Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT

Neural Networks 17 (2004) 299–305

Neural Networks

www.elsevier.com/locate/neunet

Reinforcement learning with via-point representation

Hiroyuki Miyamoto[a,b,*], Jun Morimoto[c], Kenji Doya[c], Mitsuo Kawato[c]

[a]*Kawato Dynamic Brain Project, Japan Science and Technology Corporation, Kyoto, Japan*
[b]*Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Hibikino 2-4, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0196, Japan*
[c]*ATR Computational Neuroscience Laboratories, Kyoto, Japan*

Received 25 March 2002; accepted 7 November 2003

Abstract

In this paper, we propose a new learning framework for motor control. This framework consists of two components: reinforcement learning and via-point representation. In the field of motor control, conventional reinforcement learning has been used to acquire control sequences such as cart-pole or stand-up robot control. Recently, researchers have become interested in hierarchical architecture, such as multiple levels, and multiple temporal and spatial scales. Our new framework contains two levels of hierarchical architecture. The higher level is implemented using via-point representation, which corresponds to macro-actions or multiple time scales. The lower level is implemented using a trajectory generator that produces primitive actions. Our framework can modify the ongoing movement by means of temporally localized via-points and trajectory generation. Successful results are obtained in computer simulation of the cart-pole swing up task.

Keywords: Hierarchical reinforcement learning; Via-point; Motor control; Cart-pole; Swing up; Robotics

1. Introduction

In our daily lives, it seems that we usually make good choices unconsciously when determining an action to take from among several choices under various environmental conditions. We must make such choices on both large and small scales. For example, we must select a route to a restaurant from a variety of routes. After being seated, we have more details to decide. Consider a person reaching for a menu on the table. The goal of the movement is to move the arm toward the menu to grab it. We must select one desirable trajectory out of an infinite number of possible trajectories. Animals and humans are able to empirically learn how to select good actions from past experiences. We can also learn from demonstrations performed by other people (Atkeson & Schaal, 1997a,b; Miyamoto & Kawato, 1998; Miyamoto et al., 1996; Schaal, 1999), which is called 'learning by watching' or 'teaching by showing.' However, it is still difficult to start from scratch in a conventional learning machine. A reinforcement learning framework is fascinating and can obtain such autonomous behavior (Sutton & Barto, 1998 for a detailed and comprehensive description).

A conventional reinforcement learning framework is very powerful for small-scale problems such as a simple maze on a computer screen, but it cannot be easily applied to large-scale problems (e.g. driving a car in a real environment). For application to realistic and large-scale problems, the recent trend of research in reinforcement learning is to use modularization and hierarchical architecture. Several methods for constructing such higher levels have been studied, such as macro-actions, sub-goals, and multiple time scales (McGovern, Precup, Ravindran, Singh, & Sutton, 1998; McGovern & Sutton, 1998).

Since the feedback gain of biological systems is low and delay is large, high-speed movement must be controlled with feed-forward control. Recent results of a psychological experiment indicated that the sensory-motor system modifies the ongoing movement with a continually updated internal estimate of the environment (Todorov, 1998). In this paper, we propose a reinforcement learning framework

* Corresponding author. Address: Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Hibikino 2-4, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0196, Japan. Tel./fax: +81-93-695-6126.
E-mail address: miyamo@brain.kyutech.ac.jp (H. Miyamoto).

0893-6080/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2003.11.004

for motor control. With via-point representation, this framework acquires feed-forward control, on-line planning, and environmental adaptation abilities like those of biological systems. This framework contains two levels of hierarchical architecture. The higher level, which is implemented using via-point representation, corresponds to macro-actions or multiple time scales. The lower level, which is implemented using a trajectory generator, corresponds to primitive actions. In Section 2, we present our new method in detail. In Section 3, we describe computer simulation that uses our method.

2. Reinforcement learning framework with via-point representation

In this section, we begin with a brief review of the conventional temporal difference (TD) algorithm of reinforcement learning, before giving a detailed description of TD-learning with via-point representation.

2.1. Reinforcement learning

In the reinforcement learning algorithm, a reinforcement signal (reward or punishment) is used instead of the desired output. Let $r(t+1) = r(\mathbf{s}(t), \mathbf{u}(t))$ denote the immediate reward for the state $\mathbf{s}(t) \in \mathbf{R}^n$ and the action $\mathbf{u}(t) \in \mathbf{R}^m$ at time t. The objective of the reinforcement learning is to maximize the long-term total discounted reward, the so-called value function:

$$V(\mathbf{s}(t)) = r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \cdots, \quad (1)$$

where $0 \leq \gamma \leq 1$ is a discount factor. The value function estimates the value (goodness) of a given state $\mathbf{s}$ when the system follows a policy $\pi : \mathbf{s} \rightarrow \mathbf{u}(\mathbf{s})$. The learning system tries to find a better policy by using the value function.

