optimality) method is better suited to this task. Second, Bayesian integration of reward and dynamics into RL controllers makes it easy to apply MOSAIC-MR to various applications. That is, in MOSAIC-MR, RL modules only receive an abstracted form of prior probabilities from reward and forward modules and do not need to access the detailed specifications of the task, such as detailed trajectory of movements or the amount of reward. Therefore, even if the RL controller is required to adapt to a completely new task, only the reward and forward modules need to be modified. This clear dissociation of RL modules and task specifications increases the generality and applicability of the MOSAIC-MR architecture. Related to this, the RL controller observes only such abstracted information, and it thus efficiently handles the continuous reward function and environmental dynamics. Such abstraction also helps to capture the hierarchical structure of the environment by recursively embedding this architecture and may provide reward prediction, state prediction, motion planning and other hidden-state estimation in the longer time span.

We discussed the general relationship between time constant of the value function and differentiation of the RL modules in the simulation of the pendulum. Large and small time constants of the value function have both pros and cons, and it therefore might be reasonable to use a small time constant in the initial stage of learning and then move on to a larger time

constant in the final stage.

We also showed that finite state machine (FSM) implementation in a locomotion robot can be reformulated by the MOSAIC-MR architecture, and we demonstrated that MOSAIC-MR can divide the state space for locomotion (Figure 11) automatically. Additionally, although in the hand-crafted FSM framework, collision detection between the leg and the ground requires a particular kind of sensor; MOSAIC-MR can learn to detect this only from the time series of the environmental dynamics. Even in such a complex task, where multiple reward functions and dynamics alternate intricately due to the interaction between legs and the ground, MOSAIC-MR achieved smooth walking by integrating the dynamics and reward as the RL controller's responsibility signal.

In this paper, we have mainly focused on the importance in control engineering of modularizing the reward, dynamics and RL controller individually and then combining these components dynamically. In connection with this control architecture, it is known in system neuroscience that a reward signal is processed in the midbrain and that the value function is learned in the basal ganglia and frontal cortex, regulated by the midbrain dopaminergic projections. Furthermore, it has recently been revealed that the cerebellum, one of the key brain structures for dynamics learning, communicates with the basal ganglia [35] through a bi-synaptic connection. This novel finding poses the possibility that computations for three different components in MOSIAC-MR, i.e., reward, RL controller and dynamics [36], take place in different brain areas and that these neural substrates closely interact with each other. In this sense, the MOSAIC-MR architecture may advance the understanding of such hierarchical and modular learning in the brain [18, 37].

References

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

[2] Leslie Pack Kaelbling, Michael L. Littman, and Andrew P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[3] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13:341–379, 2003.

[4] Satinder P. Singh. Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proceedings of the Ninth Machine Learning Conference*, 1992.

[5] C. L. Giles, S. J. Hanson, and J. D. Cowan, editors. *Feudal Reinforcement Learning*, San Mateo, CA, 1993. Morgan Kaufmann.

[6] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[7] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Intra-option learning about temporally abstract actions. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 556–564. Morgan Kaufmann, 1998.

[8] Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 118–126. Morgan Kaufmann, San Francisco, CA, 1998.

[9] Marco Wiering and Jürgen Schmidhuber. HQ-learning. *Adaptive Behavior*, 6(2):219–246, 1997.

[10] Jun Morimoto and Kenji Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems*, 36(1):37–51, 2001.

[11] Anders Jonsson and Andrew G. Barto. Automated state abstraction for options using the u-tree algorithm. In *Advances in Neural Information Processing Systems*, pages 1054–1060, 2000.

[12] Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst., 2000.

[13] Kazuyuki Samejima, Kenji Doya, and Mitsuo Kawato. Inter-module credit assignment in modular reinforcement learning. *Neural Networks*, 16:985–994, 2003.

[14] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1997.

[15] Lambert E. Wixson. Scaling reinforcement learning techniques via modularity. In *Proceedings of the Eighth International Conference on Machine Learning*, pages 3368–372, 1991.

[16] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In *Uncertainty in Artificial Intelligence*, pages 220–229, 1998.

[17] Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*. MIT Press., 1995.

[18] Mitsuo Kawato and Kazuyuki Samejima. Efficient reinforcement learning: computational theories, neuroscience and robotics. *Current Opinion in Neurobiology*, 17, 2007.

[19] Stuart Russell and Andrew L Zimdars. Q-decomposition for reinforcement learning agents. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 656–663, 2003.

[20] Sriraam Natarajan and Prasad Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 601–608, 2005.

[21] Satinder Singh, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems*, 2005.

