Int. J. Appl. Math. Comput. Sci., 2001, Vol.11, No.1, 77–104

MOTOR CONTROL: NEURAL MODELS AND SYSTEMS THEORY

Kenji DOYA*, Hidenori KIMURA** Aiko MIYAMURA**

In this paper, we introduce several system theoretic problems brought forward by recent studies on neural models of motor control. We focus our attention on three topics: (i) the cerebellum and adaptive control, (ii) reinforcement learning and the basal ganglia, and (iii) modular control with multiple models. We discuss these subjects from both neuroscience and systems theory viewpoints with the aim of promoting interplay between the two research communities.

Keywords: inverse model, adaptive control, cerebellum, reinforcement learning, basal ganglia, multiple models

1. Introduction

Neuroscience and systems theory play complementary roles in understanding the mechanisms of adaptive systems. Neuroscientists are faced with complex, high-performance adaptive systems and try to understand why they work so nicely. Systems theorists tend to start from simple, idealized systems but try to prove rigorously how they perform under well-defined conditions. In this paper, we introduce recent examples of converging efforts from both sides towards understanding and building adaptive autonomous systems, and aim to promote future collaboration between the neuroscience and systems theory communities.

The biological motor system is in some sense an ideal realization of control. It consists of actuators, sensors and controllers, like usual control systems do. Unlike artificial control systems, however, it exhibits much higher performance with great flexibility and versatility in spite of the nonlinearity, uncertainties and large degrees of freedom of animal bodies. Actually, motor control has been a main focus of neuroscience research for a long time, but what is strange is that control theorists have rarely tried to seriously investigate a theoretical basis of biological motor control. Fortunately, recent progress in computational neuroscience involving motor control has brought forward several system theoretic issues in much clearer ways and with deeper insight than before.

* Information Sciences Division, ATR International; CREST, Japan Science and Technology Corporation, 2-2-2 Hikaridai, Seika, Soraku, Kyoto 619–0288, Japan, e-mail: doya@isd.atr.co.jp

** Graduate School of Frontier Science, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113–8656, Japan, e-mail: {kimura,aiko}@crux.t.u-tokyo.ac.jp

We focus our attention on the following three topics:

1. The cerebellum and adaptive control,
2. Reinforcement learning and the basal ganglia, and
3. Modular control using multiple models.

These three topics are related to the functions of the cerebellum, the basal ganglia, and the cerebral cortex, respectively, which are the major components of the mammalian brain (cf. Fig. 1) (Doya, 1999).

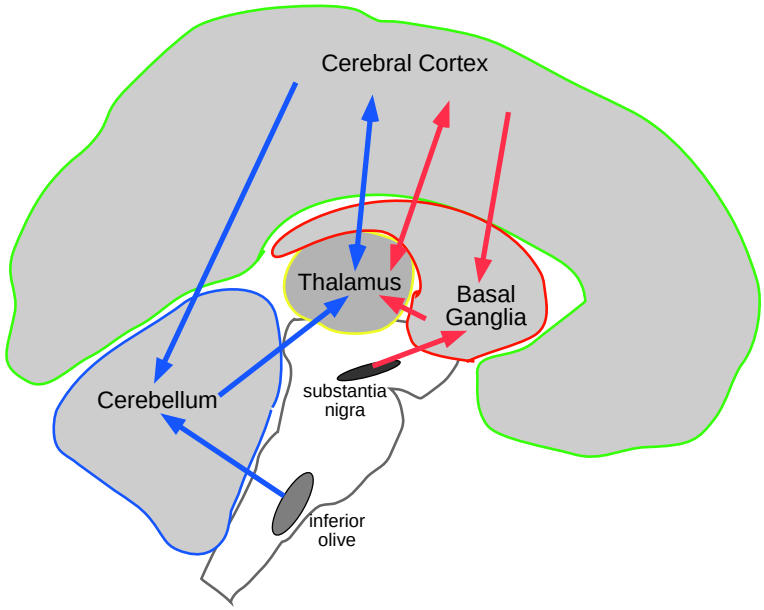


Fig. 1. The global circuit linking the cerebellum, the basal ganglia and the cerebral cortex.

For the sake of the first topic, we review the basic anatomy and physiology of the cerebellum and some neural modeling studies on how the cerebellum acquires internal models of the body and the environment. We then theoretically analyze a biologically motivated learning model of the cerebellum (Kawato *et al.*, 1987; Kawato and Gomi, 1992) and prove that it is a robust, versatile adaptive control architecture after some improvements.

For the sake of the second topic, we introduce a neural network study on "reinforcement learning" and show its relationship with the classical theory of dynamic programming. There was recently been some excitement in the neuroscience community concerning the idea that the activity of neurons that release a neurotransmitter dopamine is quite similar to the learning signal used in reinforcement learning algorithms. This has provided a strong clue as to the function of the basal ganglia, which receives strong dopaminergic projections.

For the sake of the third topic, we first review recent advances in multiple model-based control methods that ensure stability under system uncertainty. Incidentally, recent hot topics in computational neuroscience research include the switching and combination of multiple internal models. For instance, a biologically motivated learning architecture has been shown to be able to deal with both nonstationary and nonlinear control tasks quite nicely and it awaits further theoretical analyses.

From these examples, we wish to illustrate that neurobiological models of motor control are valuable candidates for theoretical analyses and real-world applications, and that systems theory can provide valuable insights into the functions of the nervous system.

2. The Cerebellum and Adaptive Control

2.1. Biological Background

The cerebellum is known to be involved in the control of quick limb movements (Ghez and Thach, 2000). For example, patients with cerebellar damage are strongly impaired in the control of multi-joint movements such as arm reaching and locomotion, although it is possible for them to execute simple, single-joint movements. It has also been demonstrated in experimental animals that the adaptation of quick eye movements is dependent on the cerebellum. Accordingly, it has been hypothesized that the major role of the cerebellum is both the temporal and spatial coordination of movements.

The cerebellar circuit has a roughly feedforward organization with two major input pathways: mossy fiber inputs and the climbing fiber inputs (Fig. 2). Mossy fibers send various sensory signals as well as motor commands to a massive number of granule cells, which in turn send parallel fibers to the Purkinje cells. Each Purkinje cell receives many thousands of parallel fiber inputs and only one climbing fiber input.

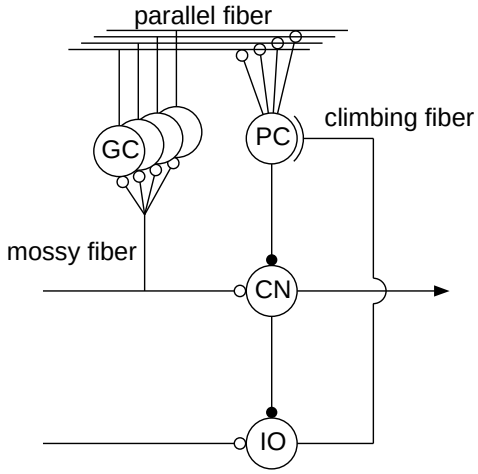


Fig. 2. The circuit of the cerebellum; GC: granule cells, PC: Purkinje cells, CN: cerebellar nucleus, IO: inferior olive, ○: excitatory synapse, ●: inhibitory synapse.

This peculiar organization had led Marr and Albus to propose a classical model of the cerebellum as a pattern recognition machine that recognizes various contexts for movement execution (Albus, 1971; Marr, 1969). In their model, granule cells work as combinatorial encoders of sensory and motor variables, and Purkinje cells work as pattern classifiers by using climbing fiber inputs as teacher signals.

In accordance with the Marr-Albus model, it was shown by Ito that the synaptic strengths of parallel fiber inputs to Purkinje cells are modified when the parallel fiber inputs are associated with climbing fiber inputs, which is known as long-term depression (LTD, cf. Ito *et al.*, 1982). Ito hypothesized that the cerebellum provides a model of the body and the physical environment so that accurate movement control is made possible despite time delays in sensory feedback (Ito, 1993).

A growing premise in control engineering at that time was that the design of a high-level control system had to be based on a model of a controlled object (plant). Ito's proposition was the first to recognize the important role of models in motor control, and was remarkably consistent with the model-based paradigm of control system design at its early stage. In control system design, plant models can be obtained through experiments and/or *a priori* physical knowledge. They can be represented in mathematical forms. In neural motor control, the situation is totally different. The brain must construct models of an object and the environment only through learning by experience and memorize them in its neural network in a format usable for motor control. Lately, the problem of how to acquire and store models of controlled objects in the neural network has come to be a central issue. Below we introduce an interesting architecture proposed by Kawato (Kawato *et al.*, 1987; Kawato and Gomi, 1992), which extends Ito's hypothesis.

2.2. Feedback-Error Learning Model of the Cerebellum

In the early 80's, the "computed torque" method was proposed for the control of robotic manipulators. The challenge here was that the multi-link arm dynamics had strong nonlinearity due to centrifugal and Coriolis forces, and that conventional linear servo control was insufficient for fast movements. In the computed torque method, the necessary torque for each motor is pre-calculated from the desired movement trajectory using a nonlinear inverse dynamics model of the arm.