Here, let $P(\mathbf{s}(t))$ denote the value function estimated by a function approximator. After each transition from state $\mathbf{s}(t)$ to state $\mathbf{s}(t+1)$, under action $\mathbf{u}(t)$ and with reward $r(t+1)$, the estimated value function should satisfy[1]

$$\begin{aligned} P(t) &= r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \cdots \\ &= r(t+1) + \gamma\{r(t+2) + \gamma r(t+3) + \cdots\} \\ &= r(t+1) + \gamma P(t+1). \end{aligned} \quad (2)$$

Then, the TD error calculated on each time step is

$$\hat{r}(t) = r(t) + \gamma P(t) - P(t-1). \quad (3)$$

In the actor–critic architecture shown in Fig. 1a, the critic estimates $P(t)$ and modifies it so that $\hat{r}(t) \rightarrow 0$. The actor takes stochastic actions, and when a large $\hat{r}(t)$ is observed, it increases the probability of taking the same action under the same state. A variety of control tasks such as cart-pole control and the stand-up task have been studied using conventional actor–critic architecture (Barto, Sutton, & Anderson, 1983; Doya, 1996, 1997, 2000; Morimoto & Doya, 1998).

[1] Eq. (2) is not always satisfied due to its stochastic nature, and what the TD-learning does is to make the equation hold in a probabilistic manner.

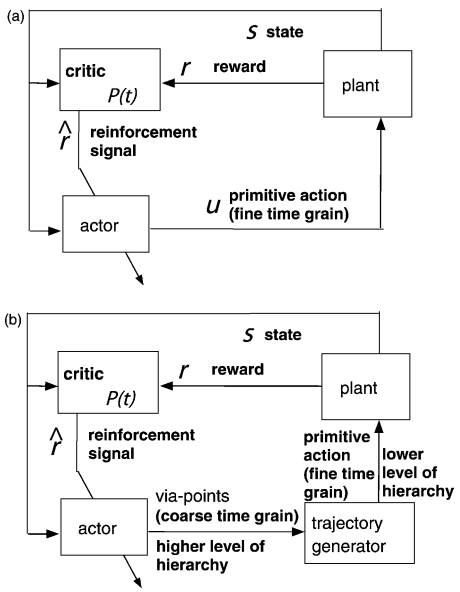

Fig. 1. (a) Actor–critic type reinforcement learning. Actor generates primitive action directly at fine time grain (every time bin). (b) Actor–critic type reinforcement learning with via-points. In the higher level of hierarchy, actor generates via-points at coarse time grain. In the lower level, trajectory generator transforms via-points to primitive action.

Fig. 1b shows our new actor–critic type reinforcement learning architecture with via-point representation. This architecture has two levels of hierarchy. The higher level is implemented using via-point representation, which works as macro-actions at multiple time scales. The lower level is implemented using a trajectory generator that generates primitive actions.

2.2. Incremental generation of via-points and trajectory

Fig. 2 illustrates how our model generates movement trajectory using via-point representation. In this method, via-points and the trajectory are generated on-line. In Fig. 2, the circles (○) denote the positions of via-points and the xs (×) denote the positions of tentative via-points. We used a minimum jerk trajectory (detailed description is in Appendix A) as the target trajectory.

In order to uniquely determine a minimum jerk trajectory from the current via-point χ_n at time t_n and the next via-point χ_{n+1} at time t_{n+1}, we have to specify the velocity and acceleration at time t_{n+1}. In our previous study of minimum jerk trajectory generation (Miyamoto et al., 1996), the velocity and acceleration of the via-point (end point) were set to 0. However, if the velocity and acceleration are set to 0 at the via-point at t_{n+1}, the movement will be awkward. To smoothly connect these via-points, we prepare a supplementary via-point χ'_{n+2} at t'_{n+2} in addition to the next via-point

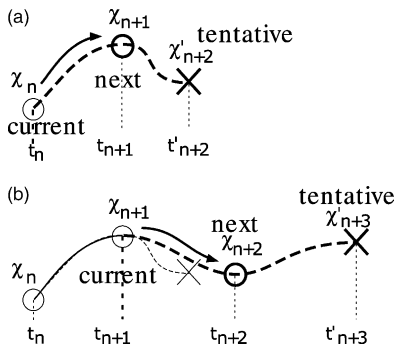

Fig. 2. Temporally localized via-points and optimal trajectory generation. Provisional trajectory (broken line) is generated by using a tentative via-point (×) in addition to the next via-point.