[22] Kumpati S. Narendra, Jeyendran Balakrishnan, and M. Kemal Ciliz. Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems Magazine*, 15(3):37–51, 1995.

[23] Jun Tani and Stefano Nolfi. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12(7-8):1131–1141, 1999.

[24] Jun Tani and Masato Ito. Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 33(4):481–488, 2003.

[25] Daniel M. Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.

[26] Klaus Pawelzik, Jens Kohlmorgen, and Klaus-Robert Müller. Annealed competition of experts for a segmentation and classification of switching dynamics. *Neural Computation*, 8(2):340–356, 1996.

[27] Masahiko Haruno, Daniel M. Wolpert, and Mitsuo Kawato. *Hierarchical MOSAIC for movement generation*, volume 1250 of *International Congress Series*. T. Ono, G. Matsumoto, R.R. Llinas, A. Berthoz, R. Norgren, H. Nishijo and R. Tamura (Eds.), Amsterdam, 2003.

[28] Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural Computation*, 14:1347–1369, 2002.

[29] Masahiko Haruno, Daniel M. Wolpert, and Mitsuo Kawato. MOSAIC model for sensorimotor learning and control. *Neural Computation*, 13:2201–2220, 2001.

[30] Lei Xu, Michael I. Jordan, and Geoffrey E. Hinton. An alternative model for mixtures of experts. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 633–640. The MIT Press, 1995.

[31] Kenji Doya. Reinforcement leaning in continuous time and space. *Neural Computation*, 12:219–245, 2000.

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

[33] Jessica K. Hodgins. Biped gait transitions. In *IEEE International Conference on Robotics and Automation*, 1991.

[34] Jesse W. Grizzle, Gabriel Abba, and Franck Plestan. Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *IEEE Transactions on Automatic Control*, 46:51–64, 2001.

[35] Eiji Hoshi, Léon Tremblay, Jean Féger, Peter L Carras, and Peter L Strick. The cerebellum communicates with the basal ganglia. *Nature Neuroscience*, 8(11):1491–1493, 2005.

[36] Biren Mehta and Stefan Schaal. Forward models in visuomotor control. *Journal of Neurophysiology*, 88:942–953, 2002.

[37] Masahiko Haruno and Mitsuo Kawato. Heterarchical reinforcement-learning model for integration of multiple cortico-striatal loops: fmri examination in stimulus-action-reward association learning. *Neural Networks*, 19:1242–1254, 2006.

List of Figures

1 Schematic diagram of MOSAIC-MR. Reward modules, displayed in top-left, approximate immediate reward, and forward modules, displayed in bottom-left, predict local dynamics of environment. Reward and forward modules decompose a complex environment into sub-environments based on approximation and prediction errors, respectively. RL modules, displayed at right, output the action to sub-environments. Each RL module is weighted based on how much it contributes to the current task. 20

2 Simulation time step Δt is 0.05 sec, and time derivative of state $\dot{\mathbf{x}}(t)$ is given by Euler method. Mass m is 1 kg, length l is 1 m, friction coefficient μ is 0.1, and acceleration of gravity g is 9.81 m/sec^2. Torque is limited $|T| < 10\,\mathrm{Nm}$ 21

3 Learning results of reward and forward modules. (a) Gray dashed lines represent each reward function (R_l, R_r), and black lines represent each reward module ($r_\mathrm{I} \sim r_\mathrm{IV}$). Horizontal axis represents θ. (b) Mean of responsibility signal after learning for each reward function. Darker black means more frequently used module. (c) Gray dashed lines represent two environmental dynamics (F_nowind, F_wind), and black lines represent each forward module ($f_1 \sim f_4$). (d) Mean of responsibility signal for each environmental dynamics (same format as (b)). . . 22

4 (a) Typical example of responsibility signal of RL module $\lambda_k^c(t)$. In each block, mean of $\lambda_k^c(t)$ is plotted. Color of each block represents a sub-environment, and darkness means how often module was selected. (b) Mean of cumulative reward for each block over 10 simulation runs. Vertical lines show standard deviations. 23

5 Learning curves in terms of TD error. 24

6 Learning curves of cumulative reward. (a) and (b) are results of flat RL and MMRL, respectively. 25

7 Responsibility signal and cumulative reward when time constant of a value function is short ($\tau = 0.5$). 26

8 Top row shows value functions with overgeneralization ($\tau = 0.5$) for single RL module controlling both purple and blue sub-environments. Solid line and white dot represent swing-up trajectories and endpoint of trajectory, respectively. Bottom row shows value functions without overgeneralization ($\tau = 1$). In the center of top and bottom row, black ellipse and circle represent the sub-environment that the RL module control. 27

