Reinforcement Learning In Continuous Time and Space

Kenji Doya*
ATR Human Information Processing Research Laboratories
2-2 Hikaridai, Seika, Soraku, Kyoto 619-0288, Japan

January 20, 1999
to appear in *Neural Computation*

Abstract

This paper presents a reinforcement learning framework for continuous-time dynamical systems without a priori discretization of time, state, and action. Based on the Hamilton-Jacobi-Bellman (HJB) equation for infinite-horizon, discounted reward problems, we derive algorithms for estimating value functions and for improving policies with the use of function approximators. The process of value function estimation is formulated as the minimization of a continuous-time form of the temporal difference (TD) error. Update methods based on backward Euler approximation and exponential eligibility traces are derived and their correspondences with the conventional residual gradient, TD(0), and TD(λ) algorithms are shown. For policy improvement, two methods, namely, a continuous actor-critic method and a value-gradient based greedy policy, are formulated. As a special case of the latter, a nonlinear feedback control law using the value gradient and the model of the input gain is derived. The "advantage updating", a model-free algorithm derived previously, is also formulated in the HJB based framework.

The performance of the proposed algorithms is first tested in a nonlinear control task of swinging up a pendulum with limited torque. It is shown in the simulations that 1) the task is accomplished by the continuous actor-critic method in a number of trials several times fewer than by the conventional discrete actor-critic method; 2) among the continuous policy update methods, the value-gradient based policy with a known or learned

*Current address: Kawato Dynamic Brain Project, ERATO, Japan Science and Technology Corporation. 2-2 Hikaridai, Seika, Soraku, Kyoto 619-0288, Japan. Phone: +81-774-95-1210. Fax: +81-774-95-3001. E-mail: doya@erato.atr.co.jp

dynamic model performs several times better than the actor-critic method; and 3) a value function update using exponential eligibility traces is more efficient and stable than that based on Euler approximation. The algorithms are then tested in a higher-dimensional task, i.e., cart-pole swing-up. This task is accomplished in several hundred trials using the value-gradient based policy with a learned dynamic model.

1 Introduction

The temporal difference (TD) family of reinforcement learning (RL) algorithms (Barto et al., 1983; Sutton, 1988; Sutton and Barto, 1998) provides an effective approach to control and decision problems for which optimal solutions are analytically unavailable or difficult to obtain. A number of successful applications to large-scale problems, such as board games (Tesauro, 1994), dispatch problems (Crites and Barto, 1996; Zhang and Dietterich, 1996; Singh and Bertsekas, 1997), and robot navigation (Mataric, 1994) have been reported (see, e.g., Kaelbling et al. (1996) and Sutton and Barto (1998) for a review). The progress of RL research so far, however, has been mostly constrained to the discrete formulation of the problem in which discrete actions are taken in discrete time steps based on the observation of the discrete state of the system.

Many interesting real-world control tasks, such as driving a car or riding a snowboard, require smooth continuous actions taken in response to high-dimensional, real-valued sensory input. In applications of RL to continuous problems, the most common approach has been first to discretize time, state, and action and then to apply an RL algorithm for a discrete stochastic system. However, this discretization approach has the following drawbacks:

1. When a coarse discretization is used, the control output is not smooth, resulting in a poor performance.

2. When a fine discretization is used, the number of states and the number of iteration steps become huge, which necessitates not only large memory storage but also many learning trials.

3. In order to keep the number of states manageable, an elaborate partitioning of the variables has to be found using prior knowledge.

Efforts have been made to eliminate some of these difficulties by using appropriate function approximators (Gordon, 1996; Sutton, 1996; Tsitsiklis and Van Roy, 1997), adaptive state partitioning and aggregation methods (Moore, 1994; Singh et al., 1995; Asada et al., 1996; Pareigis, 1998), and multiple time scale methods (Sutton, 1995).

In this paper, we consider an alternative approach in which learning algorithms are formulated for continuous-time dynamical systems without resorting to the explicit discretization of time, state and action. The continuous framework has the following possible advantages:

1. A smooth control performance can be achieved.
2. An efficient control policy can be derived using the gradient of the value function (Werbos, 1990).
3. There is no need to guess how to partition the state, action, and time: it is the task of the function approximation and numerical integration algorithms to find the right granularity.

There have been several attempts at extending RL algorithms to continuous cases. Bradtke (1993) showed convergence results for Q-learning algorithms for discrete-time, continuous-state systems with linear dynamics and quadratic costs. Bradtke and Duff (1995) derived a TD algorithm for continuous-time, discrete-state systems (semi-Markov decision problems). Baird (1993) proposed the "advantage updating" method by extending Q-learning to be used for continuous-time, continuous-state problems.

When we consider optimization problems in continuous-time systems, the Hamilton-Jacobi-Bellman (HJB) equation, which is a continuous-time counterpart of the Bellman equation for discrete-time systems, provides a sound theoretical basis (see, e.g., Bertsekas (1995) and Fleming and Soner (1993)). Methods for learning the optimal value function that satisfies the HJB equation have been studied using a grid-based discretization of space and time (Peterson, 1993) and convergence proofs have been shown for gird sizes taken to zero (Munos, 1997; Munos and Bourgine, 1998). However, the direct implementation of such methods is impractical in a high-dimensional state space. An HJB based method that uses function approximators was presented by Dayan and Singh (1996) . They proposed the learning of the gradients of the value function without learning the value function itself, but the method is applicable only to non-discounted reward problems.

This paper presents a set of RL algorithms for nonlinear dynamical systems based on the Hamilton-Jacobi-Bellman equation for infinite-horizon, discounted reward problems. A series of simulations are devised to evaluate their effectiveness when used with continuous function approximators.

We first consider methods for learning the value function on the basis of minimizing a continuous-time form of the TD error. The update algorithms are derived either by using a single step or exponentially weighed eligibility traces. The relationships of these algorithms with the residual gradient (Baird, 1995), TD(0), and TD(λ) algorithms (Sutton, 1988) for discrete cases are also shown. Next, we formulate methods for improving the policy using

the value function, namely, the continuous actor-critic method and a value-gradient based policy. Specifically, when a model is available for the input gain of the system dynamics, we derive a closed-form feedback policy that is suitable for real-time implementation. Its relationship with "advantage updating" (Baird, 1993) is also discussed.

The performance of the proposed methods is first evaluated in nonlinear control tasks of swinging up a pendulum with limited torque (Atkeson, 1994; Doya, 1996) using normalized Gaussian basis function networks for representing the value function, the policy, and the model. We test: 1) the performance of the discrete actor-critic, continuous actor-critic, and value-gradient based methods; 2) the performance of the value function update methods; and 3) the effects of the learning parameters, including the action cost, exploration noise, and landscape of the reward function. Then, we test the algorithms in a more challenging task, i.e., cart-pole swing-up (Doya, 1997), in which the state space is higher-dimensional and the system input gain is state-dependent.

2 The Optimal Value Function for a Discounted Reward Task

In this paper, we consider the continuous-time deterministic system

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{1}$$

where $\mathbf{x} \in X \subset \mathbf{R}^n$ is the state and $\mathbf{u} \in U \subset \mathbf{R}^m$ is the action (control input). We denote the immediate reward for the state and the action as

$$r(t) = r(\mathbf{x}(t), \mathbf{u}(t)). \tag{2}$$

Our goal is to find a *policy* (control law)

$$\mathbf{u}(t) = \mu(\mathbf{x}(t)) \tag{3}$$

that maximizes the cumulative future rewards

$$V^{\mu}(\mathbf{x}(t)) = \int_t^{\infty} e^{-\frac{s-t}{\tau}} r(\mathbf{x}(s), \mathbf{u}(s)) ds \tag{4}$$

for any initial state $\mathbf{x}(t)$. Note that $\mathbf{x}(s)$ and $\mathbf{u}(s)$ $(t \leq s < \infty)$ follow the system dynamics (1) and the policy (3). $V^{\mu}(\mathbf{x})$ is called the *value function* of the state $\mathbf{x}$ and τ is the time constant for discounting future rewards. An important feature of this infinite-horizon formulation is that the value function and the optimal policy do not depend explicitly on time, which is convenient in estimating them using function approximators. The discounted reward makes it unnecessary to assume that the state is attracted to a zero-reward state.