χ_{n+1} at t_{n+1}. The velocity and acceleration are unspecified at χ_{n+1} and set to 0 at the tentative via-point χ'_{n+2}. Then an optimal trajectory ($\chi_n \rightarrow \chi_{n+1} \rightarrow \chi'_{n+2}$) is uniquely determined. The first half of the trajectory ($\chi_n \rightarrow \chi_{n+1}$) is used for control. When the trajectory is followed until time t_{n+1} (Fig. 2b), the remaining trajectory ($\chi_{n+1} \rightarrow \chi'_{n+2}$) is discarded and new via-points χ_{n+2}, χ'_{n+3} are generated. The trajectory goes over the via-points just like stepping from stone to stone. In this way, the entire movement is executed with the iterative generation of the via-points and trajectories.

2.3. TD learning with via-points

Fig. 3 shows a block diagram of TD-learning with via-point representation. Let t_n denote the time at the nth via-point. At time t_n, the system receives the state vector $\mathbf{s}(t_n)$ and immediate reward $r(t_n)$. Then, the actors output the locations and the timing of the next ($(n+1)$th) and tentative ($(n+2)$th) via-points χ_{n+1}, t_{n+1}, χ'_{n+2}, t'_{n+2}. By using these via-points, as described in Section 2.2, the local trajectory

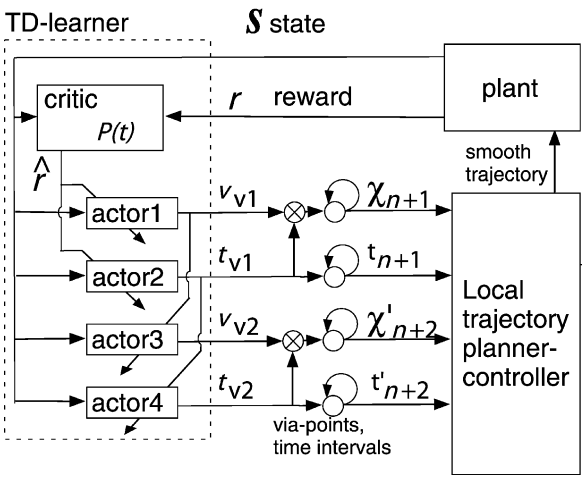

Fig. 3. TD-learning using via-points. Actors 1 and 2 create rate of change in position ($\boldsymbol{\nu}_{v1}$) and time interval (t_{v1}) of the next via-point. Actors 3 and 4 create those of the tentative via-point.

planner–controller generates a smooth trajectory that is fed to the plant.

2.3.1. Generating via-points

At time t_n, the locations and timing of the next and tentative via-points are specified as follows,

$$\chi_{n+1} = \chi_n + t_{v1}\boldsymbol{\nu}_{v1}, \tag{4}$$

$$t_{n+1} = t_n + t_{v1}, \tag{5}$$

$$\chi'_{n+2} = \chi_{n+1} + t_{v2}\boldsymbol{\nu}_{v2}, \tag{6}$$

$$t'_{n+2} = t_{n+1} + t_{v2}, \tag{7}$$

where $\boldsymbol{\nu}_{v1}$ and $\boldsymbol{\nu}_{v2}$ are the vectors denoting the rate of change in the via-point position during a unit time. t_{v1} and t_{v2} are scalar and denote the time intervals between via-points. These values are calculated by the actor using function approximators as

$$\boldsymbol{\nu}_{v1} = z_1(\mathbf{s}(t_n)) + \boldsymbol{\nu}_1(t_n), \tag{8}$$

$$t_{v1} = z_2(\mathbf{s}(t_n)) + \boldsymbol{\nu}_2(t_n), \tag{9}$$

$$\boldsymbol{\nu}_{v2} = z_3(\mathbf{s}(t_n)), \tag{10}$$

$$t_{v2} = z_4(\mathbf{s}(t_n)), \tag{11}$$

where $\boldsymbol{\nu}_1(t_n)$ and $\boldsymbol{\nu}_2(t_n)$ are perturbations for exploration. z_i is the output of the function approximator of the ith actor.

2.3.2. Learning of critic and actors

In this section, we first describe the ordinary implementation of actor and critic with via-point representation. The critic is estimated on each time step as described in Section 2.1. In the case of the actors, The TD error is calculated between time t_{n-1} and t_n at two successive via-points as follows.

$$\hat{r}(t_n) = \sum_{j=0}^{k} \gamma^j r(t+j) + \gamma^k P(\mathbf{s}(t+k)) - P(\mathbf{s}(t_{n-1}))), \tag{12}$$

where $k = t_n - t_{n-1}$. In this ordinary implementation, we have to observe reward and estimate critic at each time step.