9 A two-link robot for locomotion. Positive is clockwise rotation. Mass of each leg, m_L and m_R, is 0.5 kg, length of each leg, l_L and l_R, is 0.1016 m. 28

10 (a) Time course of angle of legs (θ_L and θ_R). Gray and black lines show left and right legs, respectively. Solid and dashed lines represent touching and swinging leg, respectively. (b) Responsibility signal of each RL module. Darker color indicates higher value. Gray and black dashed lines indicate that the left and right legs land on the ground. 29

11 Trajectory and dividing. Large circle around gray arrow shows that left leg has landed, the other large circle shows that right leg has landed. Color of each marker (gray and block) represents each reward module, and shape of each marker represents each forward module. 30

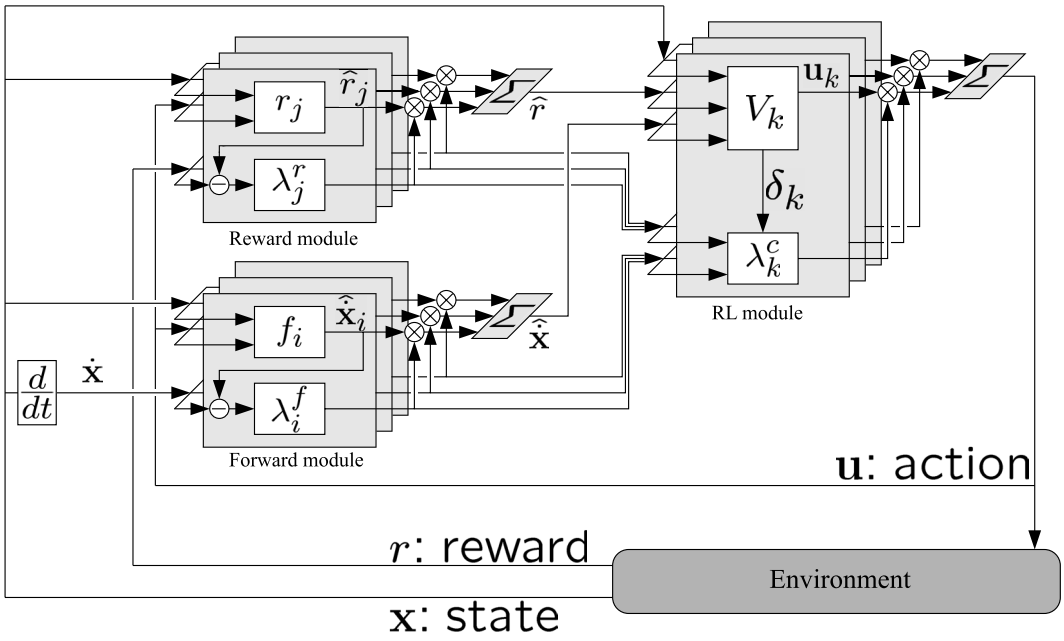

Figure 1: Schematic diagram of MOSAIC-MR. Reward modules, displayed in top-left, approximate immediate reward, and forward modules, displayed in bottom-left, predict local dynamics of environment. Reward and forward modules decompose a complex environment into sub-environments based on approximation and prediction errors, respectively. RL modules, displayed at right, output the action to sub-environments. Each RL module is weighted based on how much it contributes to the current task.