The value function V^* for the optimal policy μ^* is defined as

$$V^*(\mathbf{x}(t)) = \max_{\mathbf{u}[t,\infty)} \left[\int_t^\infty e^{-\frac{s-t}{\tau}} r(\mathbf{x}(s), \mathbf{u}(s)) ds \right], \tag{5}$$

where $\mathbf{u}[t, \infty)$ denotes the time course $\mathbf{u}(s) \in U$ for $t \leq s < \infty$. According to the principle of optimality, the condition for the *optimal value function* at time t is given by

$$\frac{1}{\tau} V^*(\mathbf{x}(t)) = \max_{\mathbf{u}(t) \in U} \left[r(\mathbf{x}(t), \mathbf{u}(t)) + \frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}(t), \mathbf{u}(t)) \right], \tag{6}$$

which is a discounted version of the Hamilton-Jacobi-Bellman equation (see Appendix A). The optimal policy is given by the action that maximizes the right-hand side of the HJB equation

$$\mathbf{u}(t) = \mu^*(\mathbf{x}(t)) = \arg\max_{\mathbf{u} \in U} \left[r(\mathbf{x}(t), \mathbf{u}) + \frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}(t), \mathbf{u}) \right]. \tag{7}$$

Reinforcement learning can be formulated as the process of bringing the current policy μ and its value function estimate V closer to the optimal policy μ^* and the optimal value function V^*. It generally involves two components:

1. Estimate the value function V based on the current policy μ.

2. Improve the policy μ by making it greedy with respect to the current estimate of the value function V.

We will consider the algorithms for these two processes in the following two sections.

3 Learning the Value Function

For the learning of the value function in a continuous state space, it is mandatory to use some form of function approximator. We denote the current estimate of the value function as

$$V^\mu(\mathbf{x}(t)) \simeq V(\mathbf{x}(t); \mathbf{w}), \tag{8}$$

where $\mathbf{w}$ is a parameter of the function approximator, or simply, $V(t)$. In the framework of TD learning, the estimate of the value function is updated using a self-consistency condition that is local in time and space. This is given by differentiating definition (4) by t as

$$\dot{V}^\mu(\mathbf{x}(t)) = \frac{1}{\tau} V^\mu(\mathbf{x}(t)) - r(t). \tag{9}$$

Note that this should hold for any policy including the optimal policy given by (7).

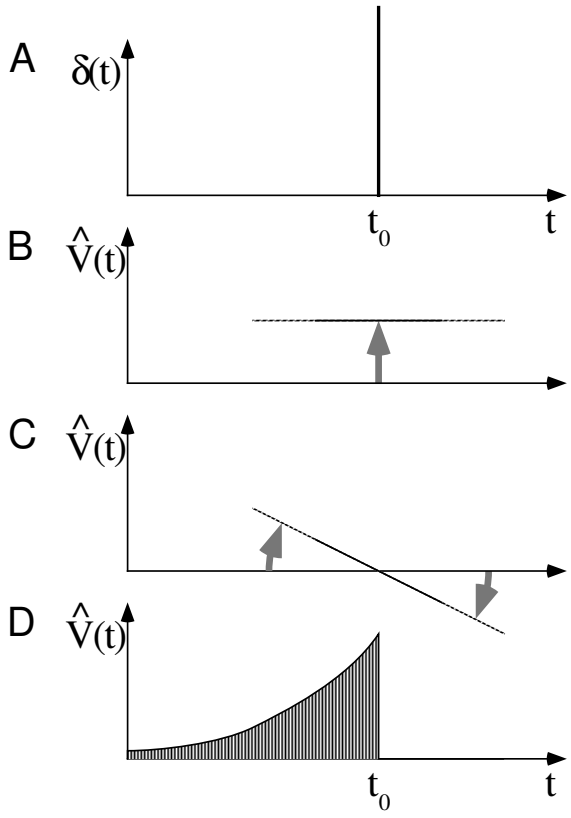

Figure 1: Possible updates for the value function estimate $\hat{V}(t)$ for an instantaneous TD error $\delta(t) = r(t) - \frac{1}{\tau}V(t) + \dot{V}(t)$. (A) A positive TD error at $t = t_0$ can be corrected by (B) an increase in $V(t)$, or (C) a decrease in the time derivative $\dot{V}(t)$, or (D) an exponentially weighted increase in $V(t)$ ($t < t_0$).

If the current estimate V of the value function is perfect, it should satisfy the consistency condition $\dot{V}(t) = \frac{1}{\tau}V(t) - r(t)$. If this condition is not satisfied, the prediction should be adjusted to decrease the inconsistency

$$\delta(t) \equiv r(t) - \frac{1}{\tau}V(t) + \dot{V}(t). \tag{10}$$

This is the continuous-time counterpart of the *TD error* (Barto et al., 1983; Sutton, 1988).

3.1 Updating the Level and the Slope

In order to bring the TD error (10) to zero, we can tune either the level of the value function $V(t)$, its time derivative $\dot{V}(t)$, or both, as illustrated in Figure 1(A,B,C). Now we consider the objective function (Baird, 1993)

$$E(t) = \frac{1}{2}|\delta(t)|^2. \tag{11}$$

From definition (10) and the chain rule $\dot{V}(t) = \frac{\partial V}{\partial \mathbf{x}}\dot{\mathbf{x}}(t)$, the gradient of the objective function with respect to a parameter w_i is given by

$$\frac{\partial E(t)}{\partial w_i} = \delta(t)\frac{\partial}{\partial w_i}\left[r(t) - \frac{1}{\tau}V(t) + \dot{V}(t)\right] = \delta(t)\left[-\frac{1}{\tau}\frac{\partial V(\mathbf{x};\mathbf{w})}{\partial w_i} + \frac{\partial}{\partial w_i}\left(\frac{\partial V(\mathbf{x};\mathbf{w})}{\partial \mathbf{x}}\right)\dot{\mathbf{x}}(t)\right].$$

Therefore, the gradient descent algorithm is given by

$$\dot{w}_i = -\eta\frac{\partial E}{\partial w_i} = \eta\delta(t)\left[\frac{1}{\tau}\frac{\partial V(\mathbf{x};\mathbf{w})}{\partial w_i} - \frac{\partial}{\partial w_i}\left(\frac{\partial V(\mathbf{x};\mathbf{w})}{\partial \mathbf{x}}\right)\dot{\mathbf{x}}(t)\right], \tag{12}$$

where η is the learning rate.

A potential problem with this update algorithm is its symmetry in time. Since the boundary condition for the value function is given at $t \to \infty$, it would be more appropriate to update the past estimates without affecting the future estimates. Below, we consider methods for implementing the "back-up" of TD errors.

3.2 Backward Euler Differentiation: Residual Gradient and TD(0)

One way of implementing the back-up of TD errors is to use the backward Euler approximation of time derivative $\dot{V}(t)$. By substituting $\dot{V}(t) = (V(t) - V(t-\Delta t))/\Delta t$ into (10), we have

$$\delta(t) = r(t) + \frac{1}{\Delta t}\left[(1 - \frac{\Delta t}{\tau})V(t) - V(t-\Delta t)\right]. \tag{13}$$

Then, the gradient of the squared TD error (11) with respect to the parameter w_i is given by

$$\frac{\partial E(t)}{\partial w_i} = \delta(t)\frac{1}{\Delta t}\left[(1 - \frac{\Delta t}{\tau})\frac{\partial V(\mathbf{x}(t);\mathbf{w})}{\partial w_i} - \frac{\partial V(\mathbf{x}(t-\Delta t);\mathbf{w})}{\partial w_i}\right].$$

A straightforward gradient descent algorithm is given by

$$\dot{w}_i = \eta\delta(t)\left[-(1-\frac{\Delta t}{\tau})\frac{\partial V(\mathbf{x}(t);\mathbf{w})}{\partial w_i} + \frac{\partial V(\mathbf{x}(t-\Delta t);\mathbf{w})}{\partial w_i}\right]. \tag{14}$$

An alternative way is to update only $V(t-\Delta t)$ without explicitly changing $V(t)$ by

$$\dot{w}_i = \eta\delta(t)\frac{\partial V(\mathbf{x}(t-\Delta t);\mathbf{w})}{\partial w_i}. \tag{15}$$

The Euler discretized TD error (13) coincides with the conventional TD error

$$\delta_t = r_t + \gamma V_t - V_{t-1}$$

by taking the discount factor $\gamma = 1 - \frac{\Delta t}{\tau} \simeq e^{-\frac{\Delta t}{\tau}}$ and rescaling the values as $V_t = \frac{1}{\Delta t}V(t)$. The update schemes (14) and (15) correspond to the residual-gradient (Baird, 1995; Harmon et al., 1996) and TD(0) algorithms, respectively. Note that time step Δt of the Euler differentiation does not have to be equal to the control cycle of the physical system.

