PERGAMON

AVAILABLE AT
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Neural Networks 16 (2003) 985–994

Neural Networks

www.elsevier.com/locate/neunet

Inter-module credit assignment in modular reinforcement learning

Kazuyuki Samejima[a,b,*], Kenji Doya[a,b], Mitsuo Kawato[a]

[a]*Human information science laboratories, ATR International,2-2-2 Hikaridai, Seika, Soraku, Kyoto 619-0288, Japan*
[b]*Creating the Brain, CREST, Japan Science and Technology Corporation,2-2-2 Hikaridai, Seika, Soraku, Kyoto 619-0288, Japan*

Received 27 July 2002; revised 15 November 2002; accepted 15 November 2002

Abstract

Critical issues in modular or hierarchical reinforcement learning (RL) are (i) how to decompose a task into sub-tasks, (ii) how to achieve independence of learning of sub-tasks, and (iii) how to assure optimality of the composite policy for the entire task. The second and last requirements are often under trade-off. We propose a method for propagating the reward for the entire task achievement between modules. This is done in the form of a 'modular reward', which is calculated from the temporal difference of the module gating signal and the value of the succeeding module. We implement modular reward for a multiple model-based reinforcement learning (MMRL) architecture and show its effectiveness in simulations of a pursuit task with hidden states and a continuous-time non-linear control task.
© 2003 Elsevier Science Ltd. All rights reserved.

Keywords: Reinforcement learning; Hierarchical/modular architecture; MOSAIC non-linear control task; Inter-module credit assignment; Modular reward

1. Introduction

In order to scale up reinforcement learning (RL) to real-world problems, modular or hierarchical RL algorithms have been proposed which decompose a complex task into simpler sub-tasks, and which reuse sub-modules for similar tasks. Crucial issues in modular or hierarchical RL are (i) how to decompose a task into sub-tasks, (ii) how to achieve independence of learning of sub-tasks, and (iii) how to assure optimality of the composite policy for the entire task. The second and third requirements are often subject to trade-off.

We have proposed multiple model-based reinforcement learning (MMRL), which adaptively decomposes a task based on the predictability of the environmental dynamics (Doya, Samejima, Katagiri, & Kawato, 2002). Here we propose a new scheme for enabling the independent learning of each module while assuring the optimality of the entire task. This scheme can be applied to MMRL and other modular RL.

Previous modular or hierarchical RL methods provided only partial solutions to the above issues. In Feudal Q learning (Dayan & Hinton, 1993), sub-tasks are learned independently based on the sub-goals set by the upper level, but there is no guarantee of optimality of the composite policy for the entire task. CQ learning (Singh, 1992) requires that the weighted sum of the modular value functions equals that of the entire task, which makes learning of modules dependent on each other. MAXQ learning (Wiering & Schmidhuber, 1997) achieves independent learning of modules by aiming for a weaker form of optimality, recursive optimality. In order for MAXQ learning to find a global optimal policy, it is necessary to design appropriate 'pseudo rewards' for sub-tasks. In both Feudal Q and MAXQ learning, task decomposition is predefined by the designer. In CQ learning, task decomposition is realized with the help of an 'augmenting bit' reporting the change in the context. In 'option' (Sutton & Precup, 1999) and HAM (Parr & Russell, 1997) approaches, modular policies are not learned at all.

In a generic form of MMRL (Doya et al., 2002), modular value functions are learned so that their weighted sum represents the value function for the entire task, the same as in CQ learning. We have also proposed a variant of MMRL, multiple linear quadratic controllers (MLQC) (Doya et al., 2002), which learns locally linear dynamic models and locally quadratic reward models for efficient design of locally optimal policies. Although

* Corresponding author. Address: Creating the Brain, CREST, Japan Science and Technology Corporation, 2-2-2 Hikaridai, Seika, Soraku, Kyoto 619-0288, Japan. Tel.: +81-774-95-1211; fax: +81-774-95-1259.
E-mail address: samejima@atr.co.jp (K. Samejima).

0893-6080/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S0893-6080(02)00235-6

MLQC demonstrated efficient learning of non-linear and non-stationary control tasks, its success depended on the landscape of the reward function so that the combination of locally optimal policies are globally optimal. For example, MLQC does not work in a task in which the reward is given only at the goal.

In this paper, we propose a new modular RL method for realizing optimality of the composite value function and policy while promoting independence of learning in separate modules. We introduce a concept, 'modular reward', which is the sum of the actual reward and the imaginary reward for passing the task on to an appropriate module. The imaginary reward is given by the product of the modular value function and the temporal difference (TD) in the module gating signal. This is a generalization of the value function update methods in MAXQ and CQ learning to cases of continuous module gating and non-unique end points of sub-tasks.

We derive a condition for the modular reward so that the standard RL of each module with the modular reward enables correct estimation of the global value function for the composite policy. We consider three candidate methods for the distribution of modular reward and show in simulation that the one that promotes the back-up of the modular reward to the finished module gives the best performance.

In Section 2, we formulate the class of modular RL architectures which uses a continuous module gating signal, including MMRL. In Section 3, we define modular reward and derive the constraints for estimation of the global value function for the composite policy while each module performs standard RL independently. Implementation of modular reward in MMRL is described in Section 4. The effectiveness of modular reward is tested in discrete and continuous state cases in Section 5. We discuss the remaining problems and possible future work in Section 6.

2. Modular reinforcement learning

2.1. Reinforcement learning

RL (Sutton & Barto, 1998) is a learning paradigm which uses restricted feedback information as an evaluation of a system's output. When the system observes environmental state $x(t)$ and outputs an action $u(t)$, the system transits into state $x(t+1)$ and receives a reward $r(t)$ as an evaluation of the output. The aim of the system is to learn the state-action map or 'policy', in order to receive maximal cumulative reward through acting in the environment

$$V^*(x(t)) = E\{r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \cdots\}, \quad (1)$$

where γ is a discount factor which determines how long the system should account for rewards in future steps.