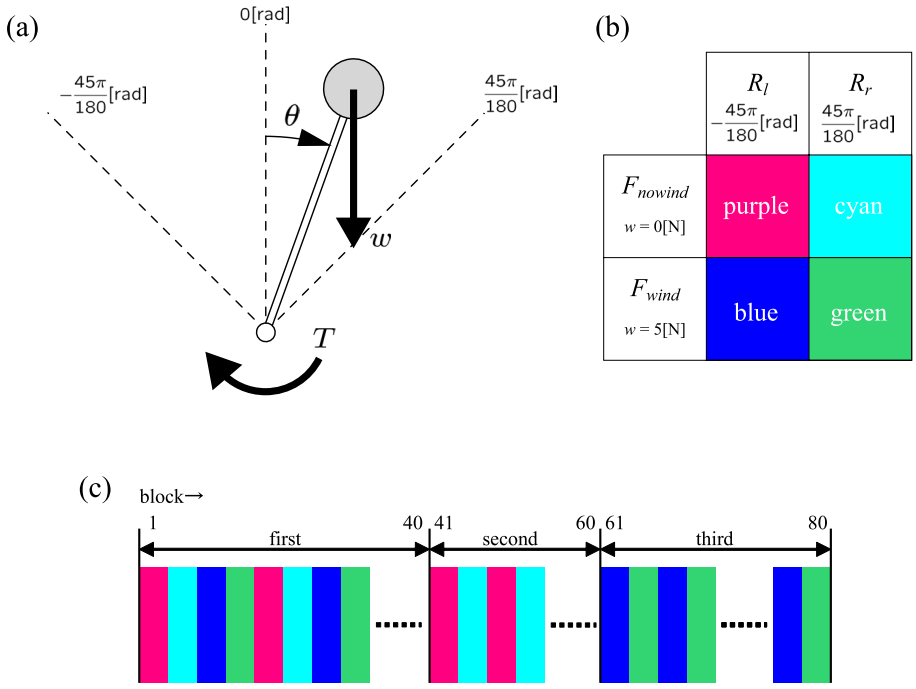

Figure 2: Simulation time step Δt is 0.05 sec, and time derivative of state $\dot{\mathbf{x}}(t)$ is given by Euler method. Mass m is 1 kg, length l is 1 m, friction coefficient μ is 0.1, and acceleration of gravity g is 9.81 m/sec^2. Torque is limited $|T| < 10$ Nm

.

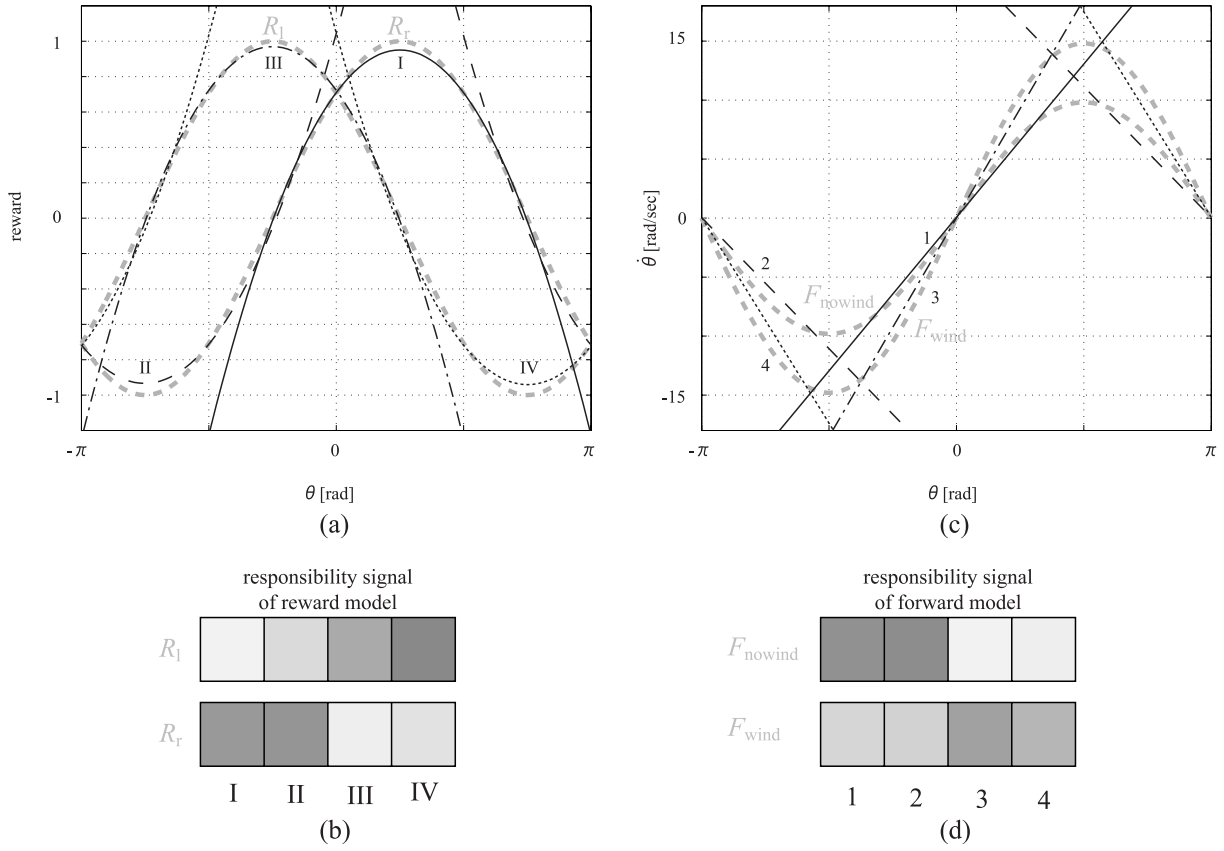

Figure 3: Learning results of reward and forward modules. (a) Gray dashed lines represent each reward function (R_l, R_r), and black lines represent each reward module ($r_\mathrm{I} \sim r_\mathrm{IV}$). Horizontal axis represents θ. (b) Mean of responsibility signal after learning for each reward function. Darker black means more frequently used module. (c) Gray dashed lines represent two environmental dynamics (F_nowind, F_wind), and black lines represent each forward module ($f_1 \sim f_4$). (d) Mean of responsibility signal for each environmental dynamics (same format as (b)).