With the knowledge of such a trend in robotic control, Kawato extended Ito's framework and proposed the feedback-error learning architecture as a model of adaptive control by the cerebellum (Kawato *et al.*, 1987). Figure 3 depicts Kawato's scheme in a block diagram form. Here, P denotes the plant, i.e., the neuromuscular system of limbs. The control architecture has two main components: a fixed linear feedback controller K_{fb} and an adaptive nonlinear feedforward controller K_{ff}. The feedback loop corresponds to the spinal and the cerebral feedback pathways which are innately programmed. The performance of such a feedback servo is limited due to the nonlinearity of the system and the delays in the sensory feedback. The nonlinear feedforward controller is assumed to take the role of the inverse model of the plant (e.g., the arm) such that it calculates the control command (e.g., the torques) τ^* necessary to achieve the desired movement trajectory (θ^*).

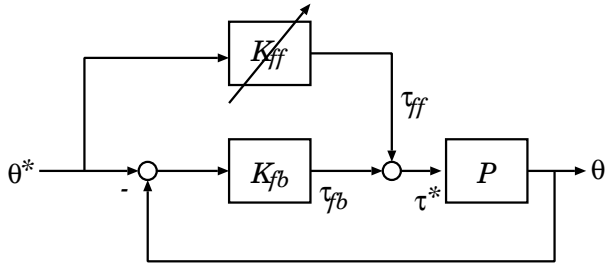


Fig. 3. Kawato's scheme of motor control.

The question in the above case was how to construct such an inverse model by learning, or more specifically, how to derive a teaching signal for the feedforward controller. Kawato's answer was to use the output of the linear feedback controller τ_{fb} as the error signal for the nonlinear, feedforward controller. It may be intriguing why the output of a "stupid" controller can be used for the training of a more clever controller. However, in both computer simulations and experiments with a PUMA robot arm, Kawato showed that the nonlinear controller can actually learn an inverse model by gradually replacing the output of the linear controller.

More specifically, the feedforward controller is characterized by the following nonlinear dynamics:

$$\tau_{ff} = \varphi\big(w, \theta, \dot{\theta}, \ddot{\theta}\big), \tag{1}$$

where w is a vector of unknown parameters to be adaptively tuned in real time. If the parameters w and the function φ are properly chosen, relation (1) generates the desired torque $\tau_{ff} = \tau^*$, which is required for the desired trajectory θ^* if we substitute θ^* for θ in (1). In this sense, the feedforward controller plays the role of the inverse dynamics of P, and adaptive tuning of the feedforward controller can be regarded as the modeling of the inverse dynamics P^{-1}. The most salient feature of Kawato's scheme of adaptive control is its adaptation law described by

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \alpha \left(\frac{\partial \tau_{ff}}{\partial w} \right)^T \tau_{fb}, \tag{2}$$

where $\alpha > 0$ is a parameter characterizing the adaptation speed. This algorithm is derived by a gradient method that minimizes the squared norm of the error

$$E = \frac{1}{2} \|\tau^* - \tau_{ff}\|^2 \tag{3}$$

under the assumption that

$$\tau^* = \tau_{ff} + \tau_{fb}. \tag{4}$$

Indeed, the gradient method for tuning the parameters w is given by

$$\frac{\mathrm{d}w}{\mathrm{d}t} = -\alpha \frac{\partial E}{\partial w},$$

which results in (2) under approximation (4). Approximation (4) implies that the input to the plant is always approximately correct. Justification of this approximation is one of the issues of the paper.

2.3. Control Theoretic Considerations

2.3.1. Feedback Error Learning Scheme as a Control Architecture

Now, we consider Kawato's feedback error learning scheme from the viewpoint of control system design in the framework of control theory. Actually, understood as a control architecture, the scheme is not new. Rather, it is a typical two-degree-of-freedom control scheme that has been known for years. The basic scheme is given in Fig. 4, where K_1 denotes a feedback controller that is responsible for the feedback properties of the control system (e.g., robustness against parameter variations, disturbance rejection, etc.), while K_2 denotes a feedforward controller that is responsible for the filtering properties (e.g., response speed, smooth tracking, etc.).

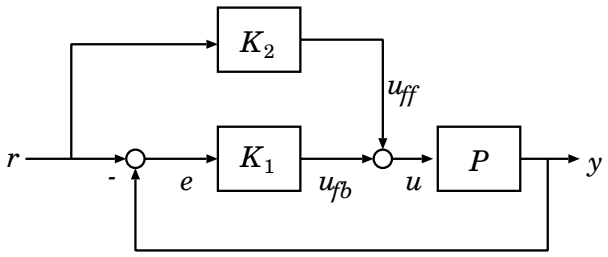


Fig. 4. Kawato's scheme in ideal situations.

In the ideal situation where the inverse of P (i.e., P^{-1}) exists and is stable, we can achieve the perfect tracking

$$e(t) = r(t) - y(t) = 0, \tag{5}$$

by choosing

$$K_2 = P^{-1} \tag{6}$$

for any stabilizing feedback controller K_1. This is the case when all of the zeros of P are stable. Unfortunately, almost all of the practically important plants do not satisfy this condition.

There are two non-ideal cases that depend on the locations of unstable zeros. The first case is where all unstable zeros are infinite. In other words, all of the finite zeros of P are stable. In that case, we can choose K_2 as

$$K_2 = P^{-1} W, \tag{7}$$

where W is a stable filter that makes K_2 stable and proper. Obviously, any stable W whose relative degree is equal to that of P is acceptable. Figure 5 represents a scheme where the tracking error $e(t)$ is given by

$$e(t) = W(s)r(t) - y(t), \tag{8}$$

rather than (5). It is clear that the perfect tracking is achieved with respect to the tracking error (8).

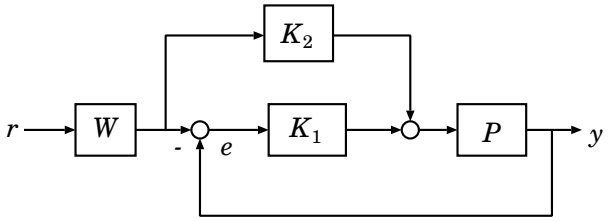


Fig. 5. A modification of Fig. 4.

Finally, we consider the case where there are finite unstable zeros. In this case, the feedforward controller (6) or (7) does not work because it must have some unstable poles, which will be canceled out by unstable zeros of the plant. In this case, we may choose K_2 as

$$K_2' = \arg \inf_{K:\text{ stable}} \|(I - PK)W\|_\infty. \tag{9}$$

This is actually the approximate inverse proposed by Zames (1981). The corresponding architecture is shown in Fig. 6. The feedforward controller K_2' given by (9) minimizes the normalized $\mathcal{L}_2$ error

$$J = \frac{\|Wr - y\|_2}{\|r\|_2}.$$

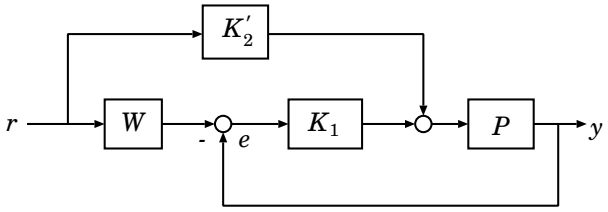


Fig. 6. A modification of Fig. 5.

2.3.2. Adaptation without Delay

Now we consider Kawato's feedback error learning method in the framework of adaptive control theory. In order to focus on the most essential features of the scheme,

we assume that the plant has a stable inverse. This is the simplest case which we discussed in the previous subsection. In this case, the ideal feedforward controller is given by (6). Hence, we must devise a tuning rule for K_2 that assures the convergence of K_2 to P^{-1}.

First, we parameterize the feedforward controller K_2 in the state space as

$$\dot{\eta}_1 = F\eta_1 + gr, \tag{10}$$

$$\dot{\eta}_2 = F\eta_2 + gu_0, \tag{11}$$

$$u_0 = c_0^T\eta_1 + d_0^T\eta_2 + k_0 r, \tag{12}$$

where $\eta_1 \in \mathbb{R}^n$ and $\eta_2 \in \mathbb{R}^n$ are the state vectors of K_2, and n denotes the McMillan degree of $P(s)$. F is any stable matrix with (F, g) being stable. The transfer function from r to u_0 is given by

$$T_{u_0 r}(s) = \frac{k_0 + c_0^T(sI - F)^{-1}g}{1 - d_0^T(sI - F)^{-1}g}. \tag{13}$$

If (F, g) is controllable, $T_{u_0 r}(s)$ can be equal to any transfer function of degree n by choosing vectors c_0 and d_0, and a scalar k_0 approximately. The parameterization (9) is used frequently in the literature on adaptive control (Åström and Wittenmark, 1989). Thus it is possible to set c_0, d_0 and k_0 so that K_2 is equal to P^{-1}. In adaptive control, however, the parameters c_0, d_0 and k_0 are unknown and we must construct a tuning rule for these parameters to produce a desired transfer function with input r and output u. More precisely, the parameterization (10)–(12) is rewritten as

$$\dot{\xi}_1 = F\xi_1 + gr, \tag{14}$$

$$\dot{\xi}_2 = F\xi_2 + gu, \tag{15}$$

$$u_{ff} = c(t)^T\xi_1 + d(t)^T\xi_2 + k(t)r, \tag{16}$$

and the unknown parameters $c(t), d(t)$ and $k(t)$ are tuned such that the error

$$e(t) = u_0(t) - u_{ff}(t)$$

tends to zero. A popular tuning rule is obtained by minimizing the squared error

$$J(t) = e(t)^2 = \big(u_0(t) - u_{ff}(t)\big)^2$$

using the gradient method. This results in the tuning rule

$$\dot{c}(t) = \alpha e(t)\xi_1(t), \tag{17}$$

$$\dot{d}(t) = \alpha e(t)\xi_2(t), \tag{18}$$

$$\dot{k}(t) = \alpha e(t)r(t), \tag{19}$$

where α is a parameter for adjusting the adaptation speed. The tuning rule (17)–(19) is essentially equivalent to (2) if we replace τ by u.

