*LETTER* **Communicated by Andrew Barto**

# Robust Reinforcement Learning

**Jun Morimoto**
*xmorimo@atr.jp*
*Computational Brain Project, ICORP, JST, Sora Ku-gun, Kyoto, 619-0288, Japan; ATR Computational Neuroscience Laboratories, Soraku-gun, Kyoto 619-0288, Japan*

**Kenji Doya**
*doya@irp.oist.jp*
*ATR Computational Neuroscience Laboratories, Soraku-gun, Kyoto 619-0288, Japan; Initial Research Project, OIST, Gushikawa, Okinawa, 904-2234, Japan; and CREST, JST, Soraku-gun, Kyoto 619-0288, Japan*

**This letter proposes a new reinforcement learning (RL) paradigm that explicitly takes into account input disturbance as well as modeling errors. The use of environmental models in RL is quite popular for both off-line learning using simulations and for online action planning. However, the difference between the model and the real environment can lead to unpredictable, and often unwanted, results. Based on the theory of $H^\infty$ control, we consider a differential game in which a "disturbing" agent tries to make the worst possible disturbance while a "control" agent tries to make the best control input. The problem is formulated as finding a min-max solution of a value function that takes into account the amount of the reward and the norm of the disturbance. We derive online learning algorithms for estimating the value function and for calculating the worst disturbance and the best control in reference to the value function. We tested the paradigm, which we call robust reinforcement learning (RRL), on the control task of an inverted pendulum. In the linear domain, the policy and the value function learned by online algorithms coincided with those derived analytically by the linear $H^\infty$ control theory. For a fully nonlinear swing-up task, RRL achieved robust performance with changes in the pendulum weight and friction, while a standard reinforcement learning algorithm could not deal with these changes. We also applied RRL to the cart-pole swing-up task, and a robust swing-up policy was acquired.**

## 1 Introduction

In this letter, we propose a new reinforcement learning paradigm that we call robust reinforcement learning (RRL). Plain, model-free reinforcement learning (RL) requires a significant number of trials to learn policies for given tasks, and therefore it is difficult to use for online learning of real-

world problems. Thus, the use of environmental models has been quite common for both online action planning (Doya, 2000) and off-line learning by simulation (Morimoto & Doya, 2000). However, no model is perfect, and modeling errors can cause unpredictable results—sometimes worse than with no model at all. In fact, robustness against model uncertainty has been the main subject of research in the control community for the past 20 years, and the result is formalized as the $H^\infty$ control theory (Zhou, Doyle & Glover, 1996).

In general, a modeling error causes a deviation of the real system state from the state predicted by the model. This can be reinterpreted as a disturbance to the model. However, the problem is that the disturbances due to modeling error can be strongly correlated, and thus standard independence assumptions may not be valid. The basic strategy to achieve robustness is to keep the sensitivity $\gamma$ of the feedback control loop against a disturbance input small enough so that any disturbance due to modeling error can be suppressed if the gain of mapping from the state error to the disturbance is bounded by $1/\gamma$. In the $H^\infty$ paradigm, those disturbance-to-error and error-to-disturbance gains are measured by the max norms of the functional mappings in order to ensure stability for any mode of disturbance.

In section 2, we briefly introduce the $H^\infty$ paradigm and show that design of a robust controller can be achieved by finding a min-max solution of a value function, which is given by the Hamilton-Jacobi-Isaacs (HJI) equation. In section 3, we formulate robust reinforcement learning. We first define an augmented value function that takes into account the amplitude of the disturbance and derive a corresponding HJI equation. We then derive an online algorithm for estimating the value function, the worst-case disturbances, and the optimal controller. In section 4, we test the validity of the algorithms first in a linear inverted pendulum task. It is verified that the value function as well as the disturbance and control policies derived by the online algorithms coincide with the analytical solution given by $H^\infty$ theory. We then compare the performance of the robust reinforcement learning algorithm with standard model-based RL in the nonlinear task of pendulum swing-up (Doya, 2000). It is shown that a robust RL controller can accommodate changes in physical parameters of the pendulum that a standard RL controller cannot cope with (Morimoto & Doya, 2001b). We also applied robust RL to the cart pole swing-up task (Doya, 2000) and tested the robustness of the learned controller.

## 2 $H^\infty$ Control

Standard $H^\infty$ control (Zhou et al., 1996) deals with a system shown in Figure 1, where $G$ is the plant, $K$ is the controller, $\mathbf{u}$ is the control input, $\mathbf{y}$ is the measurement available to the controller, $\mathbf{w}$ is an unknown disturbance, and $\mathbf{z}$ is the error output that is desired to be kept small. In general, the controller $K$ is designed to stabilize the closed-loop system based on a model of the plant

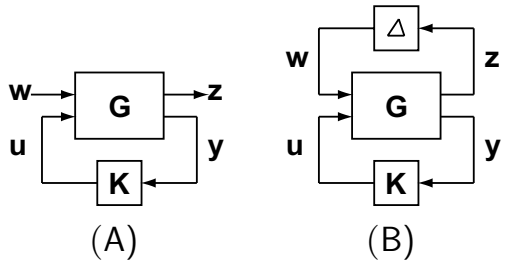



Figure 1: (A) Generalized plant $G$ and controller $K$. (B) Small gain theorem.

$G$. However, when there is a discrepancy between the model and the actual plant dynamics, the feedback loop could be unstable. The effect of modeling error can be equivalently represented as a disturbance $\mathbf{w}$ generated by an unknown mapping $\Delta$ of the plant output $\mathbf{z}$, as shown in Figure 1b.

The goal of the $H^\infty$ control problem is to design a controller $K$ that brings the error $\mathbf{z}$ to zero while minimizing the $H^\infty$ norm of the closed-loop transfer function $T_{zw}$ from the disturbance $\mathbf{w}$ to the output $\mathbf{z}$

$$\|T_{zw}\|_\infty = \sup_{\mathbf{w}} \frac{\|\mathbf{z}\|_2}{\|\mathbf{w}\|_2}, \tag{2.1}$$

where $\| \bullet \|_2$ denotes $L_2$ norm. The small gain theorem ensures that if $\|T_{zw}\|_\infty \leq \gamma$, then the system shown in Figure 1b will be stable for any stable mapping $\Delta$: $\mathbf{z} \mapsto \mathbf{w}$ with $\|\Delta\|_\infty < \frac{1}{\gamma}$ (Zhou et al., 1996).

**2.1 Min-Max Solution to $H^\infty$ Problem.** We consider the plant $G$ with dynamics given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}), \tag{2.2}$$

where $\mathbf{x} \in X \subset \mathbf{R}^n$ is the state, $\mathbf{u} \in U \subset \mathbf{R}^m$ is the control input, and $\mathbf{w} \in W \subset \mathbf{R}^l$ is the disturbance input.

From equation 2.1 and the small gain theorem, the $H^\infty$ control problem can be considered as the problem to find a controller that satisfies a constraint

$$\|T_{zw}\|_\infty{}^2 = \sup_{\mathbf{w}} \frac{\|\mathbf{z}\|_2{}^2}{\|\mathbf{w}\|_2{}^2} \leq \gamma^2, \tag{2.3}$$

where $\mathbf{z}$ is the error output. In other words, because the norm $\|\mathbf{z}\|_2$ and the norm $\|\mathbf{w}\|_2$ are defined as

$$\|\mathbf{z}\|_2{}^2 = \int_0^\infty \mathbf{z}^T(t)\mathbf{z}(t)\, dt \tag{2.4}$$

$$\|\mathbf{w}\|_2{}^2 = \int_0^\infty \mathbf{w}^T(t)\mathbf{w}(t)\, dt, \tag{2.5}$$

the $H^\infty$ control problem is equivalent to finding a control input $\mathbf{u}$ that satisfies a constraint

$$V = \int_0^\infty (\mathbf{z}^T(t)\mathbf{z}(t) - \gamma^2 \mathbf{w}^T(t)\mathbf{w}(t))\, dt \leq 0 \tag{2.6}$$

against all possible disturbances $\mathbf{w}$ with $\mathrm{x}(0) = \mathbf{0}$.