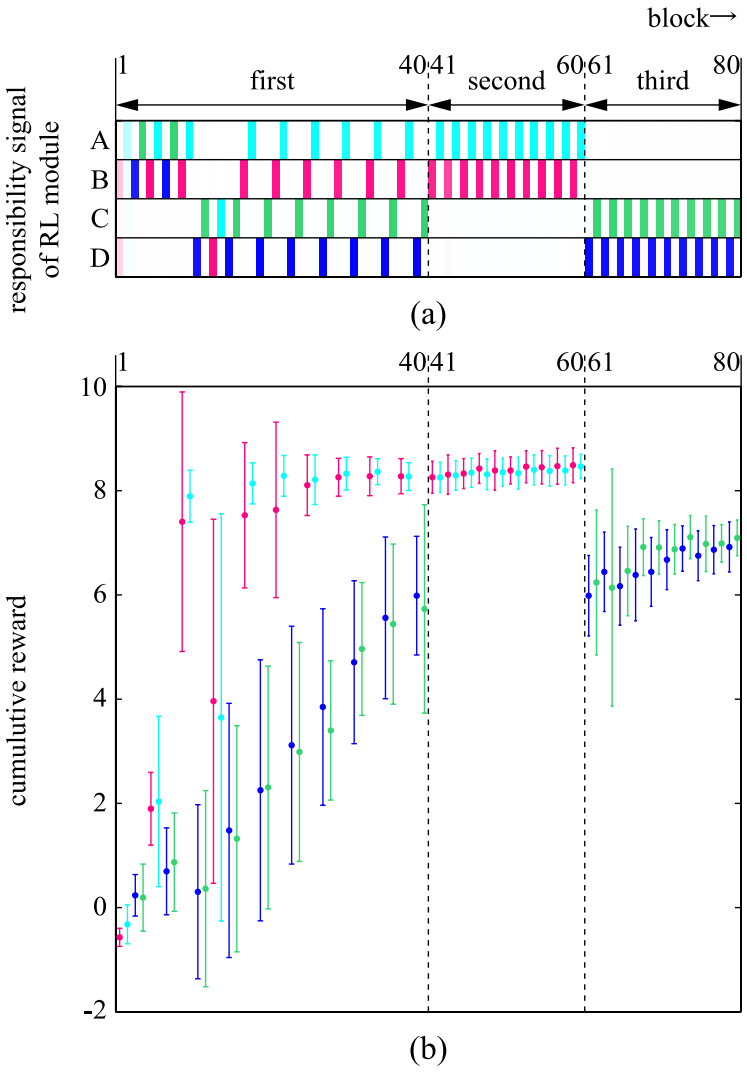

Figure 4: (a) Typical example of responsibility signal of RL module $\lambda_k^c(t)$. In each block, mean of $\lambda_k^c(t)$ is plotted. Color of each block represents a sub-environment, and darkness means how often module was selected. (b) Mean of cumulative reward for each block over 10 simulation runs. Vertical lines show standard deviations.

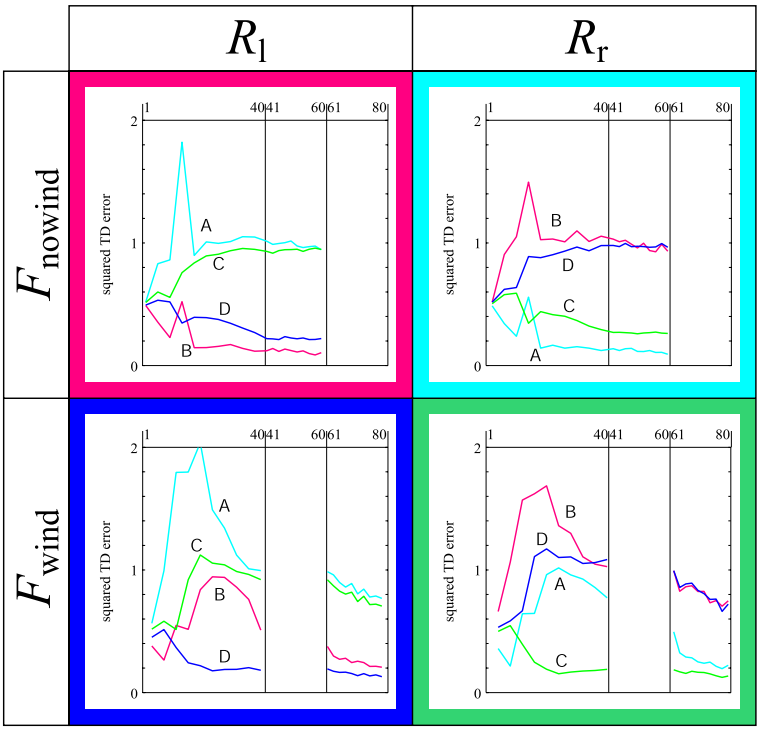

Figure 5: Learning curves in terms of TD error.