Though the system cannot know the true value of expectation of future reward, the approximated reward expectation called 'value function' $V(x(t)$ can be learned from the TD of approximated value function $\gamma V(x(t+1) - V(x(t))$ and reward $r(t)$,

$$\delta(t) = r(t) + \gamma V(x(t+1)) - V(x(t)). \quad (2)$$

The TD error δ approaches zero when the value function $V(x)$ approaches the true value of reward expectation $V^*(x)$. This is because the TD of the true expectation value of future reward is zero due to the following Bellman equation

$$V^*(x(t)) = r(t) + \sum_{x'} \gamma P(x'|x(t), u(t)) V^*(x'), \quad (3)$$

where $P(x'|x(t), u(t)$ is a transition probability when the system transits to x' when it is in $x(t)$ and performs the action $u(t)$.

2.2. Modular reinforcement learning with continuous gating signal

Modular/hierarchical architectures for RL can be categorized into two kinds according to the module gating strategy used. The first is the switching type in which the system selects one module at a time and a change in the selection occurs when the termination predicate is satisfied (Parr & Russell, 1997; Sutton & Precup, 1999) or when the system reaches a sub-goal state set by the upper layer (Dayan & Hinton, 1993; Morimoto & Doya, 1998; Wiering & Schmidhuber, 1997). The second is a modular architecture with a continuous gating signal (Doya et al., 2002; Singh, 1992), in which the output and learning of each module are weighted by 'responsibility', that is, by determining which module is responsible for the current situation. We denote the weighting for the i-th module as the responsibility signal λ_i ($0 \le \lambda_i \le 1$, $\sum_i \lambda_i = 1$).

In CQ-L (Singh, 1992), the responsibility signal is given by the gating network (Jacobs, Jordan, Nowlan, & Hinton, 1991) based on action value predictability, while in MMRL (Doya et al., 2002), it is given by the competition of multiple dynamic state predictors in each module (Wolpert & Kawato, 1998). Implementation of the responsibility signal in MMRL is described in Section 4.1.

The output of the system or policy is decided stochastically by the distribution that is the weighted summation of each module action selection probability as

$$P(u(t)|x(t)) = \sum_i \lambda_i \pi_i(u(t), x(t)), \quad (4)$$

where π_i is the conditioned probability $P(u(t)|x(t), i)$ of taking action $u(t)$ when module i is selected in state $x(t)$.

2.3. Weighted temporal difference learning

In order to make the learning of each module consistent with the goal of the entire task, a commonly used condition

is that in which the weighted sum of the modular value functions V_i is equal to the value function for the entire task

$$V(x(t)) = \sum_{i=1}^{n} \lambda_i(t) V_i(x(t)). \tag{5}$$

One straightforward method of achieving this condition is first to compute the TD error (Sutton and Barto, 1998) for the entire task

$$\delta(t) = r(t) + \gamma V(x(t+1)) - V(x(t)), \tag{6}$$

and then to distribute it to the modules in proportion to the responsibility signals (Doya et al., 2002)

$$\delta_i(t) = \lambda_i(t)\delta(t). \tag{7}$$

The modular value functions V_i are learned by the modular TD error Eq. (11) and gradient of parameter θ_i^V as

$$\theta_i^V \leftarrow \theta_i^V + \alpha\lambda_i(t)\delta_i(t)\frac{\partial V_i(x(t);\theta_i^V)}{\partial \theta_i^V}, \tag{8}$$

where $\alpha > 0$ is a parameter for the learning rate.

We can also use the modular eligibility traces for parameter

$$\varepsilon_{V_i}(t+1) = (1-\eta)\varepsilon_{V_i}(t) + \eta\lambda_i(t)\frac{\partial V_i(x(t);\theta_i^V)}{\partial \theta_i^V} \tag{9}$$

where η is a parameter for discounting the eligibility. Parameter updating with eligibility $\varepsilon_{\mathbf{v}_i}$ is given by

$$\theta_i^V \leftarrow \theta_i^V + \alpha\delta_i(t)\varepsilon_{V_i}. \tag{10}$$

In this way, if the total value function (5) becomes accurate, the TD error Eq. (6) becomes close to zero, and the learning of each module based on the TD error Eq. (7) converges.

One of the advantages of modular architecture is re-usability of modular policy for another task with the same elemental sub-tasks. However, one problem in this method of how a task can be decomposed is left open to the particular choice of function approximators used for each module. As is apparent from Eq. (5), for a given responsibility signal vector $\boldsymbol{\lambda}(t) = (\lambda_1(t), \ldots, \lambda_n(t)$, a different combination of modular value functions $(V_1(\mathbf{x}(t), \ldots, V_n(\mathbf{x}(t)))$ can achieve the total value function $V(\mathbf{x}(t))$. Even if the total value function is learned by weighted TD error Eq. (7), the modular policy based on the modular value function may not learn any goal-directed policy.

3. Modular reward

Here, we propose a method for appropriately backing up the value of the next module to the preceding module. We base our derivation on three constraints: (1) each module and the entire system follow a similar TD learning algorithm, (2) the TD learning in each module assures consistent learning of the total value function, and (3) among simultaneously activated modules, the one that is finishing a sub-task should take the largest credit.