In this paper, we adopted the following method. This method is very simple and needs less computational cost. The system observes reward and estimates critic at time t_n. Let $V(\mathbf{s}(t_n))$ denote the value function under the state $\mathbf{s}(t_n)$ and the reward $r(t_{n+1}) = r(\mathbf{s}(t_n))$.

$$V(\mathbf{s}(t_n)) = r(t_{n+1}) + \gamma_v r(t_{n+2}) + \gamma_v^2 r(t_{n+3}) + \cdots, \tag{13}$$

where $0 \le \gamma_v \le 1$ is a discount factor. Here, let $P(\mathbf{s}(t_n))$ denote the estimated value function calculated by the function approximator of the critic. The TD error is calculated as

$$\hat{r}(t_n) = r(t_n) + \gamma_v P(\mathbf{s}(t_n)) - P(\mathbf{s}(t_{n-1})). \tag{14}$$

The time intervals are not fixed in Eq. (13), although its form is similar to Eq. (1). It appears incorrect that Eqs. (13) and (14) do not include information of the time interval, but they actually work very well as shown in Section 3. Because

the information of the time interval is included in the action and the system can actively control the time interval as described in Section 2.3.1, Eqs. (13) and (14) need not include information of the time interval. Although γ_v is fixed in Eq. (14), γ_v operates differently from γ in Eq. (3). When the via-points are sparsely located in time, the large time interval $t_n - t_{n-1}$ corresponds to large γ in Eq. (3). In this case, Eq. (14) takes a long-term view of the reward accumulation. Conversely, when the via-points are densely located in time, the system tries to gain an immediate reward.

The function approximator of the critic estimates $P(\mathbf{s}(t_n))$ and modifies its weights so that $\hat{r}(t_n) \to 0$. When a positive $\hat{r}(t_n)$ is observed, the weights of the function approximator of actors 1 and 2 are modified so that they calculate the same outputs when the same state $\mathbf{s}$ appears. The weights of the function approximator of actors 3 and 4 are modified so that χ'_{n+2} and t'_{n+2} are equivalent to χ_{n+2} and t_{n+2} in a supervised fashion.

3. Computer simulation of cart-pole swing up task

We tested the framework described in section 2 with a challenging task, the cart-pole swing up task (Fig. 4) (Atkeson & Schaal, 1997a; Doya, 1997, 2000), which is a strongly nonlinear extension of the well known cart-pole balancing task (Barto et al., 1983).

3.1. Cart-pole swing up task

The pole has to be swung up from the down position ($\theta = -\pi$) and then balanced in the upright position ($\cos(\theta) = 1$).

We used the same dynamic equation as the pendulum used by Atkeson and Schaal (1997a), except for the parameters. The dynamic equation of the cart-pole is

$$\dot{\theta}_{i+1} = (1 - \alpha_1)\dot{\theta}_i + g\Delta t/l(\sin(\theta_i) + \ddot{x}_i\cos(\theta_i)/g),$$

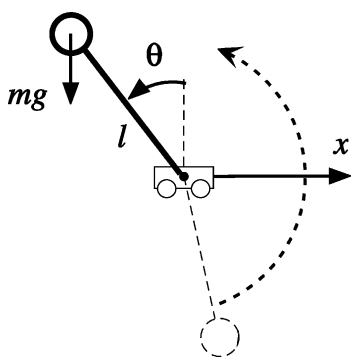

Fig. 4. Swing up task. The pole is initially hanging down ($\theta = -\pi$) from a cart. The goal of this task is to move the cart so that the pole becomes upright ($\theta = 0$ or $\theta = 2\pi$) and balanced. m, g, θ, and x denote mass of pole tip, gravity constant, angle of the pole, and horizontal position of the cart, respectively.

where θ_i, $\dot{\theta}_i$, $\ddot{\theta}_i$, $\ddot{x}_i$ are the angular position, angular velocity, angular acceleration and horizontal acceleration of the rotation axis at the ith sampling time, respectively. α_1 is the viscous term, $\Delta t = 1/60$ s is the sampling time, $g = 9.81 \times$ m/s^2 is the gravity acceleration, and $l = 0.35$ m is the length of the pole. We set $\alpha_1 = 0.01$ in this experiment.

We use cart position x as via-point location χ_n and acceleration $\ddot{x}_i$ as action $\mathbf{u}$, which can be easily obtained by differentiating the x trajectory twice. The state vector is $\mathbf{s}(t) = (\theta, \dot{\theta}, x, \dot{x}$ where x and $\dot{x}$ are the position and velocity of the cart, respectively. Each trial was started from an initial state $s(0) = (-\pi, 0, 0, 0)$. The reward was given by $r(t_n) = \cos(\theta(t_n))$. When the cart bumped into the end of the track, the terminal reward was $r(t) = -2$; otherwise, each trial lasted for 12 seconds.