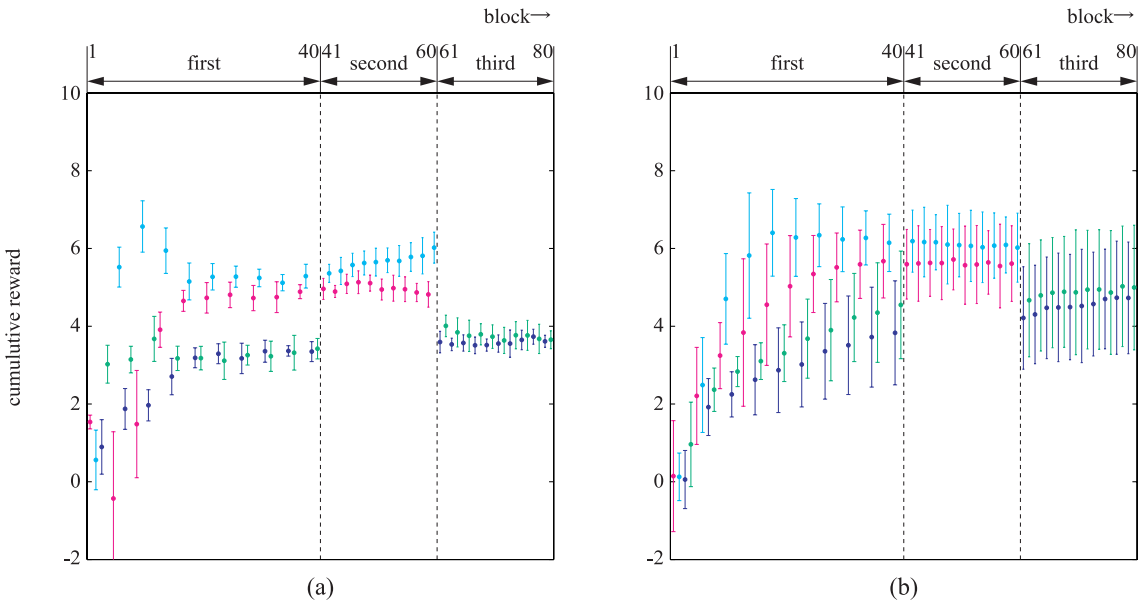

Figure 6: Learning curves of cumulative reward. (a) and (b) are results of flat RL and MMRL, respectively.