To achieve (1), we formulate a modular TD algorithm in which each modular value function $V_i(t)$ learns to reduce modular TD error

$$\delta_i(t) = r_i(t) + \gamma V_i(x(t+1) - V_i(x(t)), \tag{11}$$

where $r_i(t)$ is a 'modular reward' which is derived by constraints (2) and (3). For constraint (2), we require that the weighted sum of the modular TD error

$$\delta(t) = \sum_i \lambda_i(t)\delta_i(t) \tag{12}$$

is equivalent to the total TD error Eq. (6). This way, the reduction of the modular TD error ensures the reduction of the total TD error. From definitions (5) and (6), we have

$$\begin{aligned}\delta(t) &= r(t) + \gamma\sum_i \lambda_i(t+1)V_i(x(t+1)) - \sum_i \lambda_i(t)V_i(x(t)) \\ &= r(t) + \gamma\sum_i (\lambda_i(t+1) - \lambda_i(t))V_i(x(t+1)) \\ &\quad + \sum_i \lambda_i(t)[\gamma V_i(x(t+1)) - V_i(x(t))].\end{aligned} \tag{13}$$

Therefore, constraint (2), i.e. Eq. (12) is satisfied by defining the modular rewards $r_i(t)$ such that

$$\sum_i \lambda_i(t)r_i(t) = r(t) + \gamma\sum_i (\lambda_i(t+1) - \lambda_i(t))V_i(x(t+1)). \tag{14}$$

The second term on the right-hand side is positive if a module with a high value ($V_i > 0$) is activated ($\lambda_i(t+1) - \lambda_i(t) > 0$). Accordingly it can be regarded as a 'imaginary reward' at the time of module transition.

One possible method of distributing the right-hand side is Eq. (14) uniform distribution, as

$$r_i(t) = r(t) + \gamma\sum_j (\lambda_j(t+1) - \lambda_j(t))V_j(x(t+1)). \tag{15}$$

However, a more reasonable way to proceed is to distribute it to the module that is finishing a sub-task. To achieve constraint (3), we define a backing-up modular reward

$$r_i(t) = r(t) + \frac{\lambda_i^-(t)}{\sum_k \lambda_k^-(t)^2} \times \left[\gamma\sum_j (\lambda_j(t+1) - \lambda_j(t))V_j(x(t+1))\right] \tag{16}$$

where $\lambda_i^-(t)$ is the decreasing responsibility signal

$$\lambda_i^- = \begin{cases} \lambda_i(t) & \text{if } \lambda_i(t+1) < \lambda_i(t) \\ 0 & \text{otherwise} \end{cases}. \quad (17)$$

4. Multiple model-based reinforcement learning with modular reward

In this section, we implement modular rewards for a MMRL architecture (Doya et al., 2002). Fig. 1 shows the overall organization of the MMRL architecture. It is composed of n modules, each of which consists of a state prediction model and an RL controller.

Basically, this architecture decomposes a non-linear and/or non-tationary task into multiple domains in space and time so that within each of the domains the environmental dynamics is well predictable.

4.1. Responsibility signal by predictability

The action outputs of the RL controllers as well as the learning rate of both the predictors and controllers are weighted by the 'responsibility signal', $\lambda_i(t)$, defined by the relative accuracy of prediction by the modular predictors, $f_i(x', x, u)$, which approximates transition probability $P(x'|x, u)$ from state $\mathbf{x}$ to $\mathbf{x}'$ when the system takes action $\mathbf{u}$.

After the observation of actual transition to $x(t+1)$, we can get a posterior probability of the module selection based on the prediction model as a generative model of state transition. This posterior probability of module selection is called the 'responsibility signal' and is defined by a normalized probability of the output of state prediction models $f_i(t)$ as a likelihood of module selection

$$\lambda_i(t) = \frac{\hat{\lambda}_i(t) f_i(x(t+1), x(t), u(t))}{\sum_j^n \hat{\lambda}_j(t) f_j(x(t+1), x(t), u(t))}, \quad (18)$$

where $\hat{\lambda}_i(t)$ is the prior probability of selection of module i, which we call the responsibility predictor. One example of responsibility predictor is smoothing responsibilty predictor which use responsibilty signal in previous time step as a prior probability of present module selection (see Appendix A).

4.2. Model-based reinforcement learning

Model-based RL is an efficient RL algorithm using a state prediction model and a reward model to update the value function (Doya, 2000) and select action by longer time step planning (Sutton, 1991).

In MMRL (Doya et al., 2002), parameters θ_i^f and θ_i^r for multiple prediction models $f_i(x', x, u; \theta^f)$ and reward models $\hat{r}_i(x, u; \theta^r)$ are updated by actual observation of the next state $x(t+1)$ and the reward $r(t)$. The responsibility signal $\lambda_i(t)$ is used for weighting the parameter update of these models. In discrete state task, we implement state prediction models using histograms of observed transitions. In continuous state task, we use linear model with deviations. These implementation methods are described in Sections 5.1.2 and 5.2.2.

The reward models $\hat{r}_i$ are updated by weighted error between estimation $\hat{r}_i(x(t), u(t))$ and modular reward $r_i(t)$ of

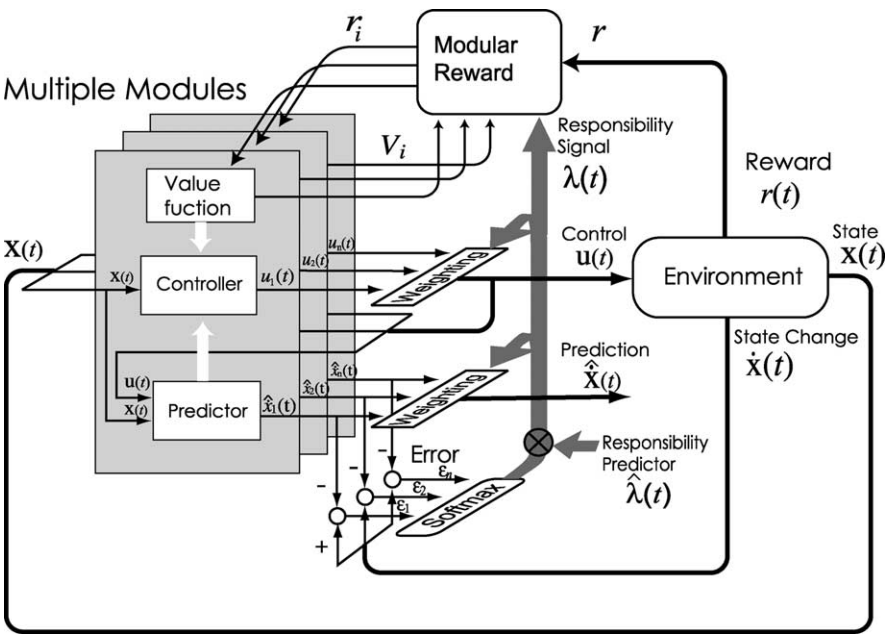

Fig. 1. MMRL.