The convergence proof of the adaptation law (14)–(19) is quite standard. Set

$$V(t) = \|c_0 - c(t)\|^2 + \|d_0 - d(t)\|^2 + \|k_0 - k(t)\|^2,$$

where c_0, d_0 and k_0 are correct parameters that generate the desired I/O relation given in (9).

Then, from (17)–(19), it follows that

$$\begin{aligned}\dot{V}(t) &= -\alpha\Big[(c_0 - c(t))^T\xi_1(t) + (d_0 - d(t))^T\xi_2(t) + (k_0 - k(t))r(t)\Big]e(t)\\ &= -\alpha(c_0^T\xi_1(t) + d_0^T\xi_2(t) + k_0 r(t) - u_{ff}(t))e(t)\\ &= -\alpha e(t)^2 + \alpha\left[\ c_0^T(\eta_1(t) - \xi_1(t)) + d_0^T(\eta_2(t) - \xi_2(t))\ \right].\end{aligned}$$

Since F is stable, $\eta_1(t) \to \xi_1(t)$ and $\eta_2(t) \to \xi_2(t)$. Therefore, we have established $e(t) \to 0$.

The above procedure cannot be applied to our purpose of identifying $P^{-1}(s)$ because we do not know the desired output $n_0(t) = P^{-1}r(t)$. To circumvent this fundamental difficulty, Kawato used the feedback signal u_{fb} as an error signal that drives learning. From Fig. 4, it follows that

$$u_{fb}(t) = u(t) - u_{ff}(t),$$

where $u(t)$ is the input to the plant. If $u(t)$ is identical to the desired output of the feedforward controller K_2, we can use the above identification scheme for identifying $P^{-1}(s)$. However, $u(t)$ is not equal to the desired output $P^{-1}(s)r(t)$. This is the difficulty in Kawato's scheme.

Now, we write the adaptation law of Kawato's scheme (2) by reading τ as u. The parameterization (14)–(16) is written as

$$\dot{\xi}_1 = F\xi_1 + gr(t), \tag{20}$$

$$\dot{\xi}_2 = F\xi_2 + g\big(u_0(t) - P^{-1}(s)e(t)\big), \tag{21}$$

$$u_{ff}(t) = c(t)^T\xi_1 + d(t)^T\xi_2 + k(t)r(t), \tag{22}$$

where $u_0(t)$ denotes the true input

$$u_0(t) = P^{-1}(s)r(t) \tag{23}$$

and $e(t)$ denotes the tracking error

$$e(t) = r(t) - P(s)u(t). \tag{24}$$

Taking the feedback controller K_1 as a constant gain, we have

$$u_{fb}(t) = K_1 e(t). \tag{25}$$

We denote by $w(t)$ the unknown vectors $c(t)$, $d(t)$ and $k(t)$, i.e.,

$$w(t) = \big(c(t)^T\ d(t)^T\ k(t)\big)^T.$$

Then, we have

$$\frac{\partial u_{ff}(t)}{\partial w} = \begin{bmatrix} \xi_1(t)^T & \xi_2(t)^T & r(t) \end{bmatrix}^T.$$

Accordingly, we have the following representation of (2):

$$\dot{c}(t) = \alpha \xi_1(t) K_1 e(t), \tag{26}$$

$$\dot{d}(t) = \alpha \xi_2(t) K_1 e(t), \tag{27}$$

$$\dot{k}(t) = \alpha r(t) K_1 e(t). \tag{28}$$

The essential feature of Kawato's scheme is the existence of the inverse in (21), which reflects the fact that we do not know the desired output $u_0(t)$ of the feedforward controller K_2. Actually, it represents the coupling of feedforward learning and feedback control.

Since $u(t) = u_{ff}(t) + K_1 e(t)$ from Fig. 4, we have

$$u_0(t) - u_{ff}(t) = (P^{-1}(s) + K_1)e(t).$$

The correct values c_0, d_0 and k_0 of the parameters $c(t)$, $d(t)$ and $k(t)$, respectively, generate the following system:

$$\dot{\eta}_1 = F\eta_1 + gr(t), \tag{29}$$

$$\dot{\eta}_2 = F\eta_2 + gu_0(t), \tag{30}$$

$$u_0 = c_0^T \eta_1 + d_0^T \eta_2 + k_0 r(t). \tag{31}$$

From (20) and (29), it follows that

$$u_0(t) - u_{ff}(t) = \Delta c(t)^T \xi_1(t) + \Delta d(t)^T \xi_2(t) + \Delta k(t) r(t) + d_0^T \big(\eta_2(t) - \xi_2(t)\big),$$

where

$$\Delta c(t) = c_0 - c(t), \tag{32}$$

$$\Delta d(t) = d_0 - d(t), \tag{33}$$

$$\Delta k(t) = k_0 - k(t). \tag{34}$$

Due to (21) and (30), we have

$$\eta_2(t) - \xi_2(t) = -(sI - F)^{-1} g P^{-1}(s) e(t).$$

Accordingly, we have obtained the representation of the tracking error

$$e(t) = G(s)^{-1}\big(\Delta c(t)\xi_1(t) + \Delta d(t)\xi_2(t) + \Delta k(t) r(t)\big), \tag{35}$$

where

$$G(s) := \big(1 - d_0^T (sI - F)^{-1} g\big) P^{-1}(s) + K_1. \tag{36}$$

The vectors c_0, d_0 and k_0 are chosen such that (13) holds for $T_{ur}(s) = P^{-1}(s)$. Therefore, we have

$$G(s) = k_0 + c_0^T(sI - F)^{-1}g + K_1. \tag{37}$$

The following result is shown in (Miyamura and Kimura, 2000).

Theorem 1. *If $G(s)$ given by (37) is strictly positive real, then the algorithm implies $e(t) \to 0$ as $t \to \infty$.*

Now, since we assume that $P^{-1}(s)$ is stable, $k_0 + c_0^T(sI - F)^{-1}g$, which represents the "numerator" of $P^{-1}(s)$, is also stably invertible. Hence, we can always make $G(s)$ strictly positive real by choosing a large K_1. Accordingly, we have verified that Kawato's scheme is always stable if K_1 is sufficiently large, provided that $P(s)$ has a stable inverse.

2.3.3. The Case with a Time Delay

The existence of significant time delays in the sensory feedback pathway was an important motivation for introducing a feedforward controller in Kawato's scheme. Now, we consider the feasibility of Kawato's scheme for the case where there is a significant time delay in the feedback loop. Figure 7 illustrates a block diagram of Kawato's scheme, where τ is the delay time. The artificial time delay is introduced after the command signal $r(t)$ to make the tracking error consistent.

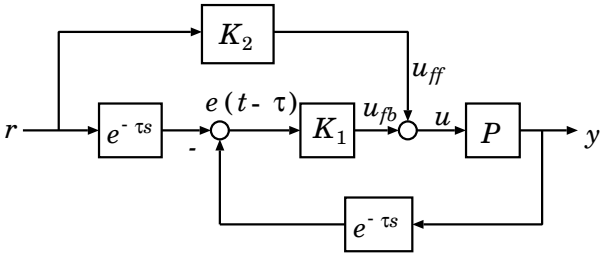


Fig. 7. Kawato's scheme with time delay.

Now, the parameterization and tuning rule are respectively described as

$$\dot{\xi}_1 = F\xi_1 + gr, \tag{38}$$

$$\dot{\xi}_2 = F\xi_2 + gu, \tag{39}$$

$$u_{ff}(t) = c(t)^T\xi_1(t) + d(t)^T\xi_2(t) + k(t)r, \tag{40}$$

$$u(t) = u_{ff}(t) + K_1e(t - \tau), \tag{41}$$

and

$$\dot{c}(t) = \alpha\xi_1(t-\tau)e(t-\tau), \tag{42}$$

$$\dot{d}(t) = \alpha\xi_2(t-\tau)e(t-\tau), \tag{43}$$

$$\dot{k}(t) = \alpha r(t-\tau)e(t-\tau). \tag{44}$$

From (35) and (38), it follows that

$$e(t) = G_\tau(s)^{-1}\big(\Delta c(t)^T\xi_1 + \Delta d(t)^T\xi_2 + \Delta k(t)r\big), \tag{45}$$

where $\Delta c(t)$, $\Delta d(t)$ and $\Delta k(t)$ are given by (36) and

$$G_\tau(s) = k_0 + c_0^\tau(sI - F)^{-1}g + K_1e^{-\tau s}. \tag{46}$$

The relation (45) corresponds to (37) for the delay-free case. The convergence of the algorithm is guaranteed only locally for the case with a delay.