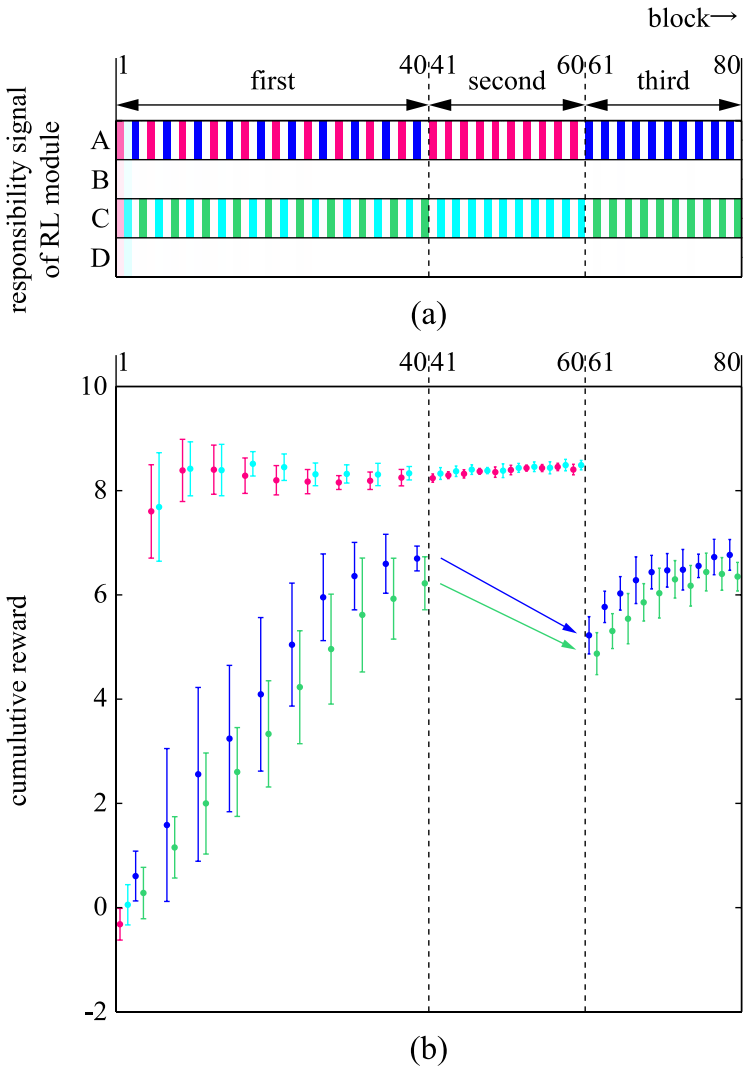

Figure 7: Responsibility signal and cumulative reward when time constant of a value function is short ($\tau = 0.5$).

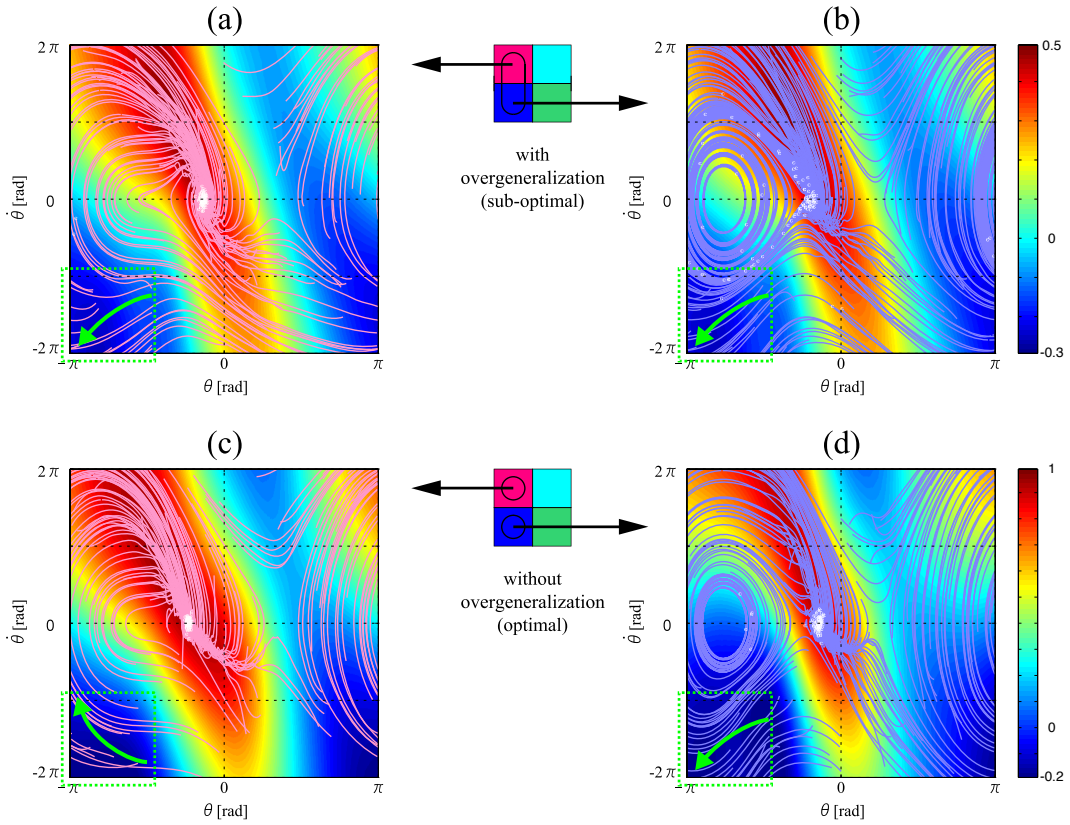

Figure 8: Top row shows value functions with overgeneralization ($\tau = 0.5$) for single RL module controlling both purple and blue sub-environments. Solid line and white dot represent swing-up trajectories and endpoint of trajectory, respectively. Bottom row shows value functions without overgeneralization ($\tau = 1$). In the center of top and bottom row, black ellipse and circle represent the sub-environment that the RL module control.