Eqs. (15) or (16)

$$\theta_i^r \leftarrow \theta_i^r + \alpha_r \lambda_i (\hat{r}_i(x,u) - r_i(t)) \frac{\partial \hat{r}_i}{\partial \theta_r}, \tag{19}$$

where $\alpha_r > 0$ is the learning rate parameter.

Using state prediction models f_i and reward models $\hat{r}_i$, each module can estimate the action value function, $\hat{r}_i(x,u') + \sum_{x'} f_i(x',x,u')V_i(x')$, for action u' by simulating value function $V_i(x')$ in next state x' and getting reward $\hat{r}_i(x,u')$. For example, if we use a greedy policy as a modular policy, the modular policies $\pi_i(u,x)$ select the best action-value in next state x' and expected reward $\hat{r}_i$ when simulated action u' is selected

$$u_i = \arg\max_{u'} \{\hat{r}_i(x,u') + \sum_{x' \in X} f_i(x',u',x)V_i(x')\}, \tag{20}$$

where X is a set of possible states.

5. Simulations

5.1. Pursuit problem

In order to test the effectiveness of MMRL with modular rewards, we investigate its performance in pursuit problem with hidden states. This task consists of four different sub-tasks reward is given at the end of only one sub-task. This problem cannot be learned by learning methods such as MLQC in which policies are derived from local rewards.

5.1.1. Task

The agent's task is to catch a moving target in a grid world of a 7×7 torus (Fig. 2). The possible actions U of the agent are one-step movements in one of four directions {N, E, S, W}. The state observation x of the agent is the relative position of the target. There are four kinds of targets $\{T_1, T_2, T_3, T_4\}$, which move in either one of four directions{NE, SE, NW, SW}. First, a target appears at a random position in the grid world. If the agent catches the target, another target appears at a random position. The agent cannot directly observe which one of the four targets is present. In the simulation below, the targets were presented in a deterministic order $\{T_1, T_2, T_3, T_4, T_1, T_2, \ldots\}$ as shown in Fig. 2(c).

The reward $r = 10$ was given only at the time where T_4 was caught. Because these is no reward, just a cost for each movement -0.01, when the agent catches T_1, T_2 and T_3, the modules solving the sub-task for catching these targets could not learn catching behavior just from getting a local reward in the sub-task.

5.1.2. Implementation of prediction model and RL controller

The state prediction model $f_i(x',x,u)$ was implemented by using a table $\Theta_i^f(x',x,u)$ of histogram of observation of state x, action u, and next state x'. The table $\Theta_i^f(x,u,x')$ for prediction model f_i was updated with weighting by responsibility

$$\Theta_i^f(x(t+1),x(t),u(t)) \leftarrow \Theta_i^f(x(t+1),x(t),u(t)) + \lambda_i(t) \times (-\xi \Theta_i^f(x(t+1),x(t),u(t)+1), \tag{21}$$

where $0 < \xi < 1$ is the forgetting rate. When the agent's state is x and action u is taken, the output of the model as a probability of going to x' is

$$f_i(x',x,u) = \frac{\Theta_i^f(x',x,u)}{\sum_{s \in X} \Theta_i^f(s,x,u)}, \tag{22}$$

where X is the set of possible states. The initial parameter $\Theta^f(x',x,u)$ is set as a small random value with uniform distribution between 0.0 and 1.0. Using this prediction model, we can get the responsibility signal with a smoothing responsibility predictor (see Appendix A) by Eqs. (A2)–(A.4).

The modular value functions $V_i(x)$, which were represented by a table for each observation x, are updated by Eq. (10). The eligibility traces for updating the modular value functions were used with $\eta = 0.5$.

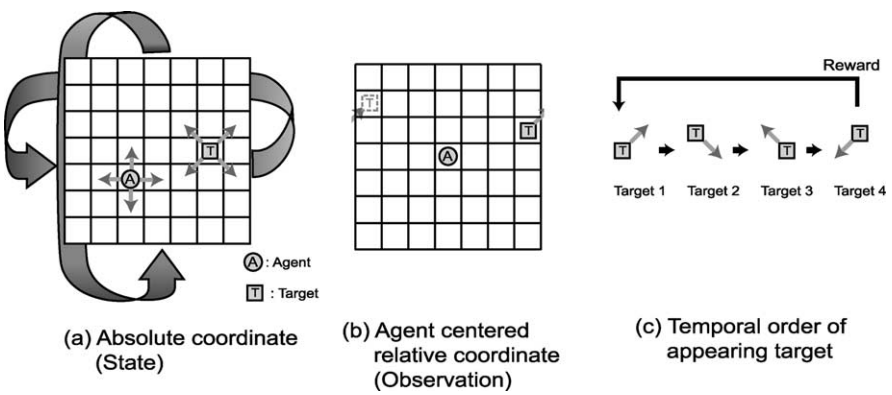

Fig. 2. Pursuit problem: (a) action of target and an agent in a grid world of a torus; (b) observation of hunter agent; (c) appearance order of targets.