3.3 Exponential Eligibility Trace: TD(λ)

Now let us consider how an instantaneous TD error should be corrected by a change in the value V as a function of time. Suppose an impulse of reward $r(t) = \delta(t-t_0)$ is given at time $t = t_0$. Then, from definition (4), the corresponding temporal profile of the value function is

$$V^{\mu}(t) = \begin{cases} e^{-\frac{t_0-t}{\tau}} & t \le t_0, \\ 0 & t > t_0, \end{cases}$$

Because the value function is linear with respect to the reward, the desired correction of the value function for an instantaneous TD error $\delta(t_0)$ is

$$\hat{V}(t) = \begin{cases} \delta(t_0)e^{-\frac{t_0-t}{\tau}} & t \le t_0, \\ 0 & t > t_0, \end{cases}$$

as illustrated in Figure 1(D). Therefore, the update of w_i given $\delta(t_0)$ should be made as

$$\dot{w}_i = \eta\int_{-\infty}^{t_0}\hat{V}(t)\frac{\partial V(\mathbf{x}(t);\mathbf{w})}{\partial w_i}dt = \eta\delta(t_0)\int_{-\infty}^{t_0}e^{-\frac{t_0-t}{\tau}}\frac{\partial V(\mathbf{x}(t);\mathbf{w})}{\partial w_i}dt. \tag{16}$$

We can consider the exponentially weighted integral of the derivatives as the eligibility trace e_i for the parameter w_i. Then, a class of learning algorithms is derived as

$$\dot{w}_i = \eta\delta(t)e_i(t),$$

$$\kappa\dot{e}_i(t) = -e_i(t) + \frac{\partial V(\mathbf{x}(t);\mathbf{w})}{\partial w_i}, \tag{17}$$

where $0 < \kappa \leq \tau$ is the time constant of the eligibility trace.

If we discretize (17) with time step Δt, it coincides with the eligibility trace update in TD(λ)

$$e_i(t + \Delta t) = \lambda\gamma e_i(t) + \frac{\partial V_t}{\partial w_i}$$

with $\lambda = \frac{\kappa - \Delta t}{\tau - \Delta t}$.

4 Improving the Policy

Now we consider ways for improving the policy $\mathbf{u}(t) = \mu(\mathbf{x}(t))$ using its associated value function $V(\mathbf{x})$. One way is to stochastically improve the policy using the actor-critic method, in which the TD error is used as the effective reinforcement signal. Another way is to take a greedy policy with respect to the current value function

$$\mathbf{u}(t) = \mu(\mathbf{x}(t)) = \arg\max_{\mathbf{u} \in U} \left[r(\mathbf{x}(t), \mathbf{u}) + \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}(t), \mathbf{u}) \right] \tag{18}$$

using the knowledge about the reward and the system dynamics.

4.1 Continuous Actor-Critic

First, we derive a continuous version of the actor-critic method (Barto et al., 1983). By comparing (10) and (18), we can see that the TD error is maximized by the greedy action $\mathbf{u}(t)$. Accordingly, in the actor-critic method, the TD error is used as the reinforcement signal for policy improvement.

We consider the policy implemented by the actor as

$$\mathbf{u}(t) = s\left(A(\mathbf{x}(t); \mathbf{w}^A) + \sigma \mathbf{n}(t)\right), \tag{19}$$

where $A(\mathbf{x}(t); \mathbf{w}^A) \in R^m$ is a function approximator with a parameter vector $\mathbf{w}^A$, $\mathbf{n}(t) \in R^m$ is noise, and $s()$ is a monotonically increasing output function. The parameters are updated by the stochastic real-valued (SRV) unit algorithm (Gullapalli, 1990) as

$$\dot{w}_i^A = \eta^A \delta(t) \mathbf{n}(t) \frac{\partial A(\mathbf{x}(t); \mathbf{w}^A)}{\partial w_i^A}. \tag{20}$$

4.2 Value-Gradient Based Policy

In discrete problems, a greedy policy can be found by one-ply search for an action that maximizes the sum of the immediate reward and the value of the next state. In the continuous case, the right hand side of (18) has to be minimized over a continuous set of actions at

every instant, which can in general be computationally expensive. However, when the reinforcement $r(\mathbf{x}, \mathbf{u})$ is convex with respect to the action $\mathbf{u}$ and the system dynamics $f(\mathbf{x}, \mathbf{u})$ is linear with respect to the action $\mathbf{u}$, the optimization problem in (18) has a unique solution and we can derive a closed-form expression of the greedy policy.

Here, we assume that the reward $r(\mathbf{x}, \mathbf{u})$ can be separated into two parts: the reward for state $R(\mathbf{x})$, which is given by the environment and unknown, and the cost for action $S(\mathbf{u})$, which can be chosen as a part of the learning strategy. We specifically consider the case

$$r(\mathbf{x}, \mathbf{u}) = R(\mathbf{x}) - \sum_{j=1}^{m} S_j(u_j), \tag{21}$$

where $S_j()$ is a cost function for action variable u_j. In this case, the condition for the greedy action (18) is given by

$$-S_j'(u_j) + \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial u_j} = 0 \qquad (j = 1, ..., m)$$

where $\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial u_j}$ is the j-th column vector of the $n \times m$ input gain matrix $\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}$ of the system dynamics. We now assume that the input gain $\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}$ is not dependent on $\mathbf{u}$, i.e., the system is linear with respect to the input, and that the action cost function $S_j()$ is convex. Then, the the above equation has a unique solution $u_j = {S_j'}^{-1}\left(\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial u_j}\right)$, where $S_j'()$ is a monotonic function. Accordingly, the greedy policy is represented in vector notation as

$$\mathbf{u} = {S'}^{-1}\left(\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}^T \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}^T\right), \tag{22}$$

where $\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}^T$ represents the steepest ascent direction of the value function, which is then transformed by the "transpose" model $\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}^T$ into a direction in the action space, and the actual amplitude of the action is determined by gain function ${S'}^{-1}()$.

Note that the gradient $\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}$ can be calculated by back-propagation when the value function is represented by a multi-layer network. The assumption of linearity with respect to the input is valid in most Newtonian mechanical systems (e.g., the acceleration is proportional to the force) and the gain matrix $\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}$ can be calculated from the inertia matrix. When the dynamics is linear and the reward is quadratic, the value function is also quadratic and (22) coincides with the optimal feedback law for a linear quadratic regulator (LQR; see, e.g., Bertsekas (1995)).

Feedback control with a sigmoid output function

A common constraint in control tasks is that the amplitude of the action, such as the force or torque, is bounded. Such a constraint can be incorporated into the above policy with an appropriate choice of the action cost.

Suppose that the amplitude of the action is limited as $|u_j| \leq u_j^{\max}$ $(j = 1, ..., m)$. We then define the action cost as

$$S_j(u_j) = c_j \int_0^{u_j} s^{-1}\left(\frac{u}{u_j^{\max}}\right) du, \tag{23}$$

where $s()$ is a sigmoid function that saturates as $s(\pm\infty) = \pm 1$. In this case, the greedy feedback policy (22) results in feedback control with a sigmoid output function

$$u_j = u_j^{\max} s\left(\frac{1}{c_j}\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial u_j}^T \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}^T\right). \tag{24}$$

In the limit of $c_j \to 0$, the policy will be a "bang-bang" control law

$$u_j = u_j^{\max} \text{ sign}\left[\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial u_j}^T \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}^T\right]. \tag{25}$$

4.3 Advantage Updating

When the model of the dynamics is not available, like in Q-learning (Watkins, 1989), we can select a greedy action by directly learning the term to be maximized in the HJB equation

$$r(\mathbf{x}(t), \mathbf{u}(t)) + \frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}(t), \mathbf{u}(t)).$$

This idea has been implemented in the "advantage updating" method (Baird, 1993; Harmon et al., 1996) in which both the value function $V(\mathbf{x})$ and the *advantage function* $A(\mathbf{x}, \mathbf{u})$ are updated. The optimal advantage function $A^*(\mathbf{x}, \mathbf{u})$ is represented in the current HJB formulation as

$$A^*(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}) - \frac{1}{\tau} V^*(\mathbf{x}) + \frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}), \tag{26}$$

which takes the maximum value of zero for the optimal action $\mathbf{u}$. The advantage function $A(\mathbf{x}, \mathbf{u})$ is updated by

$$A(\mathbf{x}, \mathbf{u}) \leftarrow \max_u [A(\mathbf{x}, \mathbf{u})] + r(\mathbf{x}, \mathbf{u}) - \frac{1}{\tau} V^*(\mathbf{x}) + \dot{V}(\mathbf{x}) = \max_u [A(\mathbf{x}, \mathbf{u})] + \delta(t) \tag{27}$$

under the constraint $\max_u [A(\mathbf{x}, \mathbf{u})] = 0$.