We can consider this problem as a differential game (Weiland, 1989) in which the best control output $\mathbf{u}$ that minimizes $V$ is sought while the worst disturbance $\mathbf{w}$ that maximizes $V$ is chosen. Thus, an optimal value function $V^*$ is defined as

$$V^* = \min_{\mathbf{u}} \max_{\mathbf{w}} \int_0^\infty (\mathbf{z}^T(t)\mathbf{z}(t) - \gamma^2 \mathbf{w}^T(t)\mathbf{w}(t))\, dt. \tag{2.7}$$

The condition for the optimal value function is given by

$$0 = \min_{\mathbf{u}} \max_{\mathbf{w}} \left[ \mathbf{z}^T\mathbf{z} - \gamma^2 \mathbf{w}^T\mathbf{w} + \frac{\partial V^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}, \mathbf{w}) \right], \tag{2.8}$$

which is known as the Hamilton-Jacobi-Isaacs (HJI) equation. From equation 2.8, we can derive the optimal control output $\mathbf{u}$ and the worst disturbance $\mathbf{w}$ by solving

$$\frac{\partial \mathbf{z}^T\mathbf{z}}{\partial \mathbf{u}} + \frac{\partial V^*}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{u}} = 0 \tag{2.9}$$

$$-2\gamma^2 \mathbf{w} + \frac{\partial V^*}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}} = 0. \tag{2.10}$$

## 3 Robust Reinforcement Learning

Here we consider a continuous-time formulation of reinforcement learning (Doya, 2000) with the system dynamics

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}), \tag{3.1}$$

and the reward $r(\mathbf{x}, \mathbf{u})$, where $\mathbf{x} \in X \subset \mathbf{R}^n$ is the state, $\mathbf{u} \in U \subset \mathbf{R}^m$ is the control output, and $\mathbf{w} \in W \subset \mathbf{R}^l$ is the disturbance input.

The basic goal is to find a policy $\mathbf{u} = g(\mathbf{x})$ that maximizes the cumulative future reward,

$$\int_t^\infty e^{-\frac{s-t}{\tau}} r(\mathbf{x}(s), \mathbf{u}(s))\, ds, \tag{3.2}$$

for any given state $\mathbf{x}(t)$, where $\tau$ is a time constant of evaluation. However, a particular policy that was optimized for a certain environment may perform

badly when the environmental setting changes. In order to ensure robust performance under a changing environment or unknown disturbance, we introduce the notion of worst disturbance in $H^\infty$ control to the reinforcement learning paradigm.

In this framework, we consider an augmented reward,

$$q(t) = r(\mathbf{x}(t), \mathbf{u}(t)) + \omega(\mathbf{w}(t)), \tag{3.3}$$

where $\omega(\mathbf{w}(t))$ is an additional reward for withstanding a disturbing input; for example, we can use quadratic cost for the disturbance input to be consistent with equation 2.7,

$$\omega(\mathbf{w}) = \gamma^2 \mathbf{w}^T \mathbf{w}, \tag{3.4}$$

where $\gamma$ is the robustness parameter. The augmented value function is then defined as

$$V(\mathbf{x}(t)) = \int_t^\infty e^{-\frac{s-t}{\tau}} q(\mathbf{x}(s), \mathbf{u}(s), \mathbf{w}(s))\, ds. \tag{3.5}$$

The optimal value function is given by the solution of a variant of HJI equation,

$$\frac{1}{\tau} V^*(\mathbf{x}) = \max_{\mathbf{u}} \min_{\mathbf{w}} \left[ r(\mathbf{x}, \mathbf{u}) + \omega(\mathbf{w}) + \frac{\partial V^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}, \mathbf{w}) \right]. \tag{3.6}$$

In the RRL paradigm, the value function is updated by using the temporal difference (TD) error (Doya, 2000),

$$\delta(t) = q(t) - \frac{1}{\tau} V(t) + \dot{V}(t), \tag{3.7}$$

while the best action and the worst disturbance are generated by maximizing and minimizing, respectively, the right-hand side of the HJI equation,

$$r(\mathbf{x}, \mathbf{u}) + \omega(\mathbf{w}) + \frac{\partial V^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}, \mathbf{w}). \tag{3.8}$$

We use a function approximator to implement the value function $V(x(t); \mathbf{v})$, where $\mathbf{v} = (v_1, \ldots, v_n)$ is a parameter vector. As in the standard continuous-time RL, we use the learning rule for the value function appoximator as

$$\dot{v}_i = \eta \delta(t) e_i(t), \tag{3.9}$$

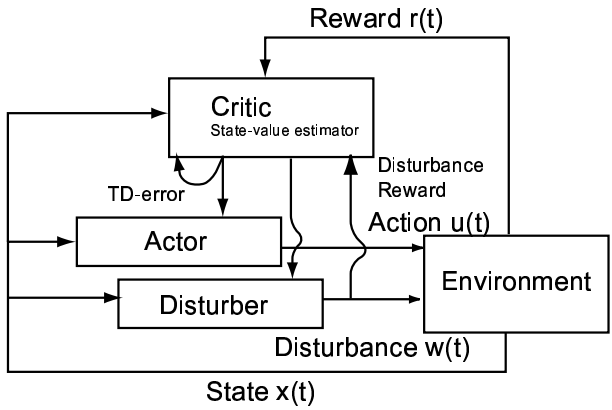



Figure 2: Actor-disturber-critic architecture.

where $\eta$ denotes the learning rate and $e_i$ denotes eligibility trace for a parameter $v_i$ (Doya, 2000). The eligibility trace is updated by

$$\dot{e}_i(t) = -\frac{1}{\kappa} e_i(t) + \frac{\partial V(t)}{\partial v_i}, \tag{3.10}$$

By comparing equations 2.8 and 3.6, we can see that the output error $\mathbf{z}^T\mathbf{z}$ in the $H^\infty$ framework is generalized as an arbitrary reward function $r(\mathbf{x}, \mathbf{u})$ in the RRL. Furthermore, the sign of the value function is flipped, and a discount factor is introduced in the RRL framework.

**3.1 Model-Free Policy: Actor-Disturber-Critic.** To implement robust RL in a model-free fashion, we propose the actor-disturber-critic architecture, shown in Figure 2. We define the policies of the actor and the disturber as

$$\mathbf{u}(t) = s_u(A_u(\mathbf{x}(t); \mathbf{v}^u) + \mathbf{n}^u(t)) \tag{3.11}$$

and

$$\mathbf{w}(t) = s_w(A_w(\mathbf{x}(t); \mathbf{v}^w) + \mathbf{n}^w(t)), \tag{3.12}$$

respectively, where $A_u(\mathbf{x}(t); \mathbf{v}^u) \in \mathbf{R}^m$ and $A_w(\mathbf{x}(t); \mathbf{v}^w) \in \mathbf{R}^l$ are function approximators with parameter vectors $\mathbf{v}^u$ and $\mathbf{v}^w$, and $\mathbf{n}_u(t) \in \mathbf{R}^m$ and $\mathbf{n}_w(t) \in \mathbf{R}^l$ are noise terms for exploration. $s_u()$ and $s_w()$ are monotonically increasing output functions. For example, we can use sigmoidal functions for $s_u()$ and $s_w()$ to saturate control output and disturbance input. In the actor-disturber-critic architecture, policies of the actor and disturber are

represented by the parameter vectors $\mathbf{v}^u$ and $\mathbf{v}^w$. The parameters of the actor and the disturber are updated by

$$\dot{v}_i^u = \eta^u \delta(t) n_i^u(t) \frac{\partial A_u(\mathbf{x}(t); \mathbf{v}^u)}{\partial v_i^u} \tag{3.13}$$

$$\dot{v}_i^w = -\eta^w \delta(t) n_i^w(t) \frac{\partial A_w(\mathbf{x}(t); \mathbf{v}^w)}{\partial v_i^w}, \tag{3.14}$$

where $\eta^u$ and $\eta^w$ denote the learning rates.