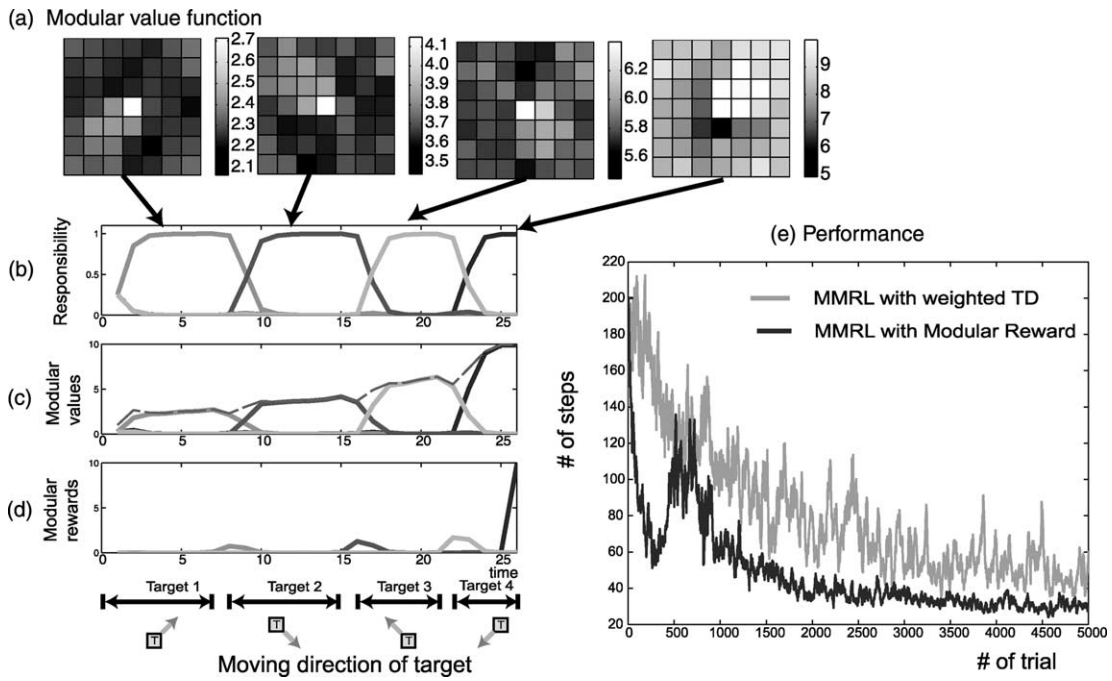

Fig. 3. Result of learning MMRL with modular reward in Pursuit problem: (a) modular value function. Time course of (b) responsibility signal λ_i, (c) modular value function $\lambda_i(t)V_i(t)$ (thick lines) and total value function $V(t) = \sum_i \lambda_i V_i$ (thin line); and (d) modular reward r_i; (e) performance measured by cumulative reward in one trial.

We implemented a model-based RL controller for each module while using modular state dynamic predictors and reward predictors. The outputs of modular RL controllers $\mathbf{u}_i$ are chosen based on the action value $Q_i(x,u) = \hat{r}_i(x,u) + \sum_{x' \in X} f_i(x', x, u) V_i(x', x, u)$. The reward model $\hat{r}_i(x,u)$ is also implemented by using parameter table $\theta_i^r(x,u) = \hat{r}_i(x,u)$ which is updated by Eq. (19).

An action was chosen from possible actions U by the Gibbs distribution

$$\pi_i(x,u) = \frac{e^{\beta Q_i(x,u)}}{\sum_{a \in U} e^{\beta Q_i(x,a)}}, \tag{23}$$

where $\beta > 0$ is a parameter for controlling the randomness of an action.

Parameters were set as $\alpha = 0.2$, $\gamma = 0.95$, $\eta = 0.5, \rho = 0.5$, and $\xi = 0.001$. The action disturbance parameter β was scheduled as $\beta(i) = n_{\text{trial}}/500$, where n_{trial} was the number of trials for annealing.

5.1.3. Results

MMRL with backing-up modular reward succeeded in learning both modular control policies and the transitions between these policies (Fig. 3(b)). Modular value functions had a high value in the catching position (0,0), and their levels are consistent with the total value function (see scale bars in Fig. 3(a)). The time course of the total value function monotonously increased toward the goal state except at the time when a new target appeared in a random position (Fig. 3(c) thin line). Fig. 3(e) compares the performances of MMRL with modular reward Eq. (16) and MMRL with weighted total TD error Eq. (7). The MMRL with modular reward achieved near-optimal policy faster than the MMRL with weighted total TD error.

We tested the proposed method with wide ranges of parameter settings, namely, learning rate for value function $\alpha = 0.01, 0.1, 0.7$, time scale of eligibility traces $\eta = 0.01, 0.2, 0.5$, and timescale of responsibility signal $\rho = 0.01, 0.1, 0.7$.

Successful swing-up was achieved except $\alpha = 0.7$ and $\rho = 0.7$, where a large time constant of responsibility resulted in delayed selection of appropriate modules.

We also tested CQ-L (Singh, 1992) with this task. CQ-L failed to assign four modules to four different targets, even in the easiest case when the reward was given after catching each of the four targets.

5.2. Pendulum swing-up task with limited torque

In the pursuit task in the previous section, each sub-tasks had only one sub-goal. In this section, we show our approach is effective in the case where module switching occurs not only at a particular point, but on distributed sets in continuous space. We implement MMRL with modular reward in a pendulum swing-up task in which reward is given only near the swinging up position (Fig. 4) (Doya, 2000; Doya et al., 2002).

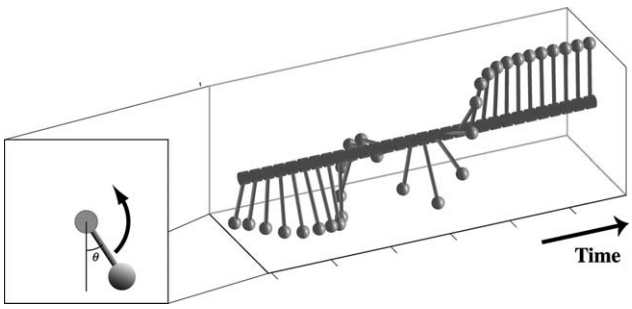

Fig. 4. Pendulum swing-up task with limited torque.

5.2.1. Task

The state space was two-dimensional, i.e. $\mathbf{x} = (x, \dot{x})^T \in S \times \mathbf{R}$, where $x \in S$ is the joint angle with periodic space in $S = [-\pi, \pi]$ and $\dot{x} \in \mathbf{R}$ is angular velocity. The driving torque $\mathbf{u} = T$ is limited in $[-T^{\max}, T^{\max}]$ with $T^{\max} < mgl$. The pendulum has to be swung back and forth at the bottom to build up enough momentum for a successful swing up.