The main difference between the advantage updating and the value-gradient based policy described above is while the value V and the advantage A are updated in the former, the value V and the model f are updated and their derivatives are used in the latter. When the input gain model $\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}$ is known or easy to learn, the use of the closed-form policy (22) in the latter approach is advantageous because it simplifies the process of maximizing the right-hand side of the HJB equation.

5 Simulations

We tested the performance of the continuous RL algorithms in two nonlinear control tasks; a pendulum swing-up task (n=2, m=1) and a cart-pole swing-up task (n=4, m=1). In each of these tasks, we compared the performance of three control schemes:

1. Actor-critic: control by (19) and learning by (20).
2. Value-gradient based policy (24) with an exact gain matrix.
3. Value-gradient based policy (24) with concurrent learning of the input gain matrix.

The value functions were updated using the exponential eligibility trace (17) except in the experiments of Figure 6.

Both the value and policy functions were implemented by normalized Gaussian networks, as described in Appendix B. A sigmoid output function $s(x) = \frac{2}{\pi}\arctan(\frac{\pi}{2}x)$ (Hopfield, 1984) was used in both (19) and (24).

In order to promote exploration, we incorporated a noise term $\sigma\mathbf{n}(t)$ in both policies (19) and (24) (see equations (33) and (34) in Appendix B). We used low-pass filtered noise $\tau_n\dot{\mathbf{n}}(t) = -\mathbf{n}(t) + \mathbf{N}(t)$ where $\mathbf{N}(t)$ denotes normal Gaussian noise. The size of the perturbation σ was tapered off as the performance improved (Gullapalli, 1990). We took the modulation scheme $\sigma = \sigma_0 \min\left[1, \max\left[0, \frac{V_1 - V(t)}{V_1 - V_0}\right]\right]$, where V_0 and V_1 are the minimal and maximal levels of the expected reward.

The physical systems were simulated by a fourth order Runge-Kutta method while the learning dynamics was simulated by a Euler method, both with the time step of 0.02 sec.

5.1 Pendulum Swing-Up with Limited Torque

First, we tested the continuous-time RL algorithms in the task of a pendulum swinging upwards with limited torque (Figure 2) (Atkeson, 1994; Doya, 1996). The control of this one degree of freedom system is non-trivial if the maximal output torque $u^{\max}$ is smaller than the maximal load torque mgl. The controller has to swing the pendulum several times to build up momentum and also has to decelerate the pendulum early enough to prevent the pendulum from falling over.

The reward was given by the height of the tip of the pendulum, i.e., $R(\mathbf{x}) = \cos\theta$. The policy and value functions were implemented by normalized Gaussian networks with 15×15 basis functions to cover the two-dimensional state space $\mathbf{x} = (\theta, \omega)$. In modeling the system dynamics, $15 \times 15 \times 2$ bases were used for the state-action space (θ, ω, u).

Each trial was started from an initial state $\mathbf{x}(0) = (\theta(0), 0)$ where $\theta(0)$ was selected randomly in $[-\pi, \pi]$. A trial lasted for 20 seconds unless the pendulum was over-rotated

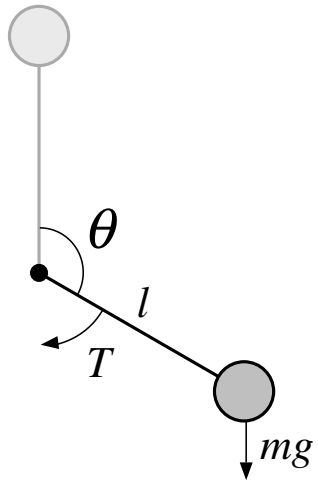

Figure 2: Control of a pendulum with limited torque. The dynamics were given by $\dot{\theta} = \omega$ and $ml^2\dot{\omega} = -\mu\omega + mgl\sin\theta + u$. The physical parameters were $m = l = 1$, $g = 9.8$, $\mu = 0.01$, and $u^{\max} = 5.0$. The learning parameters were $\tau = 1.0$, $\kappa = 0.1$, $c = 0.1$, $\tau_n = 1.0$, $\sigma_0 = 0.5$, $V_0 = 0$, $V_1 = 1$, $\eta = 1.0$, $\eta^A = 5.0$, and $\eta^M = 10.0$, in the following simulations unless otherwise specified.

($|\theta| > 5\pi$). Upon such a failure, the trial was terminated with a reward $r(t) = -1$ for one second. As a measure of the swing-up performance, we defined the time in which the pendulum stayed up ($|\theta| < \pi/4$) as t_{up}. A trial was regarded as "successful" when $t_{up} > 10$ seconds. We used the number of trials made before achieving ten successful trials as the measure of the learning speed.

Figure 3 illustrates the landscape of the value function and a typical trajectory of swing-up using the value-gradient based policy. The trajectory starts from the bottom the basin, which corresponds to the pendulum hanging down, and spirals up the hill along the ridge of the value function until it reaches to the peak, which corresponds to the pendulum standing upright.

Actor-Critic, Value Gradient and Physical Model

We first compared the performance of the three continuous RL algorithms with the discrete actor-critic algorithm (Barto et al., 1983). Figure 4 shows the time course of learning in five simulation runs and Figure 5 shows the average number of trials needed until ten successful swing-ups. The discrete actor-critic algorithm took about five times more trials than the continuous actor-critic. Note that the continuous algorithms were simulated with the same time step as the discrete algorithm. Consequently, the performance difference was due to a better spatial generalization with the normalized Gaussian networks. Whereas the

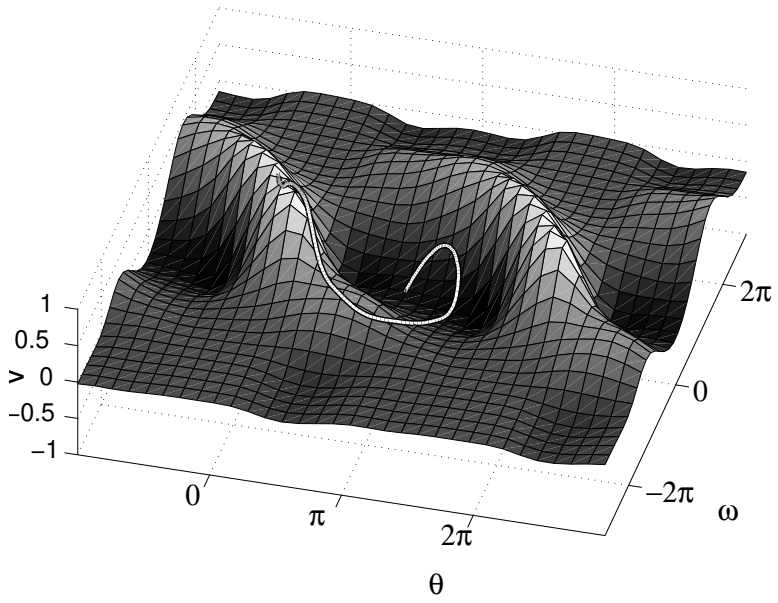

Figure 3: The landscape of the value function $V(\theta, \omega)$ for the pendulum swing-up task. The white line shows an example of a swing-up trajectory. The state space was a cylinder with $\theta = \pm\pi$ connected. The 15×15 centers of normalized Gaussian basis functions are located on a uniform grid that covers the area $[-\pi, \pi] \times [-5/4\pi, 5/4\pi]$.