Theorem 2. *If $G_\tau(s)$ is strictly positive real, then it converges for a sufficiently small α, if the initial error $e(0)$ is sufficiently small.*

Since $G_\tau(s)$ contains the delay $e^{-\tau s}$ as a coefficient of K_1, the condition that $G_\tau(s)$ is strictly positive real cannot be met for large K_1. In that case, we may choose F and g so that $G_\tau(s)$ is positive real. However, since k_0 and c_0 are unknown, it may be difficult to select such F and g.

3. Reinforcement Learning and the Basal Ganglia

3.1. Reinforcement Learning

Neural network models of reward- and penalty-based learning have been formulated under the name of "reinforcement learning" (RL) (Sutton and Barto, 1998). In the RL framework, the desired output of a control system is not explicitly specified. The controller, or the agent, has to find the right output by trying different action outputs at different states and monitoring the "rewards" associated with them.

In the early 80's, Barto and his colleagues proposed the "actor-critic" architecture for reinforcement learning of control tasks (Barto *et al.*, 1983, cf. Fig. 8). The architecture consists of an "actor" that produces stochastic action outputs and a "critic" that evaluates how good or bad the resulting state is.

Specifically, the actor implements a feedback control law, or a policy

$$\mathbf{u}(t) = G\big(\mathbf{x}(t)\big) + \nu(t), \tag{47}$$

where $\mathbf{u} \in \mathbb{R}^m$ is the action output, $\mathbf{x} \in \mathbb{R}^n$ is the system state, and $\nu \in \mathbb{R}^n$ denotes noise for exploration. In a deterministic environment, the state changes with

$$\mathbf{x}(t+1) = F\big(\mathbf{x}(t), \mathbf{u}(t)\big) \tag{48}$$

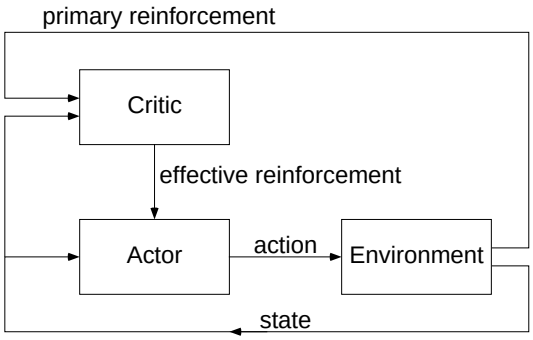


Fig. 8. The actor-critic architecture.

and a reward is given according to

$$r(t) = R\big(\mathbf{x}(t), \mathbf{u}(t)\big). \tag{49}$$

The role of the critic is to predict the cumulative future reward

$$V(\mathbf{x}) = E\big[r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \cdots \mid \mathbf{x}(t) = \mathbf{x}\big], \tag{50}$$

where $r(t+1), r(t+2), \ldots$ are determined by the evolution of the combined system (47)–(49). The parameter $0 \leq \gamma \leq 1$ is called the "discount factor." The critic makes a prediction of the above expected cumulative reward V from the current state $\mathbf{x}(t)$ using a look-up table or a function approximator like a neural network.

A variable that represents the inconsistency of temporarily adjacent predictions

$$\delta(t) = r(t) + \gamma V\big(\mathbf{x}(t+1)\big) - V\big(\mathbf{x}(t)\big), \tag{51}$$

is called the "temporal difference error" (TD error) and serves as a dual learning signal for the critic and the actor.

The goal of learning by the critic is to make an accurate prediction of cumulative future reward $V(\mathbf{x})$, i.e., to bring the TD error δ to zero. When the critic is parameterized as $V(\mathbf{x}; \mathbf{v})$, this is achieved by updating the parameters $\mathbf{v}$ by

$$\mathbf{v} := \mathbf{v} + \alpha_c \delta(t) \left(\frac{\partial V}{\partial \mathbf{v}}\right)^T, \tag{52}$$

where α_c is the learning rate.

The goal of learning by the actor is to improve the policy G so that the corresponding cumulative future reward V becomes larger. With proper learning by the critic, the average TD error δ should converge to zero. However, with the use of a stochastic policy, it fluctuates around zero. A positive TD error $\delta(t)$ implies that either the immediate reward $r(t)$ or the predicted future reward $V(\mathbf{x}(t+1))$ is greater

than expected. Accordingly, the TD error can be used to bias, or reinforce, the policy towards the direction of the previous output perturbation ν. When the actor is parameterized as $G(\mathbf{x};\mathbf{w})$, its parameters $\mathbf{w}$ are updated by

$$\mathbf{w} := \mathbf{w} + \alpha_a \delta(t)\nu(t) \left(\frac{\partial G}{\partial \mathbf{w}}\right)^T, \tag{53}$$

where α_a is the learning rate.

Although the derivations were performed by heuristics, the above TD learning algorithms have been successfully applied to several control tasks as well as game solving programs (Tesauro, 1994).

3.2. Insights from Dynamic Programming

The goal of reinforcement learning is quite similar to that of optimal control. Both are formulated as methods for maximizing the reward or minimizing the cost. A conceptual difference is that mathematical models of the system dynamics and the cost function are usually assumed in optimal control but they are supposed to be unknown, at least initially, in reinforcement learning.

In the late 80's, it was clarified that the output of the critic $V(\mathbf{x})$ is equivalent to the "value function" in the formulation of dynamic programming (DP). The main issue in the DP framework is to find a value function that satisfies the Bellman equation

$$V(\mathbf{x}) = \max_{\mathbf{u}} E\left[R(\mathbf{x},\mathbf{u}) + \gamma V\big(F(\mathbf{x},\mathbf{u})\big)\right]. \tag{54}$$

Then an optimal policy is derived from the value function V and the reward model R as

$$\mathbf{u} = G(\mathbf{x}) = \arg\max_{\mathbf{u}} E\left[R(\mathbf{x},\mathbf{u}) + \gamma V\big(F(\mathbf{x},\mathbf{u})\big)\right]. \tag{55}$$

In the conventional DP, the value function $V(\mathbf{x})$ is derived off-line by using the reward model $R(\mathbf{x},\mathbf{u})$ and the state transition model $F(\mathbf{x},\mathbf{u})$. In the actor-critic framework, the actor learns a policy that maximizes the right-hand side of the Bellman equation while the critic learns the value function to satisfy the Bellman equation by minimizing the TD error.

This interpretation of RL algorithms from the DP viewpoint enabled theoretical analyses of their convergent properties (Watkins, 1989; Dayan, 1992). It also enabled the development of novel reinforcement learning algorithms (Doya, 2000). It further promoted studies on the use of function approximation methods with DP, in which the value function may not be updated over the entire state space but appropriately learned only in those regions that are relevant for a given task. In turn, this enabled the application of RL or DP to a variety of optimization problems which had not been practically solved by traditional DP (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998).

3.2.1. Q Learning

One variant of reinforcement learning is called "Q learning," in which the term that should be maximized in the Bellman equation is defined as

$$Q(\mathbf{x},\mathbf{u}) = E\left[R(\mathbf{x},\mathbf{u}) + \gamma V\big(F(\mathbf{x},\mathbf{u})\big)\right]. \tag{56}$$

This is called the "action value function" for the action $\mathbf{u}$ at the state $\mathbf{x}$. A commonly used policy is the "softmax" function

$$P(\mathbf{u} \mid \mathbf{x}) = \frac{e^{\beta Q(\mathbf{x},\mathbf{u})}}{\sum\limits_{\mathbf{u}'\in U} e^{\beta Q(\mathbf{x},\mathbf{u}')}}, \tag{57}$$

where U is the set of possible actions. The parameter β, called the inverse temperature, controls the trade-off between the exploration of a new action and the exploitation of the knowledge learned. It is equivalent to the greedy policy (55) with $\beta \to \infty$.

In the original Q learning, the action value function is updated by

$$Q\big(\mathbf{x}(t),\mathbf{u}(t)\big) := (1-\alpha)Q\big(\mathbf{x}(t),\mathbf{u}(t)\big) + \alpha\left[r(t) + \gamma \max_{\mathbf{u}'\in U} Q\big(\mathbf{x}(t+1),\mathbf{u}'\big)\right], \tag{58}$$

where α_Q is the learning rate.