3.1.1. Approximation function

As the approximation function of the critic and the actors, we used the Adaptive Gaussian Softmax Basis Function (AGSBF), which is the same as that used by Morimoto and Doya (1998). The output of the AGSBF is calculated as follows. For a given n-dimensional input vector $\mathbf{s}$, the activation function of the kth unit is calculated by

$$a_k(\mathbf{s}) = \exp(-\|M_k(\mathbf{s} - \mathbf{c}_k)\|^2/2), \tag{15}$$

where $\mathbf{c}_k$ is the center of activation and M_k is a matrix that determines the shape of the activation function. The Softmax Basis Function is then given by

$$b_k(\mathbf{s}) = a_k(\mathbf{s}) / \sum_{l=1}^{K} a_l(\mathbf{s}), \tag{16}$$

where K is the number of basis functions. The output of the AGSBF is given by the inner product of the basis functions and the weights w_k as follows

$$y(\mathbf{s}) = \sum_{k=1}^{K} w_k b_k(\mathbf{s}). \tag{17}$$

A new unit is allocated if the error is larger than criterion $e_{\max}$ and the activation of all existing units is smaller than threshold $a_{\min}$. The new unit is initialized with $c_k = \mathbf{s}$, $M_k = \mathrm{diag}(\mu_i)$, and $w_k = y_d(\mathbf{s})$, where μ_i is the inverse of the radius of the basis function.

3.1.2. Critic

The output of the critic is $P(\mathbf{s}(t_n)) = y(\mathbf{s}(t_n))$, which is given by Eq. (17). The critic's learning uses the TD(λ) algorithm (see Sutton & Barto, 1998 for a detailed and comprehensive description) as follows.

$$\Delta w_k = \eta_c \bar{r}(t_n) e_k(t_n), \tag{18}$$

$$\bar{r}(t_n) = \begin{cases} r_+ & \text{if } \hat{r}(t_n) \geq r_+ \\ r_- & \text{if } \hat{r}(t_n) \leq r_- \\ \hat{r}(t_n) & \text{otherwise} \end{cases}, \tag{19}$$

$$e_k(t_n) = \gamma_v \lambda e_k(t_{n-1}) + b_k(\mathbf{s}(t_n)), \tag{20}$$

where $e_k(t_n)$ denotes an eligibility trace of each weight. When the time interval is increased, $\hat{r}(t_n)$ sometimes becomes large. Because the large $\hat{r}(t_n)$ tends to destabilize the critic's learning, we limited $\hat{r}(t_n)$ to $\bar{r}(t_n)$. The parameters of the critic are $\gamma_v = 0.9$, $\lambda = 1$, $\mu_i = (1/(\pi/10), 1/(\pi/2), 1/0.5, 1/5)$, $a_{\min} = 0.6$, $e_{\max} = 0.01$, $\eta_c = 0.3$, $r_+ = (P_+ - P_-)/500$, and $r_- = -(P_+ - P_-)/500$, where P_- and P_+ are the minimal and maximal levels of the estimated value function.

3.1.3. Actors

The output of the actors are $z_i(\mathbf{s}(t_n)) = y_i(\mathbf{s}(t_n))$. We limit the output $z_1(\mathbf{s}(t_n)) = \nu_{v1}(t_n)$ to $[-1,1]$ and $z_2(\mathbf{s}(t_n)) = t_{v1}(t_n)$ to [0.05,1]. The size of the perturbation tapered off as the performance improved (Gullapalli, 1990). The perturbations ν_1, ν_2 are applied to actors 1 and 2 as follows.

$$|\nu_i(t_n)| < \nu_{0i} \min\left(1, \max\left(0, \frac{P_+ - P(\mathbf{s}(t_n))}{P_+ - P_-}\right)\right), \tag{21}$$

where $i = 1, 2$, and P_- and P_+ are the minimal and maximal levels of the estimated value function. The weights of actors 1 and 2 are updated by the LMS rule

$$\Delta w_k = \begin{cases} \eta_a \nu_i(t_n) b_k(\mathbf{s}(t_n)) & \text{if } \hat{r} \geq 0 \\ -\eta_a \nu_i(t_n) b_k(\mathbf{s}(t_n)) & \text{otherwise} \end{cases}, \tag{22}$$

where η_a is the learning rate.

The trajectory is smoothly connected at via-points, if the outputs of actors 3 and 4 at time $n-1$ are the same as the outputs of actors 1 and 2 at time n, respectively. The weights of actor 3 are updated as follows.

$$\Delta w_k = \eta_a b_k(\mathbf{s}(t_{n-1}))(z_1(\mathbf{s}(t_n)) - z_3(\mathbf{s}(t_{n-1}))). \tag{23}$$

The weights of actor 4 are updated as follows.

$$\Delta w_k = \eta_a b_k(\mathbf{s}(t_{n-1}))(z_2(\mathbf{s}(t_n)) - z_4(\mathbf{s}(t_{n-1}))). \tag{24}$$

The parameters of the actors are $\mu_i = (1/(\pi/10), 1/(\pi/2), 1/0.5, 1/0.5)$, $a_{\min} = 0.6$, $e_{\max} = 0.01$, and $\eta_a = 0.3$.