**3.2 Model-Based Policy: Using Value Gradient.** When the augmented reward function $q(\mathbf{x}, \mathbf{u}, \mathbf{w})$ in equation (3.3) is convex with respect to the action $\mathbf{u}$ and disturbance $\mathbf{w}$, the HJI equation, 3.6, has a unique solution.

Here, we assume that the augmented reward $q(\mathbf{x}, \mathbf{u}, \mathbf{w})$ can be separated into three parts: the reward for the state $L(\mathbf{x})$, the cost for the action $S(\mathbf{u})$, and the cost for the disturbance $\Omega(\mathbf{w})$. We specifically consider the case

$$q(\mathbf{x}, \mathbf{u}, \mathbf{w}) = L(\mathbf{x}) - \sum_{i=1}^{m} S_i(u_i) + \sum_{j=1}^{l} \Omega_j(w_j), \tag{3.15}$$

where $S_i()$ is a convex cost function for action variable $u_i$ and $\Omega_j()$ is a convex cost function for disturbance variable $w_j$. In this case, the condition for the optimal action and the worst disturbance is given from the HJI equation, 3.6, as

$$-S_i'(u_i) + \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial u_i} = 0 \qquad (i = 1, \ldots, m) \tag{3.16}$$

$$\Omega_j'(w_j) + \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial w_j} = 0 \qquad (j = 1, \ldots, l), \tag{3.17}$$

where $\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial u_i}$ and $\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial w_j}$ are the $i$th and $j$th column vector of the $n \times m$ input gain matrix $\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial \mathbf{u}}$ and the $n \times l$ disturbance gain matrix $\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial \mathbf{w}}$, respectively. We now assume that the input gains $\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial u_i}$ and $\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial w_j}$ are not dependent on $\mathbf{u}$ and $\mathbf{w}$; that is, the system is input-affine. Then the above equations have unique solutions,

$$u_i = S_i'^{-1}\left(\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial u_i}\right) \tag{3.18}$$

$$w_j = \Omega_j'^{-1}\left(-\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial w_j}\right), \tag{3.19}$$

where $S_i'()$ and $\Omega_j'()$ are monotonic functions. Accordingly, greedy policies for action and disturbance are represented in vector notation as

$$\mathbf{u} = S'^{-1}\left(\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{u}}^T \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}^T\right) \tag{3.20}$$

$$\mathbf{w} = \Omega'^{-1}\left(-\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}}^T \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}^T\right), \tag{3.21}$$

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" models $\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial \mathbf{u}}^T$ and $\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial \mathbf{w}}^T$ into a direction in the control output and disturbance input space, respectively (Doya, 2000).

**3.3 Linear-Quadratic Case.** We consider a special case in which a linear dynamic model and quadratic reward models are given or learned as

$$\dot{\mathbf{x}} = A\mathbf{x} + B_1\mathbf{u} + B_2\mathbf{w} \tag{3.22}$$

$$q(\mathbf{x}, \mathbf{u}, \mathbf{w}) = -\mathbf{x}^T Q\mathbf{x} - \mathbf{u}^T R\mathbf{u} + \gamma^2 \mathbf{w}^T \mathbf{w}. \tag{3.23}$$

In this case, the value function is given by a quadratic form $V(\mathbf{x}) = -\mathbf{x}^T P\mathbf{x}$, where $P$ is the solution of a Riccati equation:

$$A^T P + PA + P\left(\frac{1}{\gamma^2} B_1 B_1^T - B_2 R^{-1} B_2^T\right) P + Q = \frac{1}{\tau} P. \tag{3.24}$$

From $\frac{\partial f}{\partial \mathbf{u}} = B_1$, $\frac{\partial f}{\partial \mathbf{w}} = B_2$, $S'(\mathbf{u}) = 2R\mathbf{u}$, $\Omega'(\mathbf{w}) = 2\gamma^2\mathbf{w}$, and $\frac{\partial V}{\partial \mathbf{x}} = -2\mathbf{x}^T P$, the best action $\mathbf{u}$ in equation 3.20 and the worst disturbance $\mathbf{w}$ in equation 3.21 can be derived as

$$\mathbf{u} = R^{-1} B_1^T P\mathbf{x} \tag{3.25}$$

$$\mathbf{w} = -\frac{1}{\gamma^2} B_2^T P\mathbf{x}. \tag{3.26}$$

## 4 Simulation

In this section, we show the performance of robust RL. In section 4.1, we consider the control problem of a linear inverted pendulum (see Figure 10 and appendix B) and test whether the parameters of the linear controller acquired by our robust RL algorithm converge to the analytical $H^\infty$ solution. In section 4.2, we apply robust RL to a nonlinear pendulum swing-up task (see appendix A) and compare the robustness of the controllers learned by robust RL and standard RL. We also test how the performance of the robust

RL controllers change with different settings of the robustness parameter $\gamma$. In section 4.3, we apply robust RL to the cart pole swing-up task (Doya, 2000; see Figure 11 and appendix C). We show that robust RL is applicable to a challenging nonlinear control task with a higher-dimensional state space. (Refer to the appendixes for the outline of the three benchmark tasks.)

Below are the function approximation methods we used for the value functions, the dynamics models, and the policies in the three tasks.

**Value function.** We use normalized gaussian networks as the function approximator to represent the value function

$$V(\mathbf{x}; \mathbf{v}) = \sum_{k=1}^{K} v_k b_k(\mathbf{x}), \tag{4.1}$$

where $b()$ is the normalized gaussian basis function and $\mathbf{v} = (v_1, \ldots, v_k)^T$ is the weight vector (see appendix D). In the case of linear dynamics and the quadratic reward, instead of using the normalized gaussian networks, we approximate the value function by the quadratic form so that

$$V(\mathbf{x}) = -\mathbf{x}^T P \mathbf{x}, \tag{4.2}$$

where $P$ is a symmetric matrix and the parameter of the function approximator. By using this quadratic form, we can exactly represent the actual value function. The update rules of these function approximators are given by equations 3.10 and 3.11.

**Dynamics model.** We approximate the dynamics model without disturbance input by using the function approximator $\hat{f}(\mathbf{x}, \mathbf{u}; \mathbf{v}^M)$,

$$\dot{\mathbf{x}} - \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}} \mathbf{w} \simeq \hat{f}(\mathbf{x}, \mathbf{u}; \mathbf{v}^M), \tag{4.3}$$

where $\mathbf{v}^M$ is a parameter vector. Because we assume that the system is input-affine as described in section 3.2, we can remove the disturbance term from the dynamics $f(\mathbf{x}, \mathbf{u}, \mathbf{w})$ by subtracting $\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}} \mathbf{w}$ from $\dot{\mathbf{x}}$.

Then we can approximate input gain using the approximated model $\hat{f}(\mathbf{x}, \mathbf{u}; \mathbf{v}^M)$ as

$$\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{u}} \simeq \frac{\partial \hat{f}(\mathbf{x}, \mathbf{u}; \mathbf{v}^M)}{\partial \mathbf{u}}. \tag{4.4}$$

For linear dynamics, we use the linear approximation

$$\dot{\mathbf{x}} - B_2 \mathbf{w} \simeq \hat{A}\mathbf{x} + \hat{B}_1 \mathbf{u}, \tag{4.5}$$

where $\hat{A}$ and $\hat{B}_1$ are parameter matrices. The update rule of the function approximator is given in appendix D.