Therefore, instead of using models of reward R and dynamics F as in DP, the action value function is updated by the actual experience of reward $r(t)$ and the next state $\mathbf{x}(t+1)$. The convergence of Q learning was established for discrete systems with finite states and actions (Watkins, 1989). This also served as the basis for other convergence proofs of TD-based algorithms.

3.2.2. Continuous TD Learning

Most studies on reinforcement learning have assumed discrete states and actions updated in discrete time. However, RL algorithms for the case of continuous states, actions and time are helpful in applications to control tasks in which a smooth performance is desired. This was made possible based on the continuous-time formulation of dynamic programming (Doya, 2000).

For a continuous-time system

$$\dot{\mathbf{x}}(t) = F\big(\mathbf{x}(t),\mathbf{u}(t)\big), \tag{59}$$

the value function is defined as

$$V\big(\mathbf{x}(t)\big) = \int_t^\infty e^{\frac{s-t}{\tau}} r(t)\,\mathrm{d}s, \tag{60}$$

where τ is the time scale of evaluation. The optimality condition is given by the Hamilton-Jacobi-Bellman equation

$$\frac{1}{\tau}V(\mathbf{x}) = \max_{\mathbf{u}}\left[R(\mathbf{x},\mathbf{u}) + \frac{\partial V}{\partial \mathbf{x}}F(\mathbf{x},\mathbf{u})\right]. \tag{61}$$

A continuous-time counterpart of the TD error is defined as

$$\delta(t) = r(t) + \dot{V}\big(\mathbf{x}(t)\big) - \frac{1}{\tau} V\big(\mathbf{x}(t)\big). \tag{62}$$

The basic strategy is the same as in the discrete-time case: while estimating the value function by making the TD error $\delta(t)$ zero, update the action policy so that $\delta(t)$ is maximized.

When the model of the input gain $\partial F/\partial \mathbf{u}$ is known and the cost for the action is convex with respect to each output, i.e.,

$$R(\mathbf{x}, \mathbf{u}) = R(\mathbf{x}) + \sum_j S_j(u_j), \tag{63}$$

then the action that maximizes the TD error is given by

$$\mathbf{u} = s\left(\frac{\partial F}{\partial \mathbf{u}}^T \frac{\partial V}{\partial \mathbf{x}}^T\right), \tag{64}$$

where s is a component-wise inverse of the derivative of the action cost function, i.e., $s_j = (S_j')^{-1}$ (Doya, 2000).

It has been demonstrated that this learning control scheme is applicable to difficult nonlinear control problems, such as the swing up control of an inverted pendulum (Doya, 2000) and the learning of the stand-up behavior by a real robot hardware (Morimoto and Doya, 2000).

3.3. Reinforcement Learning Model of the Basal Ganglia

Learning based on rewards and punishment is one of the most essential elements of animal behaviors. The properties and mechanisms of reward-based learning have been well studied in psychology, ethology and neurobiology. An important fact found in these experimental studies is that predictability is a very important factor in reward-based learning. For example, a food reward can be used to reinforce a certain motor response. However, if there is a predictable relationship between a sensory stimulus, e.g., a light and the reward, the reward itself looses the role as a behavioral reinforcer and the preceding light can reinforce a motor response as if it were a reward signal.

The neurotransmitter dopamine has been known to be involved in the reinforcement of animal and human behaviors. For example, electric stimulation to dopaminergic pathways as well as drugs that increase the release of dopamine are strong behavioral reinforcers. It has been recently found in the recording of dopaminergic neurons in monkeys that those neurons initially respond to rewards, such as food and juice, but they later lose such responses if the reward is predictable from preceding sensory events, such as light or sound. In turn, the sensory stimulus that enables the monkey to predict the delivery of a future reward elicits the response of dopamine neurons (Schultz *et al.*, 1997).

This finding on the dopamine neuron response was a big surprise for researchers working on reinforcement learning models. Before the learning of the value function,

i.e., $V(\mathbf{x}) \equiv 0$, the TD error (51) is simply $\delta(t) = r(t)$, that is, the same as the actual reward. However, after the value function is well learned, $\delta(t) \simeq 0$ is satisfied most of the time. However, when a reward or a reward predicting sensory input is suddenly presented, there should be a temporary increase in the TD error. Therefore, the response of the dopamine neurons to a reward predicting stimulus exactly replicates the behavior of the TD error in RL algorithms. Furthermore, the role of the dopamine as a behavioral reinforcer is in accordance with the role of the TD error in the actor-critic architecture.

The basal ganglia receive strong dopaminergic inputs. They have been known to be involved in motor control because of severe motor symptoms for diseases in the basal ganglia, such as Parkinson's disease and Huntington's disease. However, their role in motor control in the normal brain has been a big enigma. The above finding concerning dopamine neurons has led to proposals of novel functional models of the basal ganglia (Houk *et al.*, 1995; Montague *et al.*, 1996).

Figure 9 illustrates the circuit of the basal ganglia. The input stage of the basal ganglia, the striatum, is composed of two functional compartments: the striosome and the matrix. The striosome projects to the dopaminergic neurons in the substantial nigra, and the matrix projects through multiple inhibitory pathways to the motor and premotor areas of the cerebral cortex. The dopaminergic neurons project back to the striatum and make synaptic contacts onto the cortico-striatal synapses (Wilson, 1998).

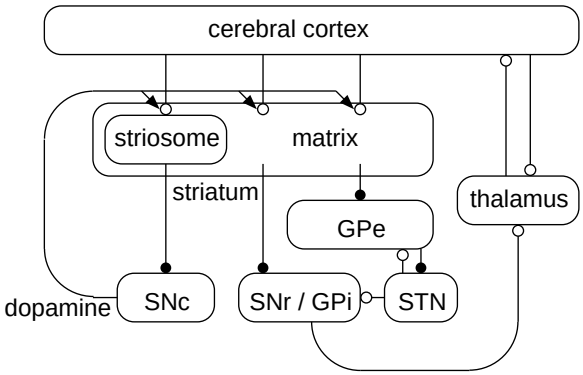


Fig. 9. The circuit of the basal ganglia; SNc and SNr: compact and reticular parts of the substantia nigra, GPe and GPi: external and internal segments of the global pallidus, STN: subthalamic nucleus, ○: excitatory synapse, ●: inhibitory synapse.

Based on this anatomical organization and data from the dopaminergic neurons, Barto, Houk and their colleagues proposed an actor-critic model of the basal ganglia (Barto, 1995; Houk *et al.*, 1995). In this model, the striosome works as a "critic" predicting the future reward, and the matrix works as an "actor" selecting motor outputs. Both compartments learn by using the TD signal carried by the dopaminergic neurons. This model nicely explains the roles of the basal ganglia in learning and execution of voluntary movements. Similar reinforcement learning models of the

basal ganglia have successfully replicated experimental data on reward-based learning (Hikosaka *et al.*, 1999; Nakahara *et al.*, 1998; Schultz *et al.*, 1997; Suri and Schultz, 1998) and contributed towards a better understanding of the function of the basal ganglia.

3.3.1. Action Value Function Model of the Basal Ganglia

It has been known that the neurons in a part of the basal ganglia called the caudate nucleus are involved in saccadic eye movements in particular directions. Recently, Kawagoe and her colleagues (Kawagoe *et al.*, 1998) showed that the activity of the caudate neurons in monkeys is strongly dependent on the expectations of reward. In an experiment in which a water or a juice reward were given only in one of four directions, the direction tuning of the caudate neurons was found to be strongly modulated by whether the movement was going to be rewarded or not. Even though eye movements in three of four directions were not rewarded, the animal performed the movement in order to move to the next trial, which might be rewarded. Consequently, the activity of the caudate neurons does not simply encode the movement command, but seems to encode the reward value associated with the movement.

Such a behavior is well characterized by the hypothesis that the caudate neurons encode the action value function $Q(\mathbf{x}, \mathbf{u})$ which is the predictor of a future reward if an action $\mathbf{u}$ is taken at a state $\mathbf{x}$. This hypothesis also suggests that a form of stochastic sampling of actual actions based on the predicted reward, such as (57), is implemented in the down-stream of the caudate nucleus. This model of action selection based on action value functions can provide a new working hypothesis about the role of multiple feed-forward connections in the basal ganglia circuit.

3.3.2. Interaction of the Basal Ganglia and the Cerebellum

Although "model-free" algorithms of reinforcement learning, such as the actor-critic and Q learning, are applicable to a wide range of tasks, their learning is desperately slow. If the model of the dynamics is known *a priori*, or has already been learned, it is advantageous to utilize such a model in reinforcement learning (Doya, 2000).

As outlined in the previous section, the most likely storage place of internal models of the body and the environment is the cerebellum. We have also mentioned the possibility that the basal ganglia learn the value function. How can these two be combined to implement a model-based reinforcement learning scheme, such as (55) and (64)? Although there is no direct anatomical connection between the cerebellum and the basal ganglia, they can communicate through the loop circuits with the cerebral cortex (Doya, 1999).