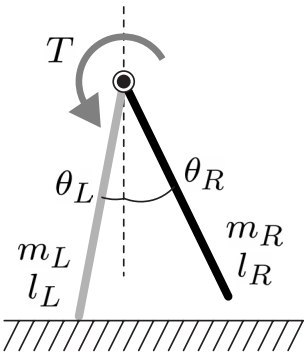

Figure 9: A two-link robot for locomotion. Positive is clockwise rotation. Mass of each leg, m_L and m_R, is 0.5 kg, length of each leg, l_L and l_R, is 0.1016 m.

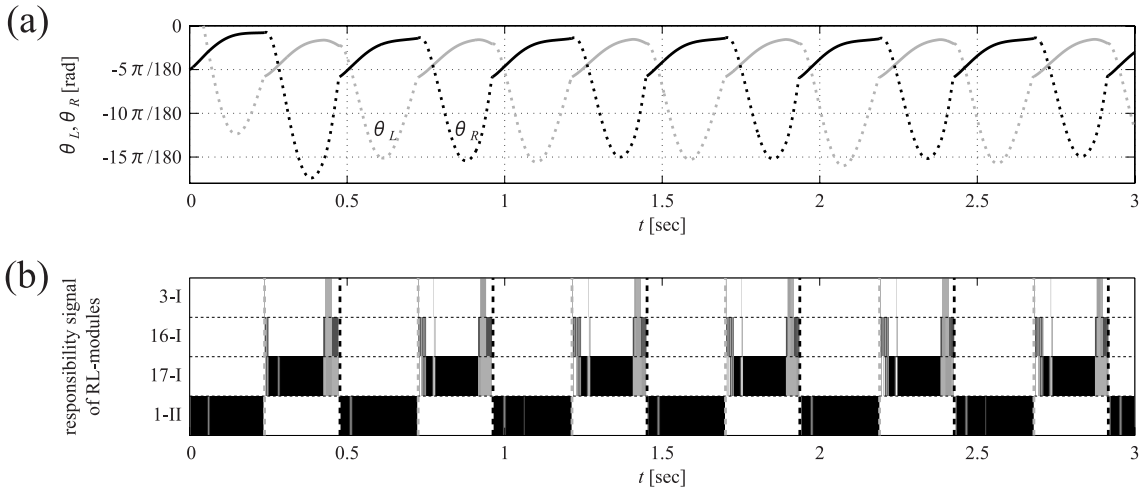

Figure 10: (a) Time course of angle of legs (θ_L and θ_R). Gray and black lines show left and right legs, respectively. Solid and dashed lines represent touching and swinging leg, respectively. (b) Responsibility signal of each RL module. Darker color indicates higher value. Gray and black dashed lines indicate that the left and right legs land on the ground.

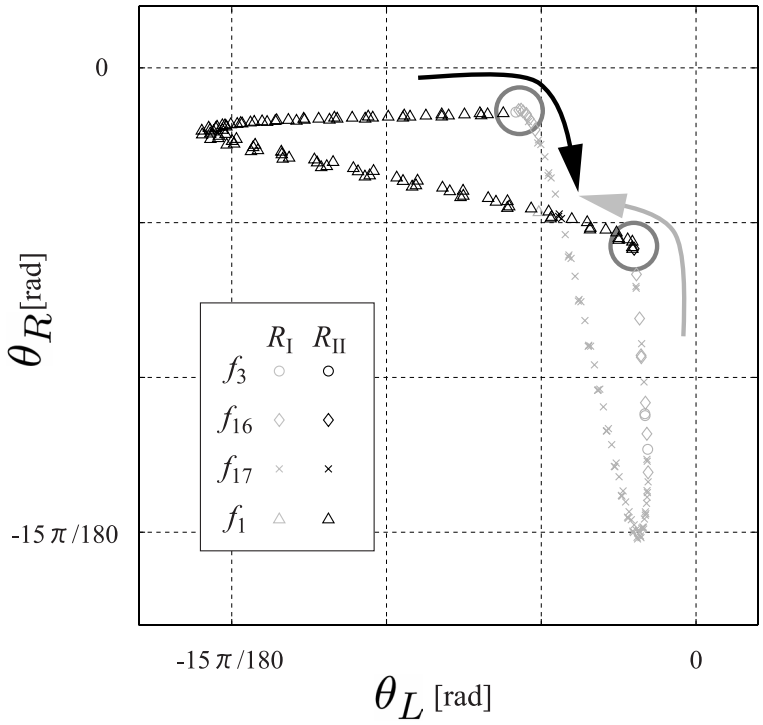

Figure 11: Trajectory and dividing. Large circle around gray arrow shows that left leg has landed, the other large circle shows that right leg has landed. Color of each marker (gray and block) represents each reward module, and shape of each marker represents each forward module.