**Actor and disturber.** Actor and disturber are represented by equations 3.12 and 3.13. Corresponding policies to the reward function (see equation A.3 in appendix A) are given by

$$u = \frac{1}{2}R^{-1}(A_u(\mathbf{x}; \mathbf{v}^u) + n^u), \tag{4.6}$$

$$w = \frac{1}{2\gamma^2}(A_w(\mathbf{x}; \mathbf{v}^w) + n^w). \tag{4.7}$$

We use normalized gaussian networks as the function approximator (see appendix D). The update rule of the function approximator is given in equations 3.14 and 3.15. In the linear quadratic case, the actor and the disturber in equations 3.12 and 3.13 are represented as linear controllers, $A_u(\mathbf{x}; \mathbf{v}_u) = \mathbf{v}_u^T\mathbf{x}$ and $A_w(\mathbf{x}; \mathbf{v}_w) = \mathbf{v}_w\mathbf{x}$, respectively, where $\mathbf{v}_u \in R^m$ and $\mathbf{v}_w \in R^l$. The output functions $s_u()$ and $s_w()$ are identities.

**Value-gradient-based policies.** Value-gradient-based policies for action and disturbance are given in equations 3.20 and 3.21. For the pendulum swing-up task, we use policies that correspond to the reward function (see equation A.3),

$$u = \frac{1}{2}R^{-1}\frac{\partial f(\mathbf{x}, u, w)}{u}^T\frac{\partial V}{\partial \mathbf{x}}^T, \tag{4.8}$$

$$w = -\frac{1}{2\gamma^2}\frac{\partial f(\mathbf{x}, u, w)}{w}^T\frac{\partial V}{\partial \mathbf{x}}^T. \tag{4.9}$$

For the cart pole swing-up task, we use value-gradient-based policies which correspond to the reward function (see equation C.7),

$$u = u^{\max}s\left(\frac{1}{c}\frac{\partial f(\mathbf{x}, u, \mathbf{w})}{\partial u}^T\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}^T\right), \tag{4.10}$$

$$w_j = w_j^{\max}s\left(-\frac{1}{d_j}\frac{\partial f(\mathbf{x}, u, \mathbf{w})}{\partial w_j}^T\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}^T\right)(j = 1, 2). \tag{4.11}$$

**Exploration strategy.** In order to promote exploration, we incorporated a noise term in both policies, in equations 3.12 and 3.13 and equations 3.21 and 3.22. We used low-pass filtered noise $\tau_n\dot{\mathbf{n}}^u(t) = -\mathbf{n}^u(t) + \sigma_u\mathbf{N}_u(t)$, and $\tau_n\dot{\mathbf{n}}^w(t) = -\mathbf{n}^w(t) + \sigma_w\mathbf{N}_w(t)$, where $\mathbf{N}_u(t)$ and $\mathbf{N}_w(t)$ denote normal gaussian noise (Doya, 2000).

**Simulation setup.** The physical systems were simulated by the fourth-order Runge-Kutta method, and the learning dynamics was simulated by the Euler method, both with the time step of 0.01 sec. Because calculating learning dynamics requires more computational resources and less accuracy than the physical dynamics of the pendulum and the cart pole, we used the faster numerical integration method (the Euler method) for the learning dynamics.

**4.1 Linear Inverted Pendulum.** We first considered a linear problem in order to test if the value function and the policy learned by robust RL coincide with the analytic solution of $H^\infty$ control problem. Thus, we considered only the pendulum dynamics near the unstable equilibrium point $\mathbf{x} = (0, 0)^T$. The parameters for the value function in equation 4.2 were given as

$$P = \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix}. \tag{4.12}$$

Each trial was started from an initial state $\mathbf{x}(0) = (\theta(0), \dot{\theta}(0))$, where $\theta(0)$ and $\dot{\theta}(0)$ were selected from a uniform distribution that ranged over $-\frac{\pi}{6} < \theta(0) < \frac{\pi}{6}$ and $-\frac{\pi}{6} < \dot{\theta}(0) < \frac{\pi}{6}$, respectively. We terminated a trial after 2 seconds, or when the state of the pendulum reached $|\theta| > \frac{\pi}{6}$.

Parameters for the learning process were given as $\eta = 1.0$, $\eta_u = 0.1$, $\eta_w = 0.1$, $\tau = 1.0$, $\kappa = 0.1$, $R = 1.0$, $\gamma = 1.5$, $\tau_n = 0.02$, $\sigma_u = 3.0$, and $\sigma_w = 2.0$.

*4.1.1 Actor-Disturber-Critic.* Here, we used robust RL implemented by the actor-disturber-critic shown in equations 4.6 and 4.7.

We initialized the parameters of the actor $\mathbf{v}^u$ as $\mathbf{v}^u = \{-10.0, -1.0\}$ and the disturber $\mathbf{v}^w = \{0.0, 0.0\}$, which made the system stable. The parameters of the value function $P$ in equation 4.12 were initialized with $\{p_{11}, p_{12}, p_{22}\} = \{-1.0, -10.0, -1.0\}$. The initial parameters of the value function are consistent with the initial parameters of the actor (i.e., $\mathbf{v}^u = R^{-1}B_1^T P$).

Figures 3A and 3B show learning performance of the actor-disturber-critic architecture. The parameters of the actor $\mathbf{v}^u$ and the disturber $\mathbf{v}^w$ converged to the values close to those of policies in equations 3.25 and 3.26 derived by solving the Riccati equation, 3.25.

*4.1.2 Value-Gradient-Based Robust Policy.* Value-gradient-based policies for action and disturbance are given based on equations 3.25 and 3.26. The parameter matrix $P$ was learned instead of using the solution of the Riccati equation, 3.25. Again, the parameters of the value function in equation 4.12 were initialized with $\{p_{11}, p_{12}, p_{22}\} = \{-1.0, -10.0, -1.0\}$. The parameter matrices $\hat{A}$ and $\hat{B}_1$ for the dynamics model are initialized as $\hat{A} = \mathbf{0}$ and $\hat{B}_1 = \mathbf{0}$.

Results in Figure 3C show that the parameters of the value function $P$ nearly converged to the solution of the Riccati equation, 3.25. As shown in Figure 3D, the model parameter $\hat{B}_1 = (b_1, b_2)^T$ used in the controller, equation 3.26, also converged to the exact model in equation A.2.

**4.2 Nonlinear Pendulum Swing-Up.** Now we consider the original nonlinear dynamics, equation A.2. We used the reward function defined

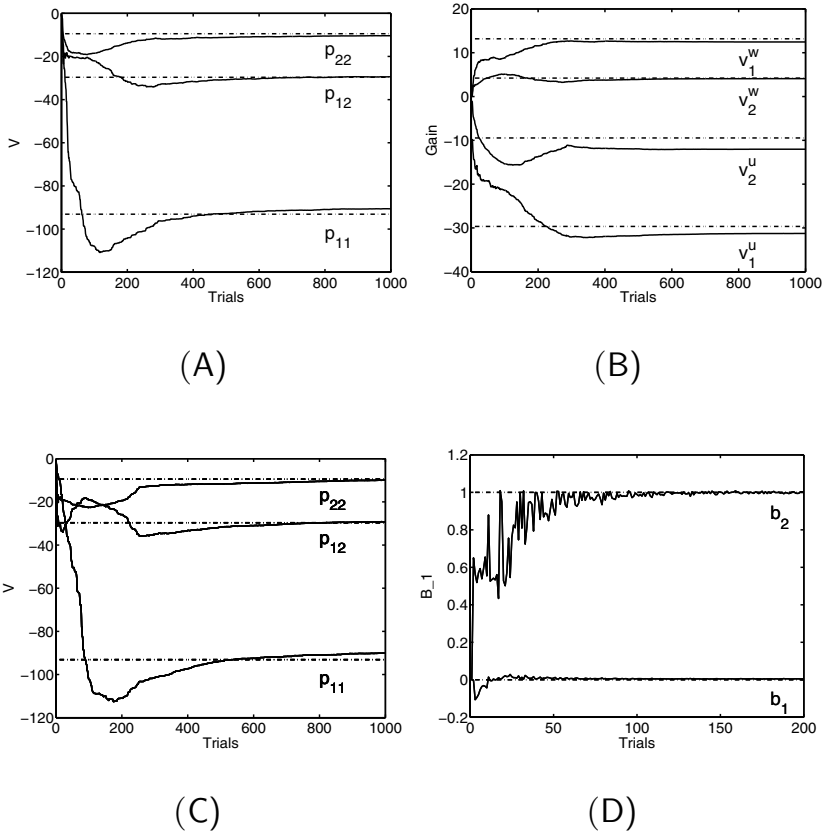



Figure 3: Learning performance of (A) estimated value function for the actor-disturber-critic, (B) the actor and the disturber, (C) estimated value function for the value-gradient-based robust policy, and (D) estimated dynamics model. The dash-dot lines show the exact values of the parameters.

in equation A.3 with the control cost $R = 0.05$. The value and policy functions were implemented by normalized gaussian networks with $21 \times 21$ bases for each dimension of the state space $\{\theta, \dot{\theta}\}$. A $4 \times 4 \times 2$ basis network for three-dimensional space $\{\theta, \dot{\theta}, u\}$ was used for modeling the system dynamics.