3.2. Results

Fig. 5 shows an example of the learned behavior of the system (the 6991st to the 7000th trials are superimposed). Fig. 5a and b show the time course of x and θ, respectively. In Fig. 5a, the circles (○) and xs (×) show the via-points and tentative via-points, respectively (the positions of ○ and × are almost the same). In the figure, the perturbation was not applied to the 7000th trial (shown as a thick line). The 6991st to 6999th trials are also shown but are indistinguishable because the lines overlap each other. After a preparatory swing to build up the inertial energy, the system succeeded in swinging up the pendulum and maintaining its balance. At the first period of the movement, the via-points are sparsely located in time. In this period, the system is controlled in a feed-forward manner. After the pole swings up, the via-points are densely located in time, since the system needs fine feedback to keep the balance.

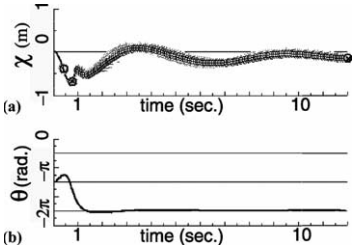

Fig. 5. An example of the behavior of the system. (a) Time course of the cart position (x). ◦ and × denote via-points and tentative via-points, respectively. First 1 s, two via-points are located sparsely in time, as with feed-forward control. To keep the pole upright, via-points are densely located in time after second one. (b) time course of the angle of the pole (θ).

The system generates such via-point locations in the following way. In the initial part of the movement, in climbing up the slope of the value function, the system can get a larger reward with a small number of steps by taking a long interval of via-points. When the system tries to keep the balance, the via-points have to be densely located in time. If the via-points are sparsely located, the state will quickly fall down from the peak of the value function.

Fig. 6 shows the relationship between the holding time at the inverted position and the trial number. As a measure of the swing up performance, we defined the time in which the pole stayed up ($\cos(\theta) > \cos(\pi/4)$) as t_{up}. Fig. 6a shows the performance averaged over 10 runs. Fig. 6b shows the individual performances of 10 runs. In Fig. 6b, each line was made by averaging every 100 trials over a single run. We can see that the system shows a good learning performance.

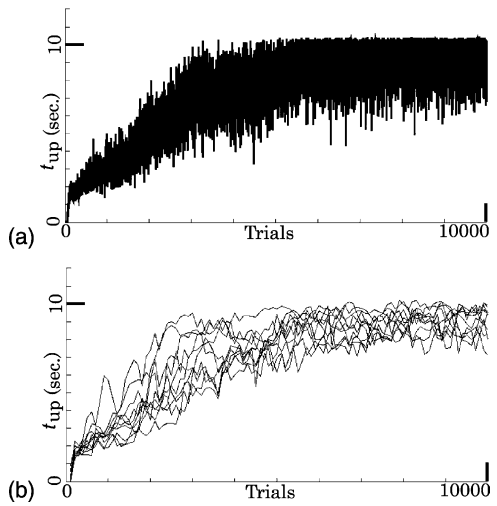

Fig. 6. Relationship between the keep time at the inverted position and the trial number. t_{up} denotes the time in which the pole stayed up ($\cos(\theta) > \cos(\pi/4)$). (a) performance averaged over 10 runs. (b) individual performances of 10 runs. Each line was made by averaging every 100 trials over a single run.

3.3. Comparison with conventional TD-learner

We compared the performance of the via-point framework (Fig. 3) with that of the conventional actor–critic framework (Fig. 1a, the actor output the desired velocity of the cart. Then the desired acceleration was calculated and fed to the cart). Fig. 7 shows the relationship between the holding time at the inverted position and the trial number of the conventional actor–critic framework. In Fig. 7, the lines show the performance averaged over 10 runs with the conventional actor–critic framework. The numerical labels '1', '5', '10', and '15' denote the actor and critic changes in output at every 1, 5, 10, and 15 sampling times, respectively, where the sampling time is $\Delta t = 1/60$ s.

In the conventional actor–critic framework illustrated in Fig. 1a, the critic and actor update their output at fixed time intervals. From Fig. 7, in the conventional actor–critic framework, good performances are obtained using intervals '5' and '10'. Intervals '1' and '15' do not show good performance in comparison with '5' and '10'. We have to choose the best control interval for the critic and actors to achieve good performance using the conventional TD-learner. On the other hand, in the via-point framework, the best control intervals are automatically obtained as a result of reinforcement learning. This is a good property for an environmental adaptation of biological systems.