Although the task setting was almost the same as in (Doya et al., 2002), the reward for the state was given only in the neighborhood of the goal state so that no reward was given near the bottom, i.e.

$$r(t) = -\mathbf{u}(t)' \Theta^c \mathbf{u}(t) + \begin{cases} 1 & \text{if } -\frac{\pi}{8} < (x \bmod 2\pi) - \pi < \frac{\pi}{8} \\ 0 & \text{otherwise} \end{cases}, \quad (24)$$

where the cost parameter $\Theta^c = 0.01$. The initial state was set randomly with $x \in [-\pi/4, \pi/4]$, $\dot{x} \in [-1, 1]$.

5.2.2. Implementation of prediction models and RL controller

MMRL can approximate the non-linear system dynamics in this case with two modular predictors of a linear model using at least two modules (Doya et al., 2002).

In this example, we use MMRL with continuous time and space in which the state prediction model tries to predict state dynamics $\dot{\mathbf{x}} = \mathrm{d}\mathbf{x}/\mathrm{d}t$. Here, we describe the modular predictors using linear modular state dynamic predictors Eq. (25) with Gaussian noise of fixed variance

$$f_i(\dot{\mathbf{x}}', \mathbf{x}(t), \mathbf{u}(t)) = \frac{1}{Z} \exp((\dot{\mathbf{x}} - A_i\mathbf{x}(t) + B_i\mathbf{u}(t))' \Sigma^{-1} (\dot{\mathbf{x}} - A_i\mathbf{x}(t) + B_i\mathbf{u}(t))), \quad (25)$$

where A_i, B_i are coefficient matrices for the linear prediction model, Σ is the covariance matrix for Gaussian noise, and Z is the normalizing constant.

The responsibility signal is given by a soft-max function of prediction error $E(t) = \dot{\mathbf{x}}(t) - \{A_i\mathbf{x}(t) + B_i\mathbf{u}(t)\}$

$$\lambda_i(t) = \frac{1}{Z} \hat{\lambda}_i \exp\left(-\tfrac{1}{2} E_i(t)^T \Sigma^{-1} E_i(t)\right) \quad (26)$$

where Z is the normalizing term

$$Z = \sum_i \hat{\lambda}_i \exp\left(-\frac{1}{2} E_i(t)^T \Sigma^{-1} E_i(t)\right).$$

We use the smoothing responsibility predictor (see Appendix B) with diffusing parameter $\tau_\rho = 1$.We use action output with Gaussian distribution

$$\pi_i(\mathbf{u}(t), \mathbf{x}(t)) = \frac{1}{Z} \exp((\mathbf{u}(t) - \hat{\mathbf{u}}_i(x(t)))^T \beta (\mathbf{u}(t) - \hat{\mathbf{u}}_i \mathbf{x}(t))) \quad (27)$$

with variance β and average control output

$$\hat{\mathbf{u}}_i(\mathbf{x}(t)) = B_i^T \left(\frac{\partial V}{\partial \mathbf{x}(t)}\right)^T \quad (28)$$

using the steepest value gradient ascending (Doya, 2000).

State prediction and action output of the entire system is given by the expectated value of the mixture of the Gaussians, namely

$$\tilde{\mathbf{x}}(t+1) = \sum_{i=1}^{n} \lambda_i(t)(A_i\mathbf{x}(t) + B_i\mathbf{u}(t)),$$

$$\mathbf{u}(t) = \sum_{i=1}^{n} \lambda_i(t)\hat{\mathbf{u}}_i(\mathbf{x}(t)) + \nu(t),$$

where $\nu(t)$ is Gaussian noise with variance β.

Parameters of state prediction model f_i are analytically derived by the system dynamic equation around the hanging down position $\mathbf{x} = (0, 0)$ and the swinging up position $\mathbf{x} = (\pi, 0)$ as

$$A_i = \begin{pmatrix} 0 & 1 \\ a_i & -0.1 \end{pmatrix}, \qquad B_1, B_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

where $a_1 = -9.8$ and $a_2 = 9.8$. The value functions V_i are approximated by Gaussian radial basis functions and updated by a continuous-time version of MMRL (Doya et al., 2002) with modular reward Eq. (B5) or Eq. (B6) (see Appendix B). We set the action perturbation parameter $\beta = 1.0$ and the variance of Gaussian forward model with variance $\Sigma = 1$.

5.2.3. Results

We compared the performances of MMRL with two kinds of modular reward Eqs. (B5) and (B6) to that of the original MMRL with weighted TD error Eq. (7). The top row of Fig. 5 compares their performance measured by the average reward $(1/d) \int_0^d r(t)\mathrm{d}t$ ($d = 20$). The middle row of Fig. 5 shows learned value functions of individual modules in the responsible regions, which are $x \in [-\pi/2, \pi/2]$ for module 1 and $x \in [\pi/2, 3\pi/2]$ for module 2. The bottom row of Fig. 5 shows modular value $\lambda_i(t)V_i(t)$ in the successful swing-up trajectories, which are superimposed on the middle row of Fig. 5.