In the discrete paradigm, model-based action selection (55) can be implemented as follows. For an imaginary action candidate $\mathbf{u}^*$, the resulting state $\mathbf{x}^* = F(\mathbf{x}, \mathbf{u}^*)$ is predicted by the internal model F in the cerebellum and represented in the cerebral cortex. It is then sent to the basal ganglia, and the corresponding value $V(\mathbf{x}^*)$ is predicted. If it is large enough, the candidate action $\mathbf{u}^*$ is put to actual execution; if it is not, another cation candidate is considered.

In the continuous paradigm (64), the direction of the steepest increase in the value function $(\partial V/\partial \mathbf{x})^T$ is predicted, possibly by the basal ganglia, and represented in the cortex. It is then converted as an action command that is necessary to achieve the state change by a "transpose model" $(\partial F/\partial \mathbf{u})^T$ in the cerebellum.

With the availability of brain imaging technologies, such hypotheses of global collaboration of brain modules are now experimentally testable. For example, in a number of experiments on skill learning, the lateral cerebellum and the rostral premotor areas are highly activated. This suggests that subjects may use the above serial search strategy in learning a new task.

As we have seen above, the computational theory of reinforcement learning and its applications as models of brain mechanisms for goal-directed learning and behaviors can be quite helpful in exploring the possible roles of the neural circuit within the basal ganglia as well as the global circuit consisting of the basal ganglia, with the cerebral cortex and the cerebellum (Doya, 1999).

4. Modular Control with Multiple Models

4.1. Switching Control

The configuration of multiple models is often used in the context of *switching control* in control theory. Switching control is a control strategy that uses several controllers one after another depending on the state of the plant and its environment.

A switching controller is composed of a set of candidate controllers and a switching logic, which specifies the switching rule of selecting one controller at a time from candidate controllers. Switching controllers were used mainly for the control of a nonlinear plant by linear controllers. Sliding mode control is a typical example of such controllers, which dates back to the idea of variable structure control.

Recently, switching control has been used for adaptive control, where an unknown plant is represented not by a single model but by several candidate models (Morse, 1996). Corresponding to each candidate model of the plant, an appropriate controller is prepared. At each instance of time, one controller among the candidate controllers is chosen by the supervisor, the selection being based on the observation of the state and environment. A standard scheme of switching control based on multi-model configuration is shown in Fig. 10. Each controller C_i is constructed from a possible plant model P_i. The supervisor decides which controller to use based on which model is most likely the one in reference to the observation data.

4.2. Modular Learning by the "Soft" Competition of Predictors

It has been demonstrated in motor adaptation experiments, such as eye movement control, that de-adaptation to a normal condition and re-adaptation to a previously learned condition are much faster than adaptation to a new condition. This suggests that humans do not just tune the parameters of a single controller but retain multiple controllers for different situations and switch them on the fly. If a condition is varied

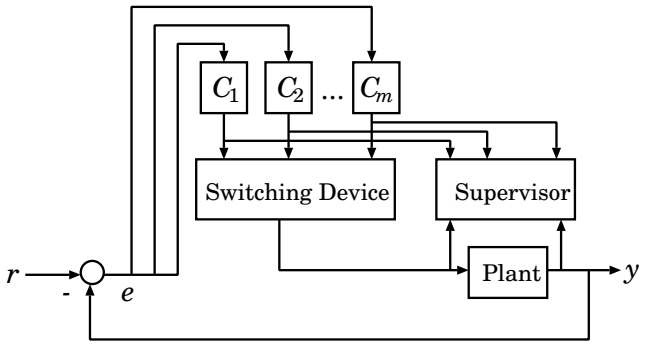


Fig. 10. A standard form of switching control.

between two learned ones, the motor output smoothly interpolates between the two cases, often following a sigmoid curve (Ghahramani and Wolpert, 1997).

Imamizu and his colleagues demonstrated that when a subject learns to use a new tool, a small part of the cerebellum is activated (Imamizu *et al.*, 2000). They also demonstrated that different local spots are activated when a subject uses different tools (Imamizu *et al.*, 1997). They suggested that such local activation spots in the cerebellum are the neural correlates of internal models of controlled objects, and that the cerebellum can learn and store multiple models for different objects and environmental settings. It has also been demonstrated in monkey experiments that a part of the premotor cortex, called pre-SMA, is activated when a motor task is switched to another (Shima *et al.*, 1996). These results suggest that the network linking the cerebellum and the premotor cortex may be involved in the selection of appropriated control modules for different conditions.

A basic question in using multiple controllers is how to select an appropriate controller under a given condition. A naive solution is to try each controller one by one and to select the one that offers the best performance. However, this is too costly when there is a large number of candidate controllers.

Wolpert and Kawato proposed a modular control architecture called the "multiple paired forward and inverse models" (MPFIM) (Wolpert and Kawato, 1998). This architecture comprises n pairs of a forward model and an inverse model of the controlled object (Fig. 11). An important fact is that although we can test only one inverse model controller at a time, it is possible to test multiple forward models simultaneously by comparing their prediction errors. This scheme was recently extended to the reinforcement learning paradigm as "multiple model-based reinforcement learning" (MMRL), in which each inverse model controller is replaced by a reinforcement learning agent (Doya *et al.*, 2000b). We call these architectures the multiple predictor-controller (MPC) architecture (Doya *et al.*, 2000a).

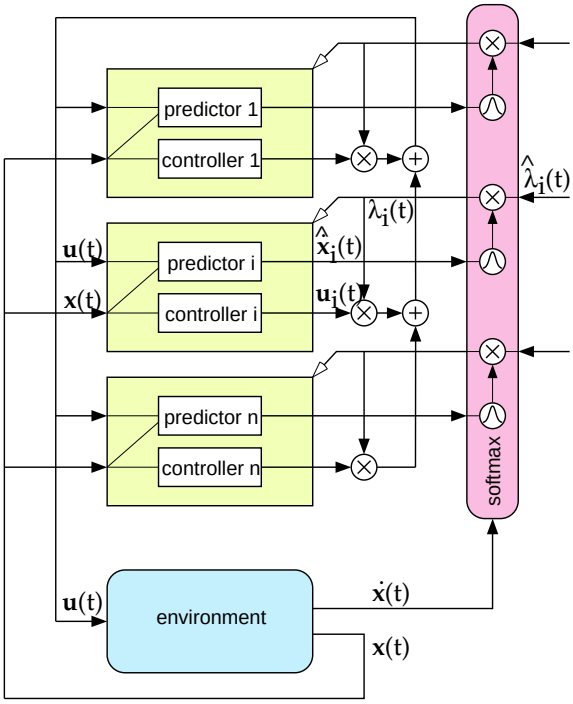


Fig. 11. The multiple predictor-controller (MPC) architecture. MPFIM uses an inverse model for the controller while a reinforcement learning module is used in MMRL.

In the MPC architecture, the outputs of the controllers as well as the learning of the predictors and the controllers are weighted by a "responsibility signal"

$$\lambda_i(t) = \frac{e^{-E_i(t)/2\sigma^2}}{\sum_{j=1}^{n} e^{-E_j(t)/2\sigma^2}}, \tag{65}$$

where $E_i(t)$ is the squared prediction error of the i-th prediction model. This formula is called the "softmax" function and the parameter σ controls the sharpness in module selection. For a continuous-time system

$$\dot{\mathbf{x}}(t) = F\big(\mathbf{x}(t), \mathbf{u}(t)\big), \tag{66}$$

we denote the output of the i-th predictor as $\hat{\dot{\mathbf{x}}}_i(t)$ and its prediction error is given by

$$E_i(t) = ||\hat{\dot{\mathbf{x}}}_i(t) - \dot{\mathbf{x}}(t)||^2. \tag{67}$$

The above responsibility signal $\lambda_i(t)$ is used for four purposes: (i) weighting the state prediction outputs, (ii) weighing the learning of prediction models, (iii) weighting the action outputs, and (iv) weighting the learning of controllers, as outlined below.

1. **State prediction:** The outputs of the prediction models are linearly weighted by the responsibility signal $\lambda_i(t)$ to make a prediction of the vector field

$$\hat{\dot{\mathbf{x}}}(t) = \sum_{i=1}^{n} \lambda_i(t) \hat{\dot{\mathbf{x}}}_i(t). \tag{68}$$

   These predictions are used in model-based control algorithms as well as for the annealing of σ, as will be described later.

2. **Prediction model learning:** $\lambda_i(t)$ is then used for weighting the parameter update of the prediction models. Namely, the error signal for the i-th prediction model is given by

$$\lambda_i(t) \left(\hat{\dot{\mathbf{x}}}_i(t) - \dot{\mathbf{x}}(t) \right). \tag{69}$$

3. **Action output:** The outputs of the controllers

$$\mathbf{u}_i(t) = g_i\big(\mathbf{x}(t)\big) \tag{70}$$

   are linearly weighted by $\lambda_i(t)$ to produce the action output to the environment

$$\mathbf{u}(t) = \sum_{i=1}^{n} \lambda_i(t) \mathbf{u}_i(t). \tag{71}$$

4. **Controller learning:** $\lambda_i(t)$ is also used for weighting the learning of the controllers. The actual equation for the parameter update varies with the choice of the control algorithms. When a temporal difference (TD) algorithm (Barto *et al.*, 1983; Sutton, 1988; Doya, 2000) is used, the TD error $\delta(t)$ weighted by the responsibility signal

$$\delta_i(t) = \lambda_i(t) \delta(t) \tag{72}$$

   is used as the teaching signal for the i-th RL controller.