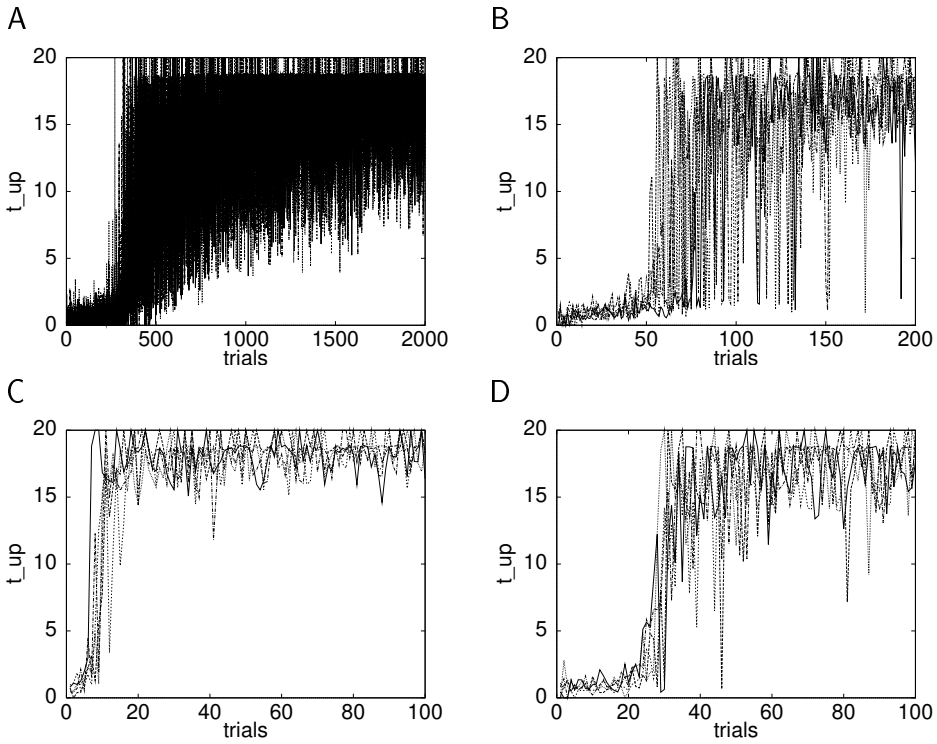

Figure 4: Comparison of the time course of learning with different control schemes: (A) discrete actor-critic, (B) continuous actor-critic, (C) value-gradient based policy with an exact model, (D) value-gradient policy with a learned model (note the different scales). t_{up}: time in which the pendulum stayed up. In the discrete actor-critic, the state space was evenly discretized into 30×30 boxes and the action was binary ($u = \pm u^{\max}$). The learning parameters were $\gamma = 0.98$, $\lambda = 0.8$, $\eta = 1.0$, and $\eta^A = 0.1$.

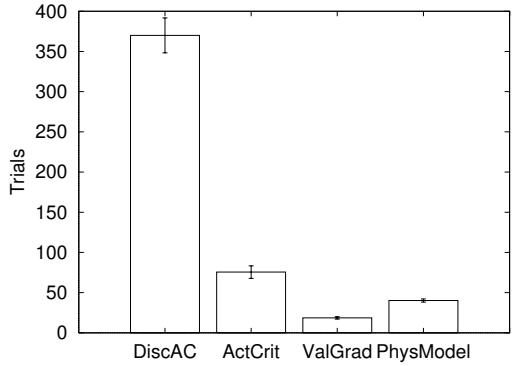

Figure 5: Comparison of learning speeds with discrete and continuous actor-critic and value-gradient based policies with an exact and learned physical models. The ordinate is the number of trials made until ten successful swing-ups.

continuous algorithms performed well with 15×15 basis functions, the discrete algorithm did not achieve the task using 15×15, 20×20, or 25×25 grids. The result shown here was obtained by 30×30 grid discretization of the state.

Among the continuous algorithms, the learning was fastest with the value-gradient based policy using an exact input gain. Concurrent learning of the input gain model resulted in slower learning. The actor-critic was the slowest. This was due to more effective exploitation of the value function in the gradient based policy (24) compared to the stochastically improved policy (19) in the actor-critic.

Methods of Value Function Update

Next, we compared the methods of value function update algorithms (14), (15), and (17) using the greedy policy with an exact gain model (Figure 6). Although the three algorithms attained comparable performances with the optimal settings for Δt and κ, the method with the exponential eligibility trace performed well in the widest range of the time constant and the learning rate. We also tested the purely symmetric update method (12), but its performance was very unstable even when the learning rates for the value and its gradient was tuned carefully.

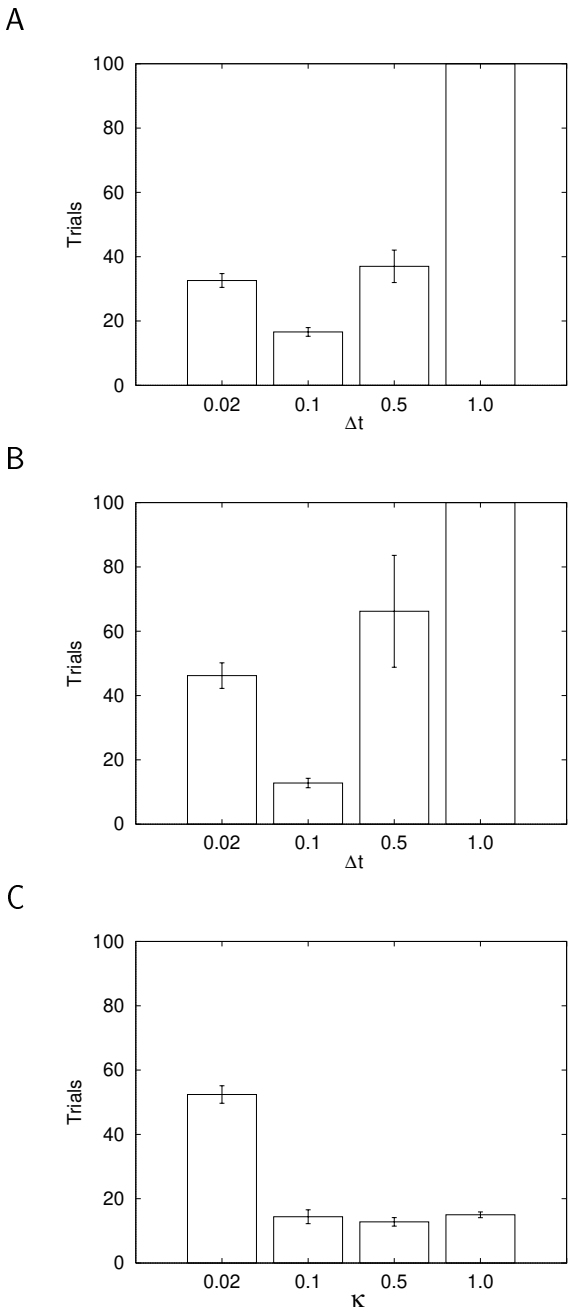

Figure 6: Comparison of different value function update methods with different settings for the time constants. (A) Residual gradient (Eq. 14). (B) Single step eligibility trace (Eq. 15). (C) Exponential eligibility (Eq. 17). The learning rate η was roughly optimized for each method and each setting of time step Δt of the Euler approximation or time constant κ of the eligibility trace. The performance was very unstable with $\Delta t = 1.0$ in the discretization-based methods.

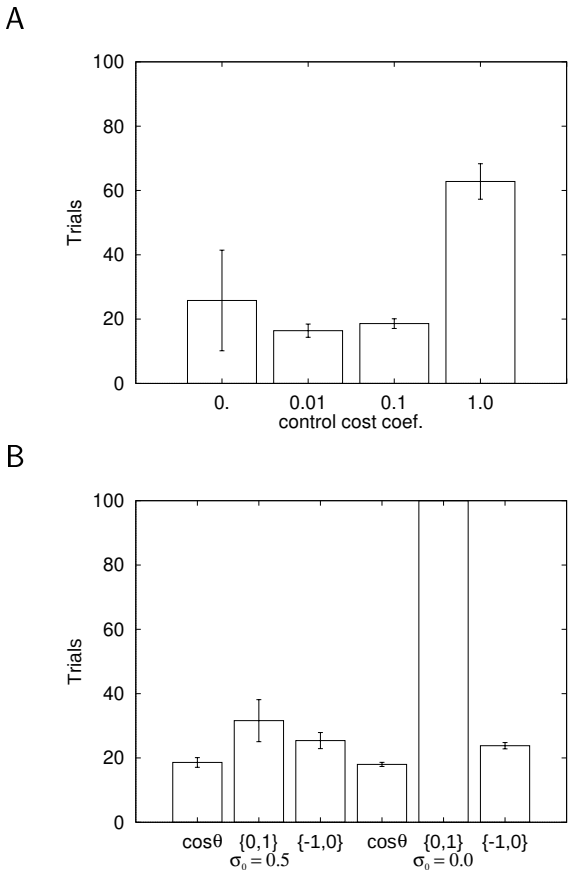

Figure 7: Effects of parameters of the policy. (A) Control cost coefficient c. (B) Reward function $R(\mathbf{x})$ and perturbation size σ_0.