In conventional actor–critic architecture, if the sampling interval is sufficiently small, the actor can output a smooth trajectory. In this case, the scale of the problem becomes too large to solve. As shown in experimental results, the system cannot find a way to swing up the pole when the interval is 1. If the time interval is large (interval 10), although the exploration is easy, the actor may output jerky motion. This is not bad for the swing up task itself, but it could induce harmful effects to mechanical systems.

Even if the smoothness is only local, it is very important for motor control. Smooth trajectory eliminates the harmful effects. This is a very good property for motor control systems and biological systems. We choose the minimum jerk trajectory as the smooth trajectory because it is well known as a model of human arm movement.

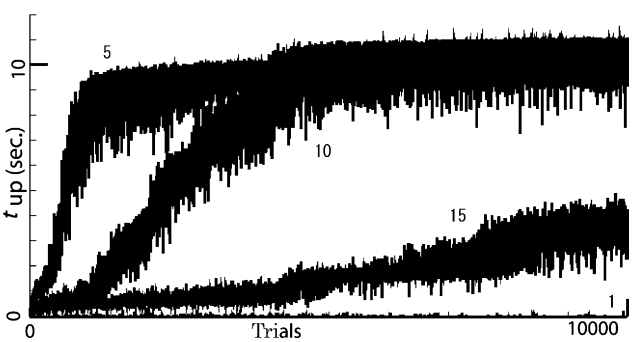

Fig. 7. Relationship between the keep time at the inverted position and the trial number with the conventional actor–critic framework. t_{up} denotes the time in which the pole stayed up ($\cos(\theta) > \cos(\pi/4)$). The numerical labels '1', '5', '10', and '15' denote time intervals.

4. Conclusions

In this paper, we demonstrated that a hierarchical reinforcement learning framework containing via-point representation is effective for learning motor control. Since the feedback gain of biological systems is low and delay is large, high-speed movement must be controlled with feed-forward control. In the framework proposed in this paper, feed-forward-like control was acquired by introducing the via-point representation. In the conventional TD-learner, we must choose the best control interval to achieve good performance. When a parameter of the environment (e.g. the mass or length of the cart) is changed, we must readjust the control interval. In the via-point framework, the best control intervals are automatically obtained accordingly, when a parameter of the environment is changed, the system is able to adapt to a new environment automatically.

According to the recent results of a psychological experiment, when we move the hand through a small set of sequential targets, the sensory-motor system modifies the ongoing movement with a continually updated internal estimate of the environment (Todorov, 1998). The on-line planning of via-points and trajectory in our model is in accordance with the results of this psychological experiment.

When humans make goal-directed arm movements, we show highly stereotypical trajectories, even though there are an infinite number of possible trajectories. Based on optimization principles for trajectory planning, several models have been proposed, such as the minimum-jerk (Flash & Hogan, 1985), the minimum-torque-change (Uno et al., 1989a), the minimum-muscle-tension-change (Uno et al., 1989b), and the minimum-motor command-change models (Kawato, 1992). Recently, Harris & Wolpert (1998) have proposed a minimum variance model to generate an optimal trajectory. They present a unifying theory of eye and arm movements based on the single physiological assumption that the neural control signals are corrupted by noise whose variance increases with size of the motor command amplitude. In the presence of such signal-dependent noise, the shape of a trajectory is selected to minimize the variance of the final eye or arm position. For the nonlinear two-jointed planar arm, they derived optimal trajectories by parameterizing them by cubic splines and by adjusting the knot locations using the simplex algorithm. However, when we consider a biologically plausible way of obtaining the minimum-variance trajectory, the reinforcement learning framework seems to be suitable. Although we used the minimum-jerk trajectory as an optimal trajectory in this paper, we will integrate such a trajectory planning mechanism into our model in the near future.

Acknowledgements

We would like to thank the members of the Kawato Dynamic Brain Project and the members of ATR

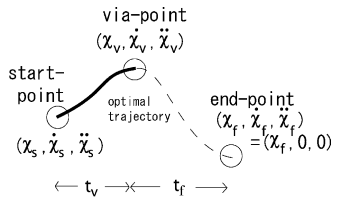

Fig. A1. Optimal trajectory (solid line) and the start, via- and end points (◦)

Computational Neuroscience Laboratories for their useful discussions.