Using the same weighting factor $\lambda_i(t)$ to train the prediction models and the controllers helps each RL controller to learn an appropriate control policy for the context in which its paired prediction model produces valid predictions.

4.3. Multiple Linear Quadratic Controllers

In the MPC architecture, the availability of the learned prediction models of the environment is helpful in the design of controllers. In general, reinforcement learning is notoriously slow for nonlinear, high-dimensional control tasks. However, when a locally linear model of the dynamics and a locally quadratic model of the reward or cost are available, an optimal value function is designed by solving the Riccati

equation. Accordingly, as a special case of the MPC architecture, we have formalized a multiple linear-quadratic controller (MLQC) architecture (Doya *et al.*, 2000b).

Figure 12 is an example of application of MLQC to the task of pendulum swing-up with a limited torque (Doya, 2000). This is a simple, but non-trivial, nonlinear control task. The goal is to bring the pendulum to the upright position. When the output torque is limited, several swinging motions are necessary to bring it upwards.

Two modules, each consisting of a locally linear dynamic model, a locally quadratic reward model, and an LQ controller, were devised. The parameters of the two modules were initially set randomly (Figs. 13(a) and (c)).

The following annealing process was used for the parameter σ which controls the sharpness of the responsibility signal (65):

$$\sigma_{k+1} = \eta a E_k + (1-\eta)\sigma_k, \tag{73}$$

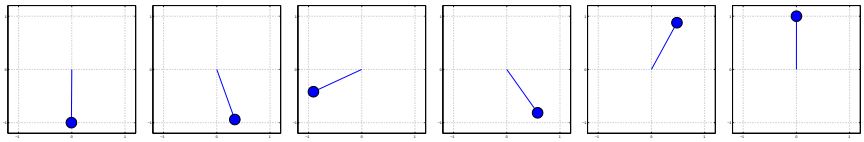
where k denotes the number of trials and E_k is the average state prediction error during the k-th trial.

Within about 100 learning trials, both the dynamic and reward models well approximated the sine- and cosine-shaped target functions, respectively, as shown in Figs. 13(b) and (d). Accordingly, successful swing-up was achieved with the first module, which made the downward position unstable, and the second module, which made the upright position stable (Fig. 12(b)). The two modules were appropriately switched by the responsibility signal $\lambda_i(t)$.

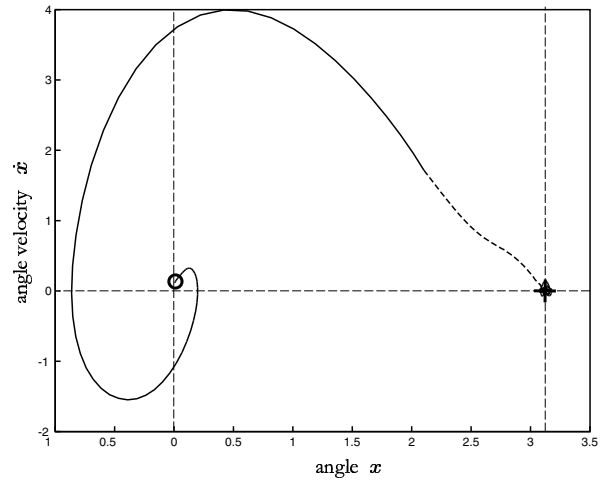
This example shows that the MPC architecture can appropriately handle a nonlinear control task by dividing it into multiple linear task domains. In other experiments, it has also been demonstrated that the MPC architecture can adapt to nonstationary environments, by selecting appropriate modules for different environmental settings (Haruno *et al.*, 1999; Doya *et al.*, 2000b).

A marked difference of MPC from the conventional switching control paradigm is the use of "soft" competition by the graded responsibility signal as opposed to "hard" switching. In the conventional switching control, the prediction models as well as the controllers are fixed while they are learned in the MPC architecture. The rationale for using soft competition is that if hard switching is used from the onset of learning, a suboptimal decomposition of the task based on premature prediction models is likely to be fixed. Therefore, it is desirable to use a large σ initially and gradually decrease σ as the learning proceeds (Pawelzik *et al.*, 1996), e.g., by (73).

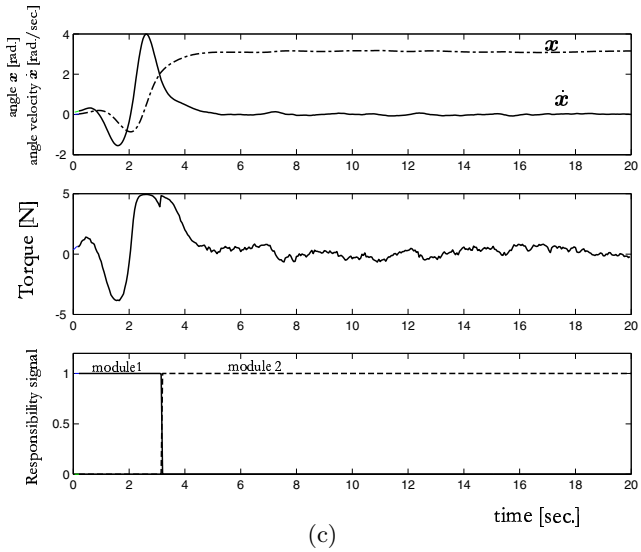
This biologically motivated control architecture using multiple adaptive models seems to have large potential in nonlinear/nonstationary control. However, there are still many open questions. In general, a mixture of outputs from two good controllers may turn out to be a bad strategy, e.g., pressing the accelerator and the brake at the same time at a yellow traffic light. Furthermore, the concatenation of locally optimal control policies is not guaranteed to produce a globally optimal policy. Such problems in adaptive multiple model-based control deserve serious investigations from the viewpoint of systems theory.



(a)



(b)



(c)

Fig. 12. (a) An example of swing-up performance for dynamics $ml^2\ddot{\theta} = -mgl\sin\theta - \mu\dot{\theta} + T$, where $m = l = 1$, $g = 9.8$, $\mu = 0.1$ and $T^{\max} = 5.0$. The reward is given by $r = -\cos(\theta) - \frac{1}{2}RT^2$. (b) The trajectory from the initial state $(0\,[\text{rad}], 0.1\,[\text{rad/s}])$; o: start, +: goal, solid line: Module 1, dashed line: Module 2. (c) Time evolution of the state (top), the action (middle) and the responsibility signal (bottom).

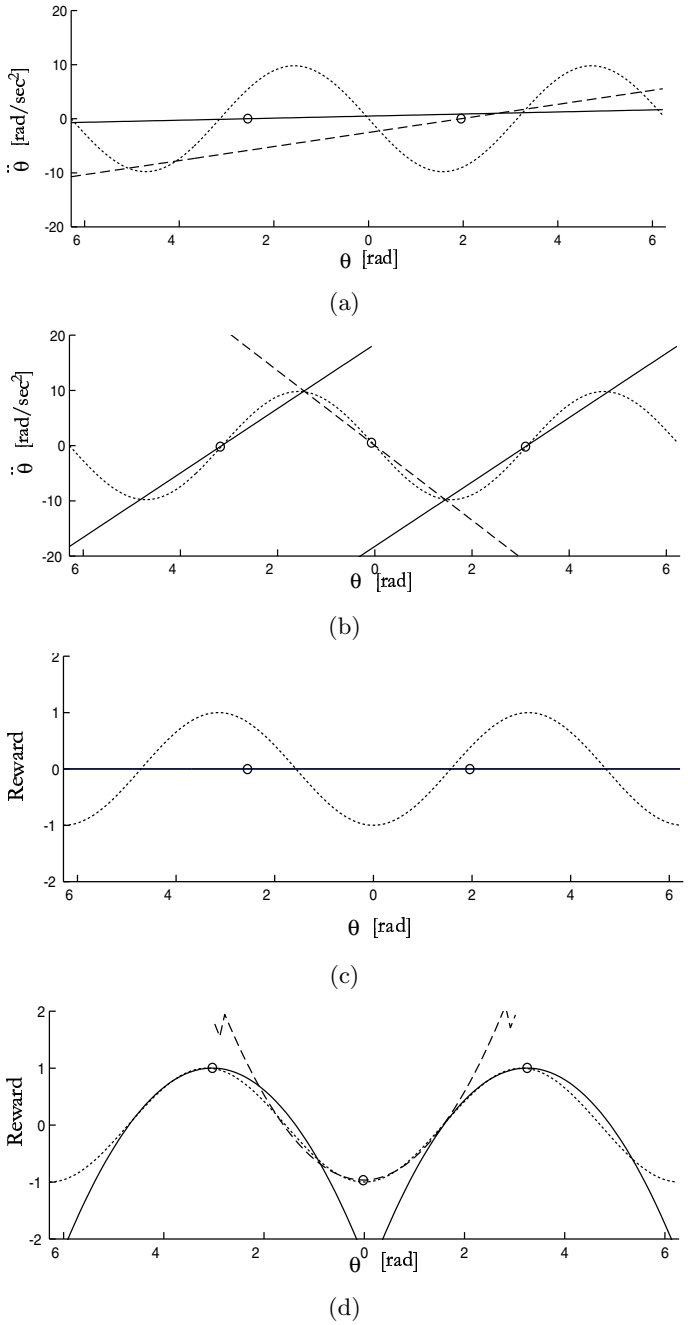


Fig. 13. Development of state and reward prediction of models. (a) and (b): Outputs of state prediction models before and after learning, respectively; (c) and (d): outputs of the reward prediction model before and after learning, respectively; solid line: Module 1, dashed line: Module 2, dotted line: targets ($\ddot{x}$ and r), ○: centers of spatial responsibility prediction $\mathbf{c}_i$.