Using the weighted total TD error, successful swing-up was rarely achieved within 200 trials. This was

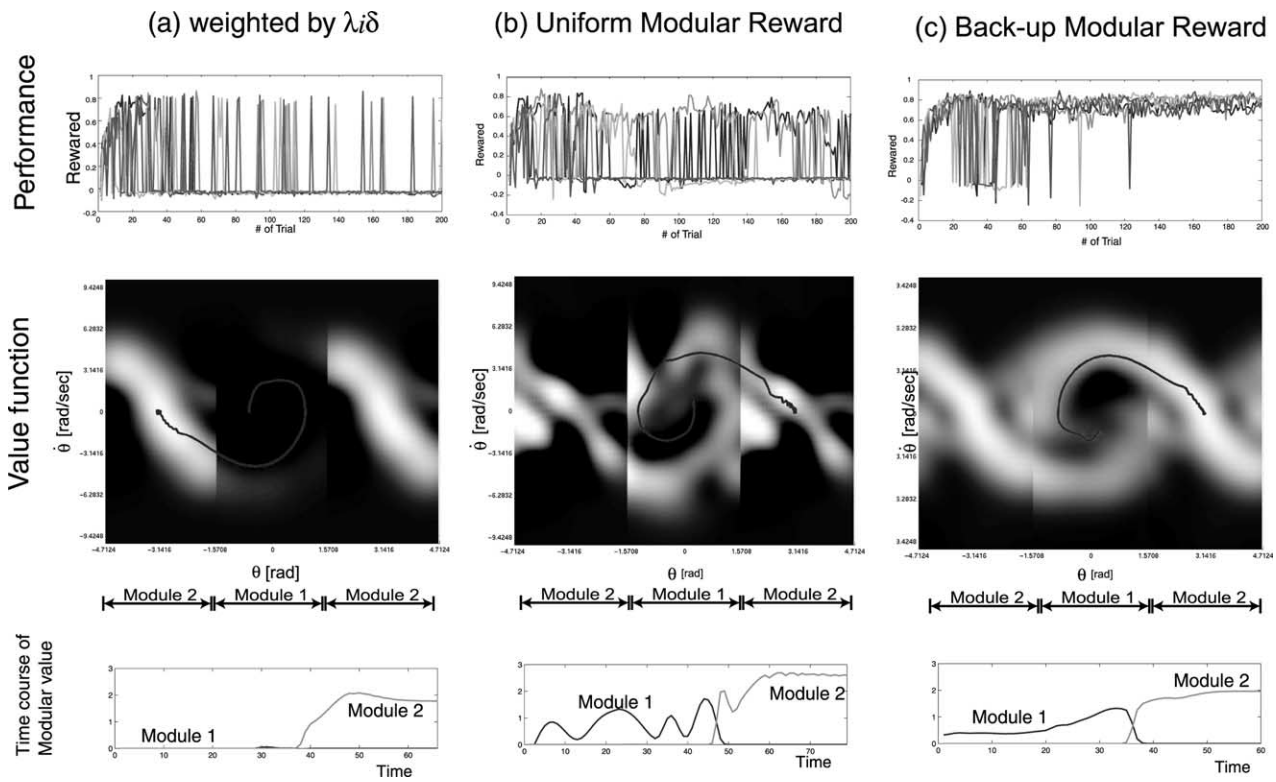

Fig. 5. Value function for each module after 200 learning trials. Top: performance as cumulative reward in five simulation runs. Middle: Modular value function in each responsible region (a lighter color indicates a higher value). Bottom: time course of modular value V_i in successful swing-up trial: (a) MMRL with total TD error; (b) MMRL with uniform modular reward; and (c) MMRL with the backing-up modular reward.

because the reward given to module 2 did not effectively motivate module 1, which was responsible in the hanging down region. By using a uniform modular reward Eq. (15), the value function near the transition points of module 1 was elevated. By comparing Figs. 5(b) and (c), we can see that the value was more effectively propagated with the backing-up modular reward Eq. (16), which enabled faster learning.

6. Conclusion

We introduced a new concept of modular reward, which enables the learning of modular policies directed toward the optimization of an entire task. A backing-up modular reward Eq. (15) is given to a module that is deactivated when another module with a higher value is activated. In the simulations of discrete-time and continuous-time tasks, we showed that a modular reward with module level backing-up enables quicker and more robust learning than MMRL using the weighted TD error of the total value function.

In the present example, the activation of modules was performed in a fixed order. An interesting future work will be the learning of sequential module activation, possibly with the introduction of an upper-level value function.

Appendix A. Smoothing responsibility predictor

A.1. Discrete-time case

We adopt the responsibility signal of the preceding step as the responsibility predictor for the present step as

$$\hat{\lambda}_i(t) = \frac{\lambda_i(t-1)^\rho}{\sum_j^n \lambda_j(t-1)^\rho}. \tag{A1}$$

This responsibility predictor restrains a rapid change in the responsibility signal. The parameter ρ ($0 < \rho < 1$) chooses the time scale for changing the responsibility signal. In this case, the responsibility signal and responsibility predictor are calculated as

$$\lambda_i(t) = \frac{e^{l_i(t)}}{\sum_j e^{l_j(t)}}, \tag{A2}$$

$$\hat{\lambda}_i(t) = \frac{e^{\rho l_i}(t-1)}{\sum_j e^{\rho l_j}(t-1)}. \tag{A3}$$

by a short-term cumulative log-likelihood $l_i(t) = \sum_{s=0}^{t} \rho^{(t-s)} \log f_i(s)$. We can calculate $l_i(t)$ incrementally as

$$l_i(t) = \log(f_i(t)) + \rho l_i(t-1) \quad \text{for } t > 0. \tag{A4}$$

A.2. Continuous-time case

In the continuous-time case, we use the Gaussian diffusing process for modular prediction models

$$f_i(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}(t+s)) = P_i(\mathbf{x}(t+s)|\mathbf{x}(t), \mathbf{u}(t)) \tag{A5}$$

$$f_i(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}(t+s)) = \frac{1}{Z}\exp(-(\mathbf{x}(t) - \mu(t,s))^T(s\Sigma)^{-1}(\mathbf{x}(t) - \mu(t,s))) \tag{A6}$$

at short time s after observation of $\mathbf{x}(t)$ and action selection $\mathbf{u}(t)$. The means state change is modeled as a linear function as

$$\mu_i(\mathbf{x}(t), \mathbf{u}(t) = A_i\mathbf{x}(t) + B_i\mathbf{u}(t), \tag{A7}$$

for module i. We set the diffusion parameter matrix Σ as the same for all modules.