Appendix A. Minimum jerk trajectory

Fig. A1 shows an optimal trajectory passing through the start, via-, and end points. Let ($\mathbf{x}_s$, $\dot{\mathbf{x}}_s$, $\ddot{\mathbf{x}}_s$ be position, velocity, and acceleration at the start point where $\mathbf{x} \in \mathbf{R}^n$. Similarly, we denote ($\mathbf{x}_v, \dot{\mathbf{x}}_v, \ddot{\mathbf{x}}_v$ and ($\mathbf{x}_f$, $\dot{\mathbf{x}}_f$, $\ddot{\mathbf{x}}_f$ as via- and end points. t_v and t_f, respectively, are the time intervals between the start and via-points and between the via- and end points. Here, we use the minimum jerk trajectory (fifth-order polynomial) as the optimal trajectory. The minimum-jerk trajectory can be obtained by using a recurrent neural network when the velocity and acceleration at the end point are equal to 0 (Hoff & Arbib, 1993). We can extend their model to generate the minimum-jerk trajectory under general conditions so that the velocity and acceleration at the end points are not equal to 0 (Wada & Kawato, 1993). If the position, velocity, and acceleration at these points are all fixed, then we can calculate the minimum jerk trajectory. Even if the velocity and acceleration at a via-point are not fixed, we can calculate $\dot{\mathbf{x}}_v$ and $\ddot{\mathbf{x}}_v$ Consequently, we can calculate the minimum jerk trajectory passing through the start, via- and end point, if the following variables are given: ($\mathbf{x}_s$, $\dot{\mathbf{x}}_s$, $\ddot{\mathbf{x}}_s$), $\mathbf{x}_v$, ($\mathbf{x}_f$, $\dot{\mathbf{x}}_f, \ddot{\mathbf{x}}_f$), t_v, t_f. In this paper, we set the velocity and acceleration at the end point to 0 for simplicity. As described in Section 2.2, the first half of the trajectory is used for control.

References

Atkeson, C. G., & Schaal, S. (1997a). Learning tasks from a single demonstration. *In IEEE International Conference on Robotics and Automation*, *2*, 1706–1712.

Atkeson, C. G., & Schaal, S (1997) Robot learning from demonstration. In *International Conference on Machine Learning (ICML97)*.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *In IEEE Transactions on Systems, Man, and Cybernetics*, *3*, 834–846.

Doya, K. (1996). Temporal difference learning in continuous time and space. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), (*8*) (pp. 1073–1079). *Advances in neural information processing systems*, Cambridge, MA: MIT Press.

Doya, K. (1997). Efficient nonlinear control with actor-tutor architecture. In M. C. Mozer, & M. I. Jordan (Eds.), (*9*) (pp. 1012–1018). *Advances in neural information processing systems*, Cambridge, MA: MIT Press.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, *12*, 243–269.

Flash, T., & Hogan, N. (1985). The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience*, *5*, 1688–1703.

Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, *3*, 671–692.

Harris, C. M., & Wolpert, D. M. (1998). Signal-dependent noise determines motor planning. *Nature*, *394*(20), 780–784.

Hoff, B., & Arbib, M. A. (1993). Models of trajectory formation and temporal interaction of reach and grasp. *Journal of Motor Behavior*, *25*(3), 175–192.

Kawato, M. (1992). Optimization and learning in neural networks for formation and control of coordinated movement. In D. Meyer, & S. Kornblum (Eds.), *Attention and performance, XIV: synergies in experimental psychology, artificial intelligence, and cognitive neuroscience—A silver jubilee* (pp. 821–849). Cambridge, MA: MIT Press.

McGovern, A., Sutton, R.S (1998) Macro-actions in reinforcement learning: An empirical analysis. Technical Report 98-70, University of Massachusetts, Department of Computer Science

McGovern, A., Precup, A. D., Ravindran, B., Singh, S., & Sutton, R. S. (1998). Hierarchical optimal control of MDPs. *In Proceedings of the Tenth Yale Workshop on Adaptive and Learning Systems*, pp. 186–191.

Miyamoto, H., & Kawato, M. (1998). A tennis serve and upswing learning robot based on dynamic optimization theory. *Neural Networks*, *11*(7-8), 1331–1344.

Miyamoto, H., Schaal, S., Gandolfo, F., Gomi, H., Koike, Y., Osu, R., Nakano, E., Wada, Y., & Kawato, M. (1996). A Kendama learning robot based on dynamic optimization theory. *Neural Networks*, *9*(8), 1281–1302.

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

Schaal, S. (1999). Is imitation learning the way to humanoid robots? *Trends in Cognitive Sciences*, *3*(6), 233–242.

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

Todorov, E. V (1998) *Studies of goal directed movements*. PhD thesis. Massachusetts Institute of Technology.

Uno, Y., Kawato, M., & Suzuki, R. (1989a). Formation and control of optimal trajectory in human multijoint arm movement—minimum torque-change model. *Biological Cybernetics*, *61*, 89–101.

Uno, Y., Suzuki, R., & Kawato, M. (1989b). Minimum muscle-tension-change model which reproduces human arm movement. *Proceedings of the 4th Symposium on Biological and Physiological Engineering* (pp. 299–302), in Japanese.

Wada, Y., & Kawato, M. (1993). A neural network model for arm trajectory formation using forward and inverse dynamics models. *Neural Networks*, *6*(7), 919–932.