*4.2.1 Learning Performance.* Each trial was started from an initial state $\mathbf{x}(0) = (\theta(0), 0.0)$, where $\theta(0)$ was selected from a uniform distribution that ranged over $-\pi < \theta < \pi$.

We set the learning parameters as $\eta = 10.0$, $\eta^u = 5.0$, $\eta^w = 5.0$, $\tau = 1.5$, $\kappa = 0.1$, and $\tau_n = 0.2$. The sizes of the perturbation $\sigma_u$ and $\sigma_w$ were tapered off as the performance improved (Gullapalli, 1990). We took the modulation scheme $\sigma_u = 2.0 \min[1, \max[0, \frac{V_1 - V(t)}{V_1 - V_0}]]$ and $\sigma_w = 1.5 \min[1, \max[0, \frac{V(t) - V_0}{V_1 - V_0}]]$, where $V_0 = -2.0$ and $V_1 = 0.0$ are the minimal and maximal levels of the expected reward.

We used a range of the robustness parameter $\gamma$, starting from 1.0 and gradually reducing by 0.05, and found the smallest value with which the learning method acquired the swing-up policy. As we make $\gamma$ smaller, the policy becomes more robust and more conservative.

For the actor-disturber-critic, we initialized the parameters of the function approximators as $\mathbf{v} = \mathbf{0}$, $\mathbf{v}^u = \mathbf{0}$, and $\mathbf{v}^w = \mathbf{0}$. The robustness parameter was set to $\gamma = 0.45$.

For the value-gradient-based robust policy, we initialized the parameter of the value function as $\mathbf{v} = \mathbf{0}$, and the dynamics model as $\mathbf{v}^M = \mathbf{0}$. The robustness parameter was set to $\gamma = 0.25$. As suggested in Doya (2000), using the learned dynamics model and the value gradient has an advantage for learning the pendulum swing-up task. We found that the value-gradient method also has an advantage for learning a more robust policy compared to the actor-disturber-critic. Therefore, we can use a smaller robustness parameter ($\gamma = 0.25$) than that of the actor-disturber-critic method ($\gamma = 0.45$).

As a measure of the swing-up performance, we defined the time for which the pendulum stayed up ($|\theta| < \frac{\pi}{4}$) as $t_{up}$. A trial lasted for 20 seconds unless the pendulum was over-rotated ($|\theta| > 5\pi$). Upon such a failure, the trial was terminated with a reward $r(t) = -2.0$ for 0.5 second. We compared the learning performance of four learning schemes: actor-disturber-critic, actor-critic, value-gradient-based robust policy, and value-gradient-based standard policy. Figure 4 shows the results of learning performance. The actor-critic and the value-gradient-based standard policy learn the swing-up task faster than the actor-disturber-critic and the value-gradient-based robust policy, respectively. Results show that learning robust policies is slower than learning standard policies because the disturbance is explicitly considered during learning.

*4.2.2 Value Functions and Policies.* Figure 5 shows the value functions acquired by the value-gradient-based robust RL with robustness parameter $\gamma = 0.25$ and by the value-gradient-based standard RL ($\gamma = \infty$). The value function acquired by the value-gradient-based robust RL has a sharper ridge around the upright position, $\mathbf{x} = (\theta, \dot{\theta}) = (0.0, 0.0)$ and a smoother slope around the bottom position, $\mathbf{x} = (\theta, \dot{\theta}) = (\pi, 0.0)$ (see Fgiure 5A). The sharp ridge keep the pendulum at the upright position, and the smoother slope, which makes more swings around the bottom position, is suitable to cope with modeling error. Figure 6 shows the acquired actor and disturber. The

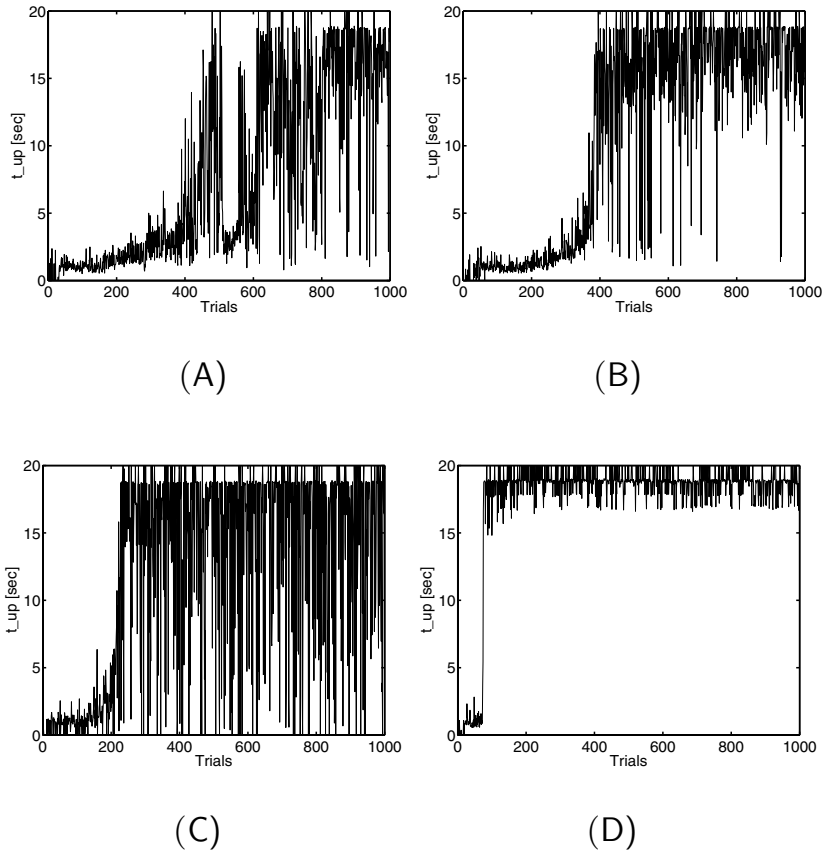



Figure 4: Comparison of the time course of learning with different control schemes: (A) actor-disturber-critic, (B) actor-critic, (C) value-gradient-based robust policy, and (D) value-gradient-based standard policy. $t_{up}$: time in which the pendulum stayed up.

disturber learned the policy that mainly disturbs the actor's policy around the upright position, $\mathbf{x} = (\theta, \dot{\theta}) = (0.0, 0.0)$.

*4.2.3 Robustness to Parameter Changes.* In Figure 7, we compare the robustness of the value-gradient-based robust policy and the value-gradient-based standard policy to the change of physical parameters. Both policies learned to swing up and hold a pendulum that has the weight $m = 1.0$ kg and the coefficient of friction $\mu = 0.01$ (see Figure 7A). The value-gradient-based robust policy took more swings, indicating its conservative control law.