The log-likelihood to select module i is

$$\log L_i(t,s) = -\frac{1}{2}sE_i(t,s)^T\Sigma^{-1}E_i(t,s), \tag{A8}$$

where

$$E(t,s) = \frac{\mathbf{x}(t+s) - \mathbf{x}(t)}{s} - (A_i\mathbf{x}(t) + B_i\mathbf{u}(t)).$$

We adopt the responsibility predictor as the diffusing probability distribution at s after selection time t as

$$\hat{\lambda}_i(t+s) = \frac{\lambda_i(t)^{e^{-s/\tau_\rho}}}{\sum_j^n \lambda_j(t)^{e^{-s/\tau_\rho}}}, \tag{A9}$$

where τ_ρ is a time constant for diffusing module selection probability.

The responsibility signal and responsibility predictor can be calculated incrementally by an equation of log-likelihood

$$l_i(t) = \log f_i(t,s) + e^{-s/\tau_\rho}l_i(t-s)$$

$$l_i(t) - l_i(t-s) = \log f_i(t,s) - (1 - e^{s/\tau_\rho})l_i(t-s). \tag{A10}$$

As a limit of $s \to +0$, Eq. (A10) becomes

$$\frac{\mathrm{d}l_i(t)}{\mathrm{d}t} = -\frac{1}{\tau_\rho}l_i(t) - \frac{1}{2}\{\dot{\mathbf{x}}(t) - \mu_i(\mathbf{x}(t), \mathbf{u}(t))\}^T \times \Sigma^{-1}\{\dot{\mathbf{x}}(t) - \mu_i(\mathbf{x}(t), \mathbf{u}(t))\}. \tag{A11}$$

The solution $l_i(t)$ of Eq. (A11) is interpreted as the short-term weighted average of the normalized squared error of linear prediction model Eq. (A7) to predict state change $\dot{\mathbf{x}}(t)$.

The responsibility signal λ_i is given by Eq. (A2) using solution $l_i(t)$ of differential Eq. (A11). In the continuous-time case, the responsibility predictor is the same as the responsibility signal.

Appendix B. Continuous-time reinforcement learning and modular reward

B.1. Continuous-time and -space RL

In continuous-time and -space TD learning (Doya, 2000), TD error is given by

$$\delta(t) = r(t) + \dot{V}(\mathbf{x}(t)) - \frac{1}{\tau}V(\mathbf{x}(t)), \tag{B1}$$

where $1/\tau$ corresponds to discount factor γ of discrete time TD learning.

B.2. Modular reward

Modular TD error is given by

$$\delta_i(t) = r_i(t) + \dot{V}_i(\mathbf{x}(t+1)) - \frac{1}{\tau}V_i(\mathbf{x}(t)). \tag{B2}$$

The weighted sum of modular TD error Eq. (B2) is equivalent to total TD error Eq. (B1) to achieve constraint 2), i.e. Eq. (12). From the definition of Eq. (5) and its change

$$\dot{V}(\mathbf{x}(t)) = \sum_{i=1}^n \{\lambda_i(t)\dot{V}_i(\mathbf{x}(t) + \dot{\lambda}_i(t)V_i(\mathbf{x}(t))\},$$

we have

$$\begin{aligned}\delta(t) &= r(t) + \sum_{i=1}^n \{\lambda_i(t)\dot{V}_i(\mathbf{x}(t)) + \dot{\lambda}_i(t)V_i(\mathbf{x}(t))\} \\ &\quad - \frac{1}{\tau}\sum_{i=1}^n \lambda_i(t)V_i(\mathbf{x}(t)) \\ &= r(t) + \sum_{i=1}^n \dot{\lambda}_i(t)\{V_i(\mathbf{x}(t)) + \sum_{i=1}^n \lambda_i t\{\dot{V}_i(\mathbf{x}(t)) - \frac{1}{\tau}V_i(\mathbf{x}(t))\}\end{aligned} \tag{B3}$$

Therefore, Eq. (12) is satisfied by defining the continuous-time modular reward $r_i(t)$ such that

$$\sum_i \lambda_i(t)r_i(t) = r(t) + \sum_{i=1}^n \dot{\lambda}_i(t)V_i(\mathbf{x}(t)). \tag{B4}$$

Moreover, the continuous-time uniform distributed modular reward is given by

$$r_i(t) = r(t) + \sum_{j=1}^n \dot{\lambda}_j(t)V_j(\mathbf{x}(t). \tag{B5}$$

To achieve constraint (3), the backing-up modular reward is given by

$$r_i(t) = r(t) + \frac{\lambda^-(t)}{\sum_j \lambda^-(t)^2}\left[\sum_{j=1}^n \dot{\lambda}_j(t)V_j(\mathbf{x}(t))\right] \tag{B6}$$

where $\lambda_i^-(t)$ is the decreasing responsibility signal

$$\lambda_i^- = \begin{cases} \lambda_i(t) & \text{if} \dot{\lambda}_i(t) < 0 \\ 0 & \text{otherwise} \end{cases}. \tag{B7}$$

References

Dayan, P., & Hinton, G. (1993). *Feudal reinforcement learning* (*vol. 5*). *Advances in neural information processing systems*, Cambridge, MA: MIT press, pp. 271-278.

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.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*, 79–87.

Morimoto, J., & Doya, K. (1998). Reinforcement learning of dynamic motor sequence: learning to stand up. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, *3*, 1721–1726.

Parr, R., & Russell, S. (1997). *Reinforcement learning with hierarchies of machines* (*vol. 10*). *Advances in Neural Information Processing Systems*, Cambridge: MIT Press, pp. 1043–1049.

Sutton, R. S. (1991). Plannning by incremental dynamic programing. In L. A. Birnbaum, R. S. Sutton, & G. C. Collins (Eds.), *Proceedings of the Eighteenth International Workshop on Machine Learning* (pp. 353–357). San Mateo, CA: Morgan Kaufmann.

Singh, S. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, *8*, 323–339.

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning*. Cambridge, MA: MIT Press.

Wiering, M., & Schmidhuber, J. (1997). HQ-learning. *Adaptive Behavior*, *6*(2).

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