Action Cost, Graded Reward, and Exploration

We then tested how the performance depended on the action cost c, the shape of the reward function $R(\mathbf{x})$, and the size of the exploration noise σ_0. Figure 7(A) compares the performance with different action costs c=0, 0.01, 0.1, and 1.0. The learning was slower with a large cost for the torque ($c = 1$) because of the weak output in the early stage. The results with bang-bang control ($c = 0$) tended to be less consistent than with sigmoid control with the small costs ($c = 0.01, 0.1$).

Figure 7(B) summarizes the effects of the reward function and exploration noise. When binary reward function

$$R(\mathbf{x}) = \begin{cases} 1 & |\theta| < \pi/4 \\ 0 & \text{otherwise} \end{cases}$$

was used instead of $\cos(\theta)$, the task was more difficult to learn. However, a better perfor-

mance was observed with the use of negative binary reward function

$$R(\mathbf{x}) = \begin{cases} 0 & |\theta| < \pi/4 \\ -1 & \text{otherwise.} \end{cases}$$

The difference was more drastic with a fixed initial state $\mathbf{x}(0) = (\pi, 0)$ and no noise $\sigma = 0$, for which no success was achieved with the positive binary reward. The better performance with the negative reward was due to the initialization of the value function as $V(\mathbf{x}) = 0$. As the value function near $\theta = \pi$ is learned as $V(\mathbf{x}) \simeq -1$, the value-gradient based policy drives the state to unexplored areas which are assigned higher values $V(\mathbf{x}) \simeq 0$ by default.

5.2 Cart-Pole Swing-Up Task

Next, we tested the learning schemes in a more challenging task of cart-pole swing-up (Figure 8), which is a strongly nonlinear extension to the common cart-pole balancing task (Barto et al., 1983). The physical parameters of the cart-pole were the same as in (Barto et al., 1983), but the pole had to be swung up from an arbitrary angle and balanced. Marked differences from the previous task were that the dimension of the state space was higher and the input gain was state dependent.

The state vector was $\mathbf{x} = (x, v, \theta, \omega)$, where x and v are the position and the velocity of the cart. The value and policy functions were implemented by normalized Gaussian networks with $7 \times 7 \times 15 \times 15$ bases. A $2 \times 2 \times 4 \times 2 \times 2$ basis network was used for modeling the system dynamics. The reward was given by $R(\mathbf{x}) = \frac{\cos\theta - 1}{2}$.

When the cart bumped into the end of the track or when the pole over-rotated ($|\theta| > 5\pi$), a terminal reward $r(t) = 1$ was given for 0.5 second. Otherwise, a trial lasted for 20 seconds. Figure 8 illustrates the control performance after 1000 learning trials with the greedy policy using the value gradient and a learned input gain model.

Figure 9(A) shows the value function in the 4D state space $\mathbf{x} = (x, v, \theta, \omega)$. Each of the 3×3 squares represents a subspace (θ, ω) with different values of (x, v). We can see the ∞-shaped ridge of the value function, which is similar to the one seen in the pendulum swing-up task (Figure 3). Also note the lower values with x and v both positive or negative, which signal the danger of bumping into the end of the track.

Figure 9(B) shows the most critical components of the input gain vector $\frac{\partial \dot{\omega}}{\partial u}$. The gain represents how the force applied to the cart is transformed as the torque of the pole. The gain model could successfully capture the change of the sign with the upward ($|\theta| < \pi/2$) orientation and the downward ($|\theta| > \pi/2$) orientation of the pole.

Figure 10 is a comparison of the number of trials necessary for 100 successful swing-ups. The value gradient based greedy policies performed about three times faster than the actor-critic. The performances with the exact and learned input gains were comparable in this

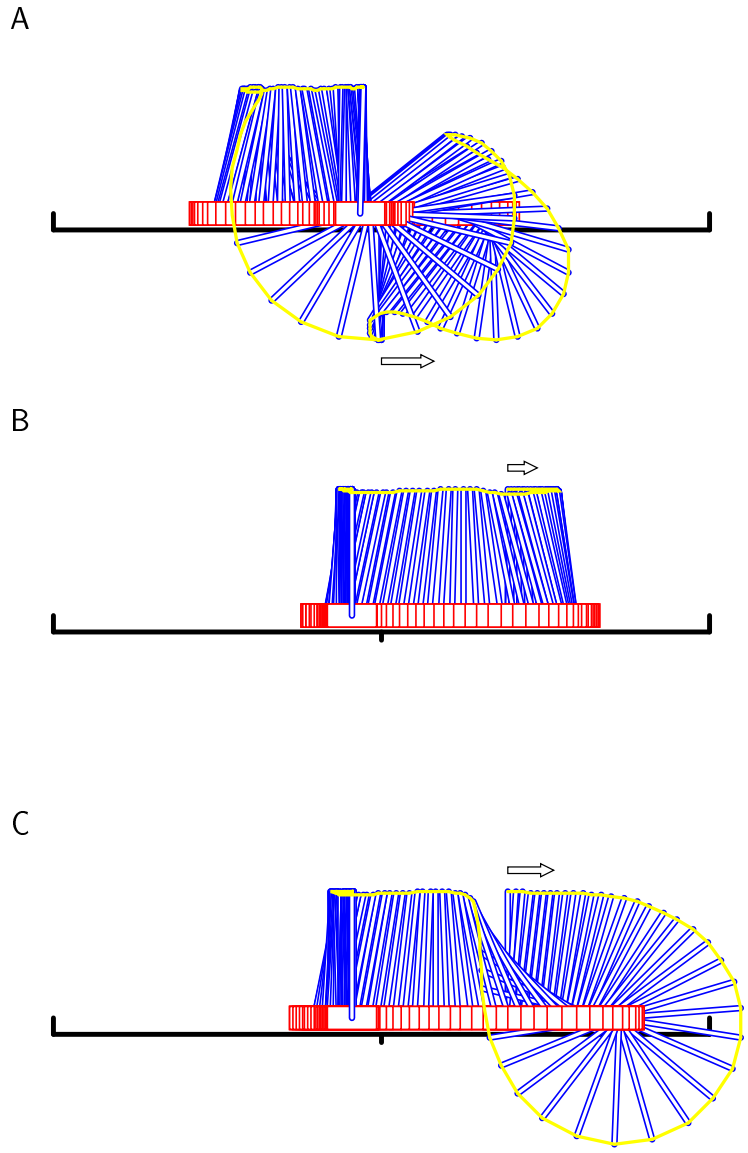

Figure 8: Examples of cart-pole swing up trajectories. The arrows indicate the initial position of the pole. (A) A typical swing up from the bottom position. (B) When a small perturbation $\omega > 0$ is given, the cart moves to the right and keeps the pole upright. (C) When a larger perturbation is given, the cart initially tries to keep the pole upright, but then brakes to avoid collision with the end of the track and swings the pole up on the left side. The learning parameters were $\tau = 1.0$, $\kappa = 0.5$, $c = 0.01$, $\tau_n = 0.5$, $\sigma_0 = 0.5$, $V_0 = 0$, $V_1 = 1$, $\eta = 5.0$, $\eta^A = 10.0$, and $\eta^M = 10.0$.