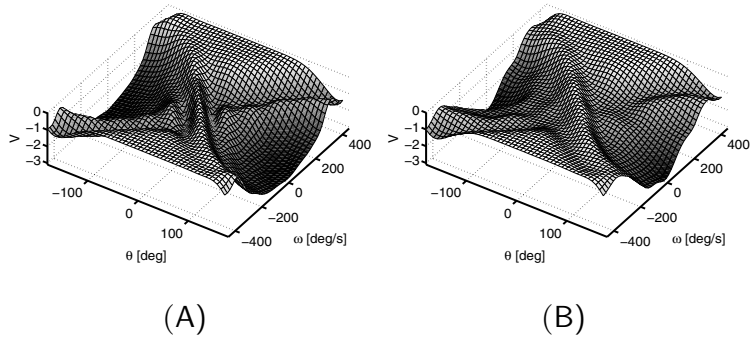



Figure 5: Shape of the value function acquired by the value-gradient-based RL after 1000 learning trials. (A) Robust RL ($\gamma = 0.25$). (B) Standard RL ($\gamma = \infty$).

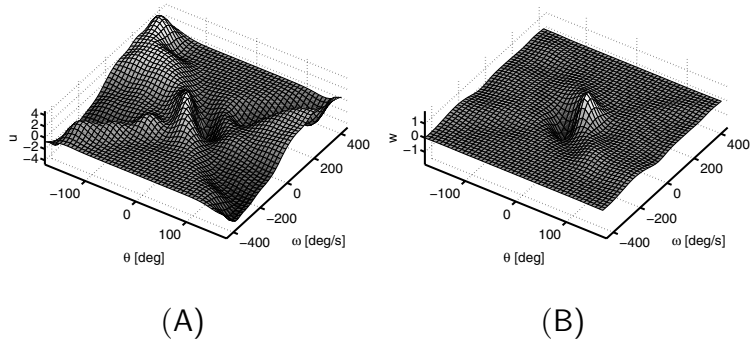



Figure 6: Shape of the control and disturbance function after 1000 learning trials. the robustness parameter was $\gamma = 0.45$ (A) Actor. (B) Disturber.

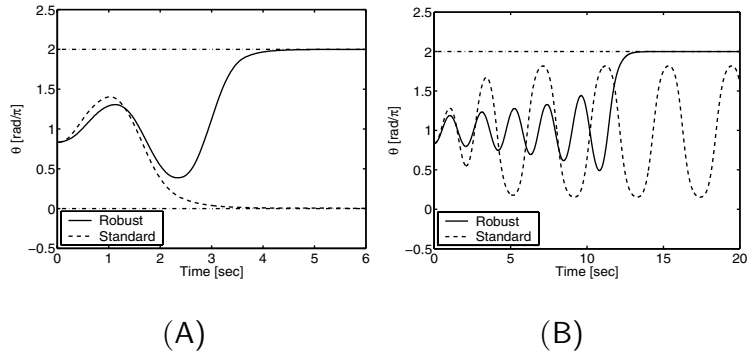



Figure 7: Swing-up trajectories with a pendulum with different weight and friction. The dash-dot lines show the upright position. The initial position was $\theta(0) = \frac{5}{6}\pi$. (A) $m = 1.0$, $\mu = 0.01$. (B) $m = 3.0$, $\mu = 0.5$.

Table 1: Comparison with Different Robustness Parameter (Average Performance of 10 Learned Controllers).

| $\gamma$ | 0.25 | 0.5 | 1.0 | $\infty$ |
|---|---|---|---|---|
| Max m[kg] | 5.9 | 3.9 | 3.2 | 3.1 |

Next, we applied both robust and standard policies to a pendulum that has different physical parameters ($m = 3.0$ kg, $\mu = 0.5$) from the originally learned environment ($m = 1.0$ kg, $\mu = 0.01$).

As shown in Figure 7B, the robust policy could successfully swing the pendulum up, while the standard policy could not. This result shows the robustness of the policy acquired by robust RL.

We also tested how the robustness parameter $\gamma$ affects robust performance of the learned controller. We compare the value-gradient-based robust policy with the three robustness parameters ($\gamma = 0.25, 0.5, 1.0$) and the value-gradient-based standard policy ($\gamma = \infty$). We trained these controllers with the pendulum mass $m = 1.0$ kg. Then we compared the maximum weight that each controller can successfully swing up, where the initial joint angle was $\theta(0) = \frac{5}{6}\pi$ and the coefficient of friction was the same as the originally learned environment, $\mu = 0.01$. A trial was regarded as successful when $t_{up} > 5$ seconds. We generated 10 controllers for each robustness parameter $\gamma$ and then compared the average performance.

The results of the comparison in Table 1 show a more robust performance with smaller $\gamma$. This results are consistent with $H^\infty$ control theory (Zhou et al., 1996).

**4.3 Cart Pole Swing-Up.** We compared the robust performance of the robust RL controller with the standard RL controller in the cart pole swing-up task. The value and policy functions were implemented by normalized gaussian networks with $7 \times 7 \times 21 \times 21$ bases for each dimension of the state space $\{x, v, \theta, \dot{\theta}\}$. A $2 \times 2 \times 4 \times 2 \times 4$ basis network for five-dimensional space $\{x, v, \theta, \dot{\theta}, u\}$ was used for modeling the system dynamics. When the cart bumped into the end of the track or when the pole overrotated ($|\theta| > 5\pi$), a terminal reward $r(t) = -1.0$ was given for 0.5 second. Otherwise a trial lasted for 20 seconds.

We set the learning parameters as $\eta = 20.0$, $\tau = 1.0$, $\kappa = 0.3$, $\eta_M = 10.0$, $\tau_n = 0.2$, $\sigma_u = 2.0 \min[1, \max[0, \frac{V_1 - V(t)}{V_1 - V_0}]]$, and $\sigma_w = 2.0 \min[1, \max[0, \frac{V(t) - V_0}{V_1 - V_0}]]$, where $V_0 = -1.0$ and $V_1 = 0.0$. Each trial was started from an initial state $\mathbf{x}(0) = (0.0, 0.0, \theta(0), 0.0)$, where $\theta(0)$ were selected from a uniform distribution that ranged over $-\pi < \theta(0) < \pi$.

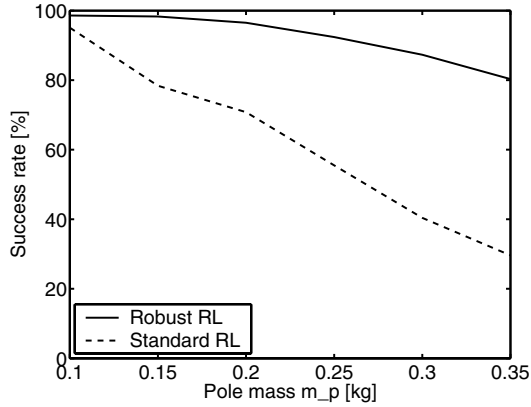



Figure 8: Success rate of the cart pole swing-up task (average success rate of 10 learned controllers).