5. Conclusion

As we have seen in the three examples, neuroscience and systems theory can play complementary roles in elucidating the design principles of robust, adaptive systems. Bidirectional interaction between the two research disciplines is highly important and can be very productive. We hope that better collaboration between the two communities will give rise to a better understanding of the principles of complex adaptive systems.

Acknowledgements

We thank Mitsno Kawato for helpful discussions. Kenji Doya is grateful to Kazuyuki Sameyima for preparation of figures.

References

Albus J.S. (1971): *A theory of cerebellar function.* — Math. Biosci., Vol.10, pp.25–61.

Åström K.J. and Wittenmark B. (1989): *Adaptive Control.* — Massachusetts: Addison Wesley.

Barto A.G. (1995): *Adaptive critics and the basal ganglia*, In: *Models of Information Processing in the Basal Ganglia* (Houk J.C., Davis J.L. and Beiser D.G., Eds.). — Cambridge, MA: MIT Press, pp.215–232.

Barto A.G., Sutton R.S. and Anderson C.W. (1983): *Neuronlike adaptive elements that can solve difficult learning control problems.* — IEEE Trans. Syst. Man Cybern., Vol.13, pp.834–846.

Bertsekas D.P. and Tsitsiklis J.N. (1996): *Neuro-Dynamic Programming.* — Belmont, MA: Athena Scientific.

Dayan P. (1992): *The convergence of TD(λ) for general λ.* — Machine Learn., Vol.8, pp.341–362.

Doya K. (1999): *What are the computations of the cerebellum, the basal ganglia, and the cerebral cortex.* — Neural Netw., Vol.12, No.7–8, pp.961–974.

Doya K. (2000): *Reinforcement learning in continuous time and space.* — Neural Comp., Vol.12, No.1, pp.243–269.

Doya K., Katagiri K., Wolpert D.M. and Kawato M. (2000a): *Recognition and imitation of movement paterns by a multiple predictor-controller architecture.* — Tech. Rep. Institute of Electronic, Information, and Communication Engineers, TL2000-11, pp.33–40.

Doya K., Samejima K., Katagiri K. and Kawato M. (2000b): *Multiple model-based reinforcement learning.* — Tech. Rep. KDB-08, Kawato Dynamic Brain Project, ERATO, Japan Science and Technology Corporation.

Ghahramani Z. and Wolpert D.M. (1997): *Modular decomposition in visuomotor learning.* — Nature, Vol.386, pp.392–395.

Ghez C. and Thach W.T. (2000): *The cerebellum*, In: *Principles of Neural Science, 4th Ed.* (Kandel E.R., Schwartz J.H. and Jessell T.M., Eds.). — New York: McGraw-Hill, pp.832–852.

Haruno M., Wolpert D.M. and Kawato M. (1999): *Multiple paired forward-inverse models for human motor learning and control*, In: *Advances in Neural Information Processing Systems, No.11* (Kearns M.S., Solla S.A. and Cohen D.A., Eds.). — Cambridge, MA: MIT Press, pp.31–37.

Hikosaka O., Nakahara H., Rand M.K., Sakai K., Lu X., Nakamura K., Miyachi S. and Doya K. (1999): *Parallel neural networks for learning sequential procedures.* — Trends in Neurosci., Vol.22, No.10, pp.464–471.

Houk J.C., Adams J.L. and Barto A.G. (1995): *A model of how the basal ganglia generate and use neural signals that predict reinforcement*, In: *Models of Information Processing in the Basal Ganglia* (Houk J.C., Davis J.L. and Beiser D.G., Eds.). — Cambridge, MA: MIT Press, pp.249–270.

Imamizu H., Miyauchi S., Sasaki Y., Takino R., Pütz B. and Kawato M. (1997): *Separated modules for visuomotor control and learning in the cerebellum: A functional MRI study*, In: *NeuroImage: Third International Conference on Functional Mapping of the Human Brain* (Toga A.W., Frackowiak R.S.J. and Mazziotta J.C., Eds.). — Copenhagen, Denmark: Academic Press, Vol.5, p.S598.

Imamizu H., Miyauchi S., Tamada T., Sasaki Y., Takino R., Putz B., Yoshioka T. and Kawato M. (2000): *Human cerebellar activity reflecting an acquired internal model of a new tool.* — Nature, Vol.403, pp.192–195.

Ito M. (1993): *Movement and thought: identical control mechanisms by the cerebellum.* — Trends in Neurosci., Vol.16, No.11, pp.448–450.

Ito M., Sakurai M. and Tongroach P. (1982): *Climbing fibre induced depression of both mossy fibre responsiveness and glutamate sensitivity of cerebellar purkinje cells.* — J. Physiol., Vol.324, pp.113–134.

Kawagoe R., Takikiwa Y. and Hikosaka O. (1998): *Expectation of reward modulates cognitive signals in the basal ganglia.* — Nature Neurosci., Vol.1, No.5, pp.411–416.

Kawato M., Furukawa K. and Suzuki R. (1987): *A hierarchical neural network model for control and learning of voluntary movement.* — Biol. Cybern., Vol.57, pp.169–185.

Kawato M. and Gomi H. (1992): *A computational model of four regions of the cerebellum based on feedback-error learning.* — Biol. Cybern., Vol.68, pp.95–103.

Marr D. (1969): *A theory of cerebellar cortex.* — J. Physiol., Vol.202, pp.437–470.

Miyamura A. and Kimura H. (2000): *Mathematical foundations of feedback error learning method.* — (submitted).

Montague P.R., Dayan P. and Sejnowski T.J. (1996): *A framework for mesencephalic dopamine systems based on predictive Hebbian learning.* — J. Neurosci., Vol.16, pp.1936–1947.

Morimoto J. and Doya K. (2000): *Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning.* — Proc. 17th Int. Conf. *Machine Learning*, Vancouver, Vol.1, pp.623–630.

Morse A.S. (1996): *Supervisory control of families of linear set-point controllers-part 1: Exact matching.* — IEEE Trans. Automat. Contr., Vol.41, No.10, pp.1413–1431.

Nakahara H., Doya K. and Hikosaka O. (1998): *Benefit of multiple representaitons for motor sequence control in the basal ganglia loops.* — BSIS Tech. Rep. 98–05, RIKEN Brain Science Institute.

Pawelzik K., Kohlmorge J. and Müller K.R. (1996): *Annealed competition of experts for a segmentation and classification of switching dynamics.* — Neural Comput., Vol.8, pp.340–356.

Schultz W., Dayan P. and Montague P.R. (1997): *A neural substrate of prediction and reward.* — Science, Vol.275, pp.1593–1599.

Shima K., Mushiake H., Saito N. and Tanji J. (1996): *Role for cells in the presupplementary motor area in updating motor plans.* — Proc. Nat. Acad. Sci., Vol.93, pp.8694–8698.

Suri R.E. and Schultz W. (1998): *Learning of sequential movements by neural network model with dopamine-like reinforcement signal.* — Experim. Brain Res., Vol.121, pp.350–354.

Sutton R.S. (1988): *Learning to predict by the methods of temporal difference.* — Machine Learn., Vol.3, pp.9–44.

Sutton R.S. and Barto A.G. (1998): *Reinforcement Learning.* — Cambridge, MA: MIT Press.

Tesauro G. (1994): *TD-Gammon, a self teaching backgammon program, achieves master-level play.* — Neural Comput., Vol.6, pp.215–219.

Watkins C.J.C.H. (1989): *Learning from delayed rewards.* — Ph.D. Thesis, Cambridge University.

Wilson C.W. (1998): *Basal ganglia*, In: *The Synaptic Organization of the Brain, 3rd Ed.* (Shepherd G.M., Ed.). — New York: Oxford University Press, pp.329–375.

Wolpert D.M. and Kawato M. (1998): *Multiple paired forward and inverse models for motor control.* — Neural Netw., Vol.11, pp.1317–1329.

Zames G. (1981): *Feedback and optimal sensitivity: Model reference transformations multiplicative seminors, and approximate inverses.* — IEEE Trans. Automat. Contr., Vol.AC-26, No.2, pp.301–320.