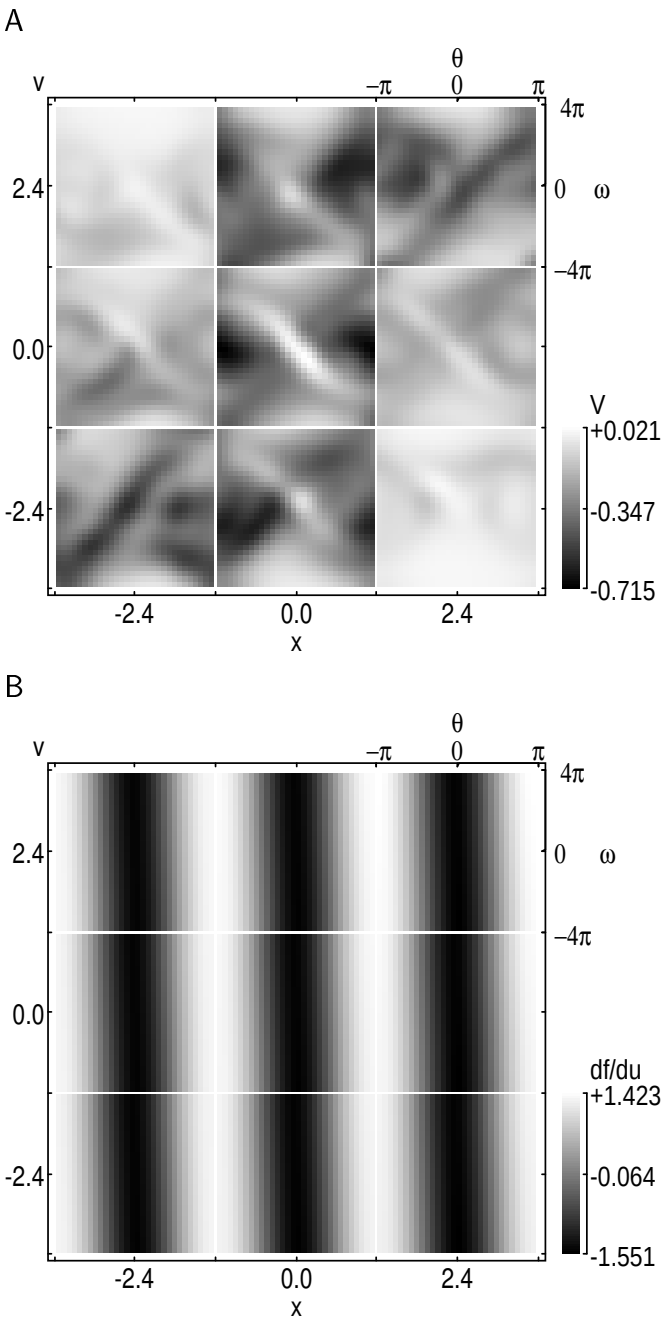

Figure 9: (A) The landscape of the value function for the cart-pole swing-up task. (B) The learned gain model $\frac{\partial \dot{\omega}}{\partial u}$.

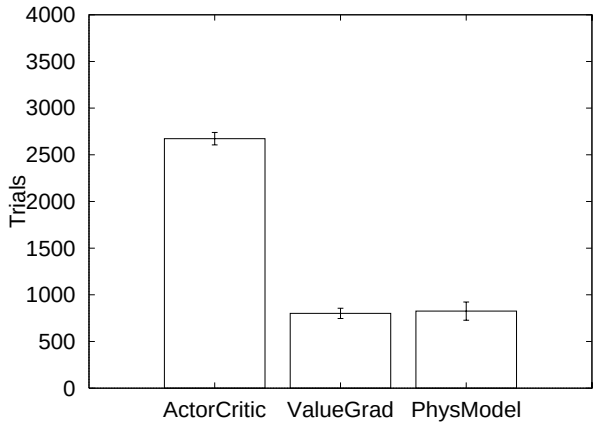

Figure 10: Comparison of the number of trials until 100 successful swing-ups with the actor-critic, value-gradient based policy with an exact and learned physical models.

case. This was because the learning of the physical model was relatively easy compared to the learning of the value function.

6 Discussion

The results of the above simulations can be summarized as follows. 1) The swing-up task was accomplished by the continuous actor-critic in a number of trials several times fewer than by the conventional discrete actor-critic (Figures 4 and 5). 2) Among the continuous methods, the value-gradient based policy with a known or learned dynamic model performed significantly better than the actor-critic (Figures 4, 5, and 10). 3) The value function update methods using exponential eligibility traces was more efficient and stable than the methods based on Euler approximation (Figure 6). 4) Reward related parameters, such as the landscape and the baseline level of the reward function, greatly affect the speed of learning (Figure 7). 5) The value-gradient based method worked well even when the input gain was state dependent (Figures 9 and 10)

Among the three major RL methods, namely, the actor critic, Q-learning, and model-based look ahead, only the Q-learning has been extended to continuous-time cases as advantage updating (Baird, 1993). This paper presents continuous-time counterparts for all three of the methods based on HJB equation (6) and therefore provides a more complete repertoire of continuous RL methods. A major contribution of this paper is the derivation of the closed-form policy (22) using the value gradient and the dynamic model. One critical issue in advantage updating is the need for finding the maximum of the advantage function

on every control cycle, which can be computationally expensive except in special cases like linear quadratic problems (Harmon et al., 1996). As illustrated by simulation, the value gradient based policy (22) can be applied to a broad class of physical control problems using *a priori* or learned models of the system dynamics.

The usefulness of value gradients in RL was considered by Werbos (1990) for discrete-time cases. The use of value gradients was also proposed by Dayan and Singh (1996) with the motivation being to eliminate the need for updating the value function in advantage updating. Their method, however, in which the value gradients $\frac{\partial V}{\partial \mathbf{x}}$ are updated without updating the value function $V(\mathbf{x})$ itself, is applicable only to non-discounted problems.

When the system or the policy is stochastic, HJB equation (6) will include second-order partial derivatives of the value function

$$\frac{1}{\tau}V^*(\mathbf{x}(t)) = \max_{\mathbf{u}(t)\in U}\left[r(\mathbf{x}(t),\mathbf{u}(t)) + \frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}}f(\mathbf{x}(t),\mathbf{u}(t)) + \mathrm{tr}\left(\frac{\partial^2 V^*(\mathbf{x})}{\partial \mathbf{x}^2}C\right)\right], \qquad (28)$$

where C is the covariance matrix of the system noise (see, e.g., (Fleming and Soner, 1993)).

In our simulations, the methods based on deterministic HJB equation (6) worked well although we incorporated noise terms in the policies to promote exploration. One reason for this is that the noise was small enough so that the contribution of the second-order term was minor. Another reason could be that the second-order term has a smoothing effect on the value function and this was implicitly achieved by the use of the smooth function approximator. This point needs further investigation.

The convergent properties of HJB-based RL algorithms were recently shown for deterministic (Munos, 1997) and stochastic (Munos and Bourgine, 1998) cases using a grid-based discretization of space and time. However, the convergent properties of continuous RL algorithms combined with function approximators remain to be studied. When a continuous RL algorithm is numerically implemented with a finite time step, as shown in sections 3.2 and 3.3, it becomes equivalent to a discrete-time TD algorithm, for which some convergent properties have been shown with the use of function approximators (Gordon, 1995; Tsitsiklis and Van Roy, 1997). For example, the convergence of TD algorithms has been shown with the use of a linear function approximator and on-line sampling (Tsitsiklis and Van Roy, 1997), which was the case with our simulations.

However, the above result only considers value function approximation for a given policy and does not guarantee the convergence of the entire RL process to a satisfactory solution. For example, in our swing-up tasks, the learning sometimes got stuck in a locally optimal solution of endless rotation of the pendulum when a penalty for over-rotation was not given.

The use of fixed smooth basis functions has a limitation in that steep cliffs in the value or policy functions cannot be achieved. Despite some negative didactic examples (Tsitsiklis and Van Roy, 1997), methods that dynamically allocate or reshape basis functions have been

successfully used with continuous RL algorithms, for example, in a swing-up task (Schaal, 1997) and in a stand-up task for a three-link robot (Morimoto and Doya, 1998). Elucidation of the conditions under which the proposed continuous RL algorithms work successfully, for example, the properties of the function approximators and the methods for exploration, remains the subject of future empirical and theoretical studies.

Acknowledgments

I thank Mitsuo Kawato, Stefan Schaal, and Chris Atkeson for their helpful discussions.