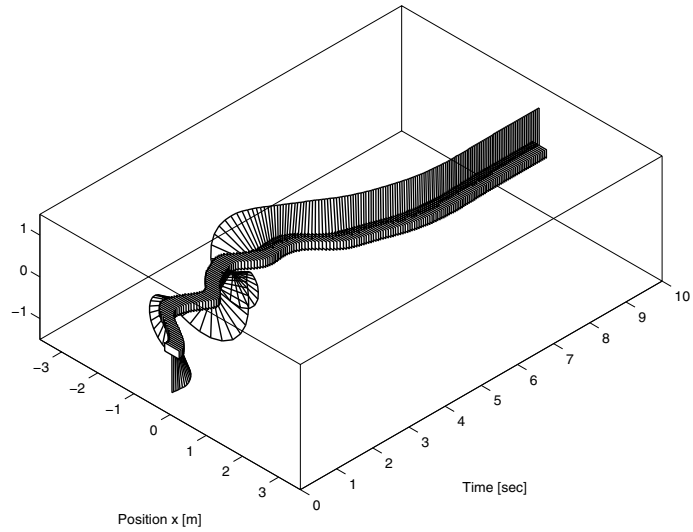
Here, we compared the average swing-up performance after 1000 learning trials. We changed the pole mass from $m_p = 0.1$ kg to $m_p = 0.35$ kg by 0.05 kg. Each trial was started from an initial joint angle $\theta(0)$ selected from a uniform distribution that ranged over $-\pi < \theta(0) < \pi$. Figure 8 shows the success rate of the cart pole swing-up task. The success rate was derived from the number of successful swing-up times in 100 trials. A trial was regarded as successful when $t_{up} > 5$ seconds, where we defined the time for which the pole stayed up ($|\theta| < 12$ deg) as $t_{up}$. We generated 10 robust controllers (policies) and standard controllers (policies) and then compared average success rate. We used robustness parameters defined in equation C.9 in appendix C. The success rate using the robust policy with the pole mass $m_p = 0.35$ kg was about 80%, while using standard policy was about 30%. This result shows the robustness of the acquired robust policy.

Figure 9A shows that the robust policy could swing up the pole with the cart pole that has different physical parameters (pole mass $m_p = 0.35$ kg, pole friction $\mu_p = 0.005$, the cart mass $m_c = 1.5$ kg) from the original environment ($m_p = 0.1$ kg, $\mu_p = 2.0 \times 10^{-6}$, $m_c = 1.0$ kg). Figure 9B shows that the standard policy failed to swing up the pole with the same cart pole used in Figure 9A.

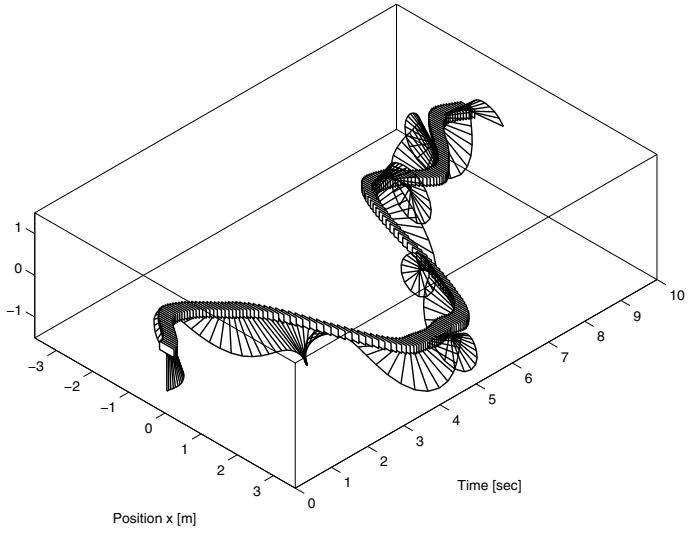
These results suggest that the robust RL can possibly be applied to a more difficult task with a higher-dimensional state space.

## 5 Discussion

$H^\infty$ control theory gives an analytical solution only for linear systems. For nonlinear systems, there is no analytical way of solving the HJI equation.




(A)




(B)

Figure 9: Cart pole swing-up trajectories with pole mass $m_p = 0.35$ kg, pole friction $\mu_p = 0.005$, and cart mass $m_c = 1.5$ kg. The initial joint angle was $\theta(0) = \pi$. (A) Value-gradiant-based robust policy. (B) Value-gradient-based standard policy.

In order to derive a nonlinear $H^\infty$ controller, the value function is usually derived by using dynamic programming (Imafuku, 1999; Coraluppi & Marcus, 1999). A similar approach to the methods using dynamics programming was proposed by Nilim and Ghaoui (2004) to cope with uncertain state transition matrices. However, these methods need off-line calculation and an environmental model. Robust RL can derive a nonlinear $H^\infty$ controller by online calculation and without any prior knowledge of an environmental model.

Robust RL provides a new way of using min-max solution in RL problems. Min-max RL was applied to games like backgammon (Tesauro, 1992) and Othello (Yoshioka & Ishii, 1998). Minimax Q-learning was proposed by Littman (1994, 2001) and applied to zero-sum games. However, in these studies, each player takes the same role. The min-max RL was also applied to a problem in which an airplane tries to avoid a missile and the missile tries to catch the airplane (Harmon, Baird, & Klopf, 1995). However, this study focuses on only linear control problems. We applied min-max RL in which each of two players (the control agent and the disturbing agent) has a different ability with the nonlinear problems of pendulum swing-up and cart pole swing-up.

Risk-sensitive control studies are also related to our RRL framework and have an interesting application to the field of finance. Neuneirer and Mihatsch (1998) proposed risk-sensitive reinforcement learning and applied their method to acquire an optimal investment policy for an artificial stock price.

Robustness during the learning process in RL framework was studied in Singh, Barto, Gruppen, and Connolly (1994) by using a constrained policy space and in Kretchmar et al. (2001) by combining standard RL with a linear robust controller. Although the target of our study is different from these studies, this kind of robustness is also important when we apply RL to real-world problems.

## 6 Conclusion

In this study, we proposed a new RL paradigm called robust reinforcement learning (RRL). We showed that RRL can learn the analytic solution of the linear $H^\infty$ controller for the inverted pendulum dynamics around the upright position and also showed that RRL can deal with modeling error that standard RL cannot in the nonlinear inverted pendulum swing-up simulation example. We also applied RRL to the cart pole swing-up task, and a robust swing-up policy was acquired by RRL. We will apply RRL to more complex tasks like learning stand-up behaviors (Morimoto & Doya, 2000, 2001a) and biped locomotion (Atkeson & Morimoto, 2003; Morimoto & Atkeson, 2003) in future work.

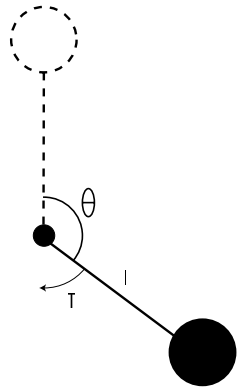



Figure 10: Single-pendulum swing-up task. $\theta$: joint angle, $T$: input torque, $l$: length of the pendulum.

## Appendix A: Pendulum Swing-Up Task

The task of swinging a pendulum up (Doya, 2000) is a simple but strongly nonlinear control task. The dynamics of the pendulum is given by

$$ml^2\ddot{\theta} = -\mu\dot{\theta} + mgl\sin\theta + T, \tag{A.1}$$

where $\theta$ is the angle from the upright position, $T$ is the input torque, $\mu = 0.01$ is the coefficient of friction, $m = 1.0$ kg is the weight of the pendulum, $l = 1.0$ m is the length of the pendulum, and $g = 9.8$ m/s$^2$ is the gravity acceleration (see Figure 10).

We define the state vector as $\mathbf{x} = (\theta, \dot{\theta})^T$ and the action as $u = T$. We consider disturbance input $w$ to the acceleration $\ddot{\theta}$. Thus the dynamics is given by an input-affine system:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\theta} \\ \frac{g}{l}\sin\theta - \frac{\mu}{ml^2}\dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{ml^2} \end{pmatrix} u + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w. \tag{A.2}$$

The augmented reward function is given by

$$q(\mathbf{x}, u, w) = (\cos\theta - 1) - Ru^2 + \gamma^2 w^2. \tag{A.3}$$

## Appendix B: Linear Inverted Pendulum

For comparison of robust RL with analytical linear $H^\infty$ control, we use a local linearization of the pendulum dynamics, equation A.2, near the unstable equilibrium point $\mathbf{x} = (0, 0)^T$. The coefficient matrices of the linear from

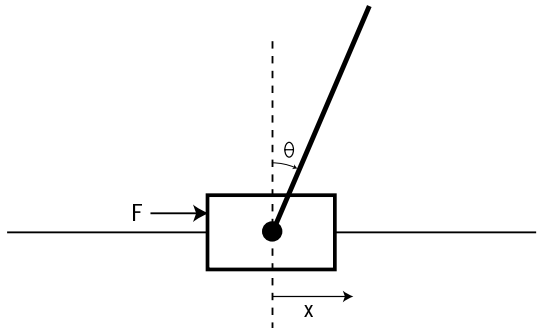



Figure 11: Cart pole swing-up task. $\theta$: joint angle, $x$: cart position $F$: input force.

equation 3.23 are given by

$$A = \begin{pmatrix} 0 & 1 \\ \frac{g}{l} & -\frac{\mu}{ml^2} \end{pmatrix}, \; B_1 = \begin{pmatrix} 0 \\ \frac{1}{ml^2} \end{pmatrix}, \; B_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{B.1}$$

The quadratic approximation of the augmented reward function, equation A.3, is given by

$$q(t) = -\mathbf{x}^T Q \mathbf{x} - Ru^2 + \gamma^2 w^2, \tag{B.2}$$

where $Q = diag\{0.5, 0\}$.