Appendix A: HJB Equation for Discounted Reward

According to the optimality principle, we divide the integral in (5) into two parts $[t, t+\Delta t]$ and $[t+\Delta t, \infty)$ and then solve a short-term optimization problem

$$V^*(\mathbf{x}(t)) = \max_{\mathbf{u}[t,t+\Delta t]} \left[\int_t^{t+\Delta t} e^{-\frac{s-t}{\tau}} r(\mathbf{x}(s), \mathbf{u}(s)) ds + e^{-\frac{\Delta t}{\tau}} V^*(\mathbf{x}(t+\Delta t)) \right]. \tag{29}$$

For a small Δt, the first term is approximated as

$$r(\mathbf{x}(t), \mathbf{u}(t))\Delta t + o(\Delta t)$$

and the second term is Taylor expanded as

$$V^*(\mathbf{x}(t+\Delta t)) = V^*(\mathbf{x}(t)) + \frac{\partial V^*}{\partial \mathbf{x}(t)} f(\mathbf{x}(t), \mathbf{u}(t))\Delta t + o(\Delta t).$$

By substituting them into (29) and collecting $V^*(\mathbf{x}(t))$ on the left-hand side, we have an optimality condition for $[t, t+\Delta t]$ as

$$(1 - e^{-\frac{\Delta t}{\tau}}) V^*(\mathbf{x}(t)) = \max_{\mathbf{u}[t,t+\Delta t]} \left[r(\mathbf{x}(t), \mathbf{u}(t))\Delta t + e^{-\frac{\Delta t}{\tau}} \frac{\partial V^*}{\partial \mathbf{x}(t)} f(\mathbf{x}(t), \mathbf{u}(t))\Delta t + o(\Delta t) \right]. \tag{30}$$

By dividing both sides by Δt and taking Δt to zero, we have the condition for the optimal value function

$$\frac{1}{\tau} V^*(\mathbf{x}(t)) = \max_{\mathbf{u}(t) \in U} \left[r(\mathbf{x}(t), \mathbf{u}(t)) + \frac{\partial V^*}{\partial \mathbf{x}} f(\mathbf{x}(t), \mathbf{u}(t)) \right]. \tag{31}$$

Appendix B: Normalized Gaussian Network

A value function is represented by

$$V(\mathbf{x}; \mathbf{w}) = \sum_{k=1}^{K} w_k b_k(\mathbf{x}) \tag{32}$$

where

$$b_k(\mathbf{x}) = \frac{a_k(\mathbf{x})}{\sum_{l=1}^{K} a_l(\mathbf{x})}, \qquad a_k(\mathbf{x}) = e^{-||\mathbf{s}_k^T(\mathbf{x}-\mathbf{c}_k)||^2}.$$

The vectors $\mathbf{c}_k$ and $\mathbf{s}_k$ define the center and the size of the k-th basis function. Note that the basis functions located on the ends of the grids are extended like sigmoid functions by the effect of normalization.

In the current simulations, the centers are fixed in a grid, which is analogous to the "boxes" approach (Barto et al., 1983) often used in discrete RL. Grid allocation of the basis functions enables efficient calculation of their activation as the outer product of the activation vectors for individual input variables.

In the actor-critic method, the policy is implemented as

$$\mathbf{u}(t) = \mathbf{u}^{\max} s\left(\sum_k w_k^A b_k(\mathbf{x}(t)) + \sigma \mathbf{n}(t)\right), \tag{33}$$

where s is a component-wise sigmoid function and $\mathbf{n}(t)$ is the noise.

In the value gradient-based methods, the policy is given by

$$\mathbf{u}(t) = \mathbf{u}^{\max} s\left(\frac{1}{c}\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}^T \sum_k w_k \frac{\partial b_k(\mathbf{x})}{\partial \mathbf{x}}^T + \sigma \mathbf{n}(t)\right). \tag{34}$$

To implement the input gain model, a network is trained to predict the time derivative of the state from $\mathbf{x}$ and $\mathbf{u}$

$$\dot{\mathbf{x}}(t) \simeq \hat{f}(\mathbf{x},\mathbf{u}) = \sum_k w_k^M b_k(\mathbf{x}(t),\mathbf{u}(t)). \tag{35}$$

The weights are updated by

$$\dot{w}_k^M(t) = \eta^M(\dot{\mathbf{x}}(t) - \hat{f}(\mathbf{x}(t),\mathbf{u}(t)))b_k(\mathbf{x}(t),\mathbf{u}(t)), \tag{36}$$

and the input gain of the system dynamics is given by

$$\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}} \simeq \sum_k w_k^M \left.\frac{\partial b_k(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}=0}. \tag{37}$$

References

Asada, M., Noda, S., and Hosoda, K. (1996). Action-based sensor space categorization for robot learning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1502–1509.

Atkeson, C. G. (1994). Using local trajectory optimizers to speed up global optimization in dynamic programming. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing System*, volume 6, pages 663–670. Morgan Kaufmann, San Mateo, CA, USA.

Baird, L. C. (1993). Advantage updating. Technical Report WL-TR-93-1146, Wright Laboratory, Wright-Patterson Air Force Base, OH 45433-7301, USA.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In Prieditis, A. and Russel, S., editors, *Machine Learning: Proceedings of the Twelfth International Conference*, San Francisco. Morgan Kaufmann.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846.

Bradtke, S. J. (1993). Reinforcement learning applied to linear quadratic regulation. In Giles, C. L., Hanson, S. J., and Cowan, J. D., editors, *Advances in Neural Information Processing Systems 5*, pages 295–302. Morgan Kaufmann, San Mateo, CA, USA.

Bradtke, S. J. and Duff, M. O. (1995). Reinforcement learning methods for continuous-time Markov decision problems. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 393–400. MIT Press, Cambridge, MA, USA.

Crites, R. H. and Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 1017–1023. MIT Press, Cambridge, MA.

Dayan, P. and Singh, S. P. (1996). Improving policies without measuring merits. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 1059–1065. MIT Press, Cambridge, MA, USA.

Doya, K. (1996). Temporal difference learning in continuous time and space. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 1073–1079. MIT Press.

Doya, K. (1997). Efficient nonlinear control with actor-tutor architecture. In M. C. Mozer, M. I. Jordan, T. P., editor, *Advances in Neural Information Processing Systems 9*, pages 1012–1018. MIT Press.

Fleming, W. H. and Soner, H. M. (1993). *Controlled Markov Processes and Viscosity Solutions*. Applications of Mathematics. Springer-Verlag, New York.

Gordon, G. J. (1995). Stable function approximation in dynamic programming. In Prieditis, A. and Russel, S., editors, *Machine Learning: Proceedings of the Twelfth International Conference*. Morgan Kaufmann, San Francisco.

Gordon, G. J. (1996). Stable fitted reinforcement learning. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 1052–1058. MIT Press, Cambridge, MA, USA.

Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3:671–192.

Harmon, M. E., Baird, III, L. C., and Klopf, A. H. (1996). Reinforcement learning applied to a differential game. *Adaptive Behavior*, 4:3–28.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of National Academy of Science*, 81:3088–3092.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Mataric, M. J. (1994). Reward functions for accelerated learning. In Cohen, W. W. and Hirsh, H., editors, *Proceedings of the 11th International Conference on Machine Learning*. Morgan Kaufmann.

Moore, A. W. (1994). The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 711–718. Morgan Kaufmann, San Francisco, CA, USA.

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

Munos, R. (1997). A convergent reinforcement learning algorithm in the continuous case based on a finite difference method. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 826–831.

Munos, R. and Bourgine, P. (1998). Reinforcement learning for continuous stochastic control problems. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, *Advances in Neural Information Processing Systems 10*, pages 1029–1035, Cambridge, MA, USA. MIT Press.

Pareigis, S. (1998). Adaptive choice of grid and time in reinforcement learning. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, *Advances in Neural Information Processing Systems 10*, pages 1036–1042, Cambridge, MA, USA. MIT Press.

Peterson, J. K. (1993). On-line estimation of the optimal value function: HJB-estimators. In Giles, C. L., Hanson, S. J., and Cowan, J. D., editors, *Advances in Neural Information Processing Systems 5*, pages 319–326. Morgan Kaufmann, San Mateo, CA, USA.

Schaal, S. (1997). Learning from demonstration. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 1040–1046, Cambridge, MA, USA. MIT Press.

Singh, S. and Bertsekas, D. (1997). Reinforcement learning for dynamic channel allocation in cellular telephone systems. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 974–980. MIT Press, Cambridge, MA, USA.

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1995). Reinforcement learning with soft state aggregation. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 361–368. MIT Press, Cambridge, MA, USA.

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

Sutton, R. S. (1995). TD models: Modeling the world at a mixture of time scales. In *Proceedings of the 12th International Conference on Machine Learning*, pages 531–539. Morgan Kaufmann.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, Cambridge, MA, USA.

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

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

Tsitsiklis, J. N. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 1075–1081, Cambridge, MA, USA. MIT Press.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, Cambridge University.

Werbos, P. J. (1990). A menu of designs for reinforcement learning over time. In Miller, W. T., Sutton, R. S., and Werbos, P. J., editors, *Neural Networks for Control*, pages 67–95. MIT Press, Cambridge, MA.

Zhang, W. and Dietterich, T. G. (1996). High-performance job-shop scheduling with a time-delay TD(λ) network. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge, MA, USA.