## Appendix C: Cart Pole Swing-Up Task

The cart pole swing-up (see Figure 11; Doya, 2000) is a strongly nonlinear extension to the common cart pole balancing task (Barto, Sutton, & Anderson, 1983). The physical parameters of the cart pole were the same in Barto et al. (1983) and Doya (2000). Marked differences from the pendulum swing-up task are that the dimension of the state space is higher and that the disturbance input has two degrees of freedom.

The state vector is $\mathbf{x} = \{x, v, \theta, \dot{\theta}\}$, where $x$ and $v$ are, respectively, the position and the velocity of the cart. The dynamics of the cart pole is modeled by the following nonlinear ordinary differential equations:

$$\ddot{\theta} = \frac{g \sin\theta + \frac{\cos\theta(\mu_c \mathrm{sign}(\dot{x}) - F - m_p l \dot{\theta}^2 \sin\theta)}{m_c + m_p} - \frac{\mu_p \dot{\theta}}{m_p l}}{l\left(\frac{4}{3} - \frac{m_p \cos^2\theta}{m_c + m_p}\right)}, \tag{C.1}$$

$$\ddot{x} = \frac{F + m_p l(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta) - \mu_c \mathrm{sign}(\dot{x})}{m_c + m_p}. \tag{C.2}$$

The following specifications were used for the simulated pole and cart: the track ranged from −3.6 m to 3.6 m; mass of the cart: $m_c = 1.0$ kg; mass of the pole: $m_p = 0.1$ kg; half-length of the pole $l = 0.5$ m; gravity $g = 9.8\text{m/s}^2$; coefficient of friction of cart on track: $\mu_c = 5.0 \times 10^{-4}$; coefficient of friction of pole on cart: $\mu_p = 2.0 \times 10^{-6}$. $F$ denotes applied force, $x$ denotes the cart position, and $\theta$ denotes joint angle of the pole.

The cart-pole dynamics can be written as an input-affine system,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u + B_2\mathbf{w}, \tag{C.3}$$

where

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} v \\ \frac{m_p l(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta) - \mu_c sign(\dot{x})}{m_c + m_p} \\ \dot{\theta} \\ \frac{g\sin\theta + \frac{\cos\theta(\mu_c sign(\dot{x}) - m_p l\dot{\theta}^2 \sin\theta)}{m_c+m_p} - \frac{\mu_p \dot{\theta}}{m_p l}}{l\left(\frac{4}{3} - \frac{m_p \cos^2\theta}{m_c+m_p}\right)} \end{pmatrix}, \tag{C.4}$$

$$\mathbf{g}(\mathbf{x}) = \begin{pmatrix} v \\ \frac{1}{m_c+m_p} \\ \dot{\theta} \\ -\frac{\frac{\cos\theta}{m_c+m_p}}{l\left(\frac{4}{3} - \frac{m_p\cos^2\theta}{m_c+m_p}\right)} \end{pmatrix}, \tag{C.5}$$

and we used the input matrix for disturbance $B_2$:

$$B_2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}. \tag{C.6}$$

The augmented reward was given by

$$q(\mathbf{x}, u, \mathbf{w}) = 0.5(\cos\theta - 1) - S(u) + \sum_{j=1}^{2} \Omega_j(w_j), \tag{C.7}$$

where

$$S(u) = c\int_0^u s^{-1}\left(\frac{a}{u^{\max}}\right) da, \tag{C.8}$$

$$\Omega_j(w_j) = d_j \int_0^{w_j} s^1\left(\frac{a}{w_j^{\max}}\right) da \quad (j = 1, 2), \tag{C.9}$$

$$s(x) = \frac{\pi}{2} \arctan\left(\frac{2}{\pi} x\right), \tag{C.10}$$

$u^{\max} = 20.0\ N$, $w_1^{\max} = 1.0\ N$, $w_2^{\max} = 10.0\ N \cdot m$, $c = 1.0$, $d_1 = 1.0$, and $d_2 = 1.0$.

## Appendix D: Normalized Gaussian Network

A value function is represented by

$$V(\mathbf{x}; \mathbf{v}) = \sum_{k=1}^{K} v_k b_k(\mathbf{x}), \tag{D.1}$$

where the normalized gaussian basis function is represented by

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

The vectors $\mathbf{c}_k$ and $\mathbf{s}_k$ define the center and the size of the $k$th basis function, respectively. Note that if there is no neighboring basis function, the shape of the basis functions extends 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-disturber-critic method, the polices are implemented as

$$\mathbf{u}(t) = s_u\left(\sum_k v_k^u b_k(\mathbf{x}(t)) + \mathbf{n}^u(t)\right), \tag{D.3}$$

$$\mathbf{w}(t) = s_w\left(\sum_k v_k^w b_k(\mathbf{x}(t)) + \mathbf{n}^w(t)\right), \tag{D.4}$$

where $s_w()$ and $s_w()$ are monotonically increasing output function, and $\mathbf{n}^u$ and $\mathbf{n}^w$ are the noise. In the value-gradient-based methods, the control and disturbance policies are given by

$$\mathbf{u}(t) = s_u\left(\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{u}}^T \sum_k v_k \frac{\partial b_k(\mathbf{x})}{\partial \mathbf{x}} + \mathbf{n}^u(t)\right), \tag{D.5}$$

$$\mathbf{w}(t) = s_w\left(-\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}}^T \sum_k v_k \frac{\partial b_k(\mathbf{x})}{\partial \mathbf{x}}^T + \mathbf{n}^w(t)\right). \tag{D.6}$$

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

$$\dot{\mathbf{x}}(t) - \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}}\mathbf{w}(t) \simeq \hat{f}(\mathbf{x}, \mathbf{u}) = \sum_k v_k^M b_k(\mathbf{x}(t), \mathbf{u}(t)). \tag{D.7}$$

The weights are updated by

$$\dot{v}_k^M(t) = \eta^M \left( \dot{\mathbf{x}}(t) - \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}}\mathbf{w}(t) - \hat{f}(\mathbf{x}(t), \mathbf{u}(t)) \right) b_k(\mathbf{x}(t), \mathbf{u}(t)), \tag{D.8}$$

where the learning rate was set to $\eta^M = 10.0$ for all simulations, and the input gain of the system dynamics is given by

$$\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \simeq \sum_k v_k^M \frac{\partial b_k(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=0}. \tag{D.9}$$

**Acknowledgments**

We thank Christopher G. Atkeson, Mitsuo Kawato, and the anonymous reviewers for their helpful comments.

**References**

Atkeson, C. G., & Morimoto, J. (2003). Nonparametric representation of policies and value functions: A trajectory-based approach. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems, 15* (pp. 643–1650). Cambridge, MA: MIT Press.

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

Coraluppi, S. P., & Marcus, S. I. (1999). Risk-sensitive and minmax control of discrete-time finite-state Markov decision processes. *Automatica, 35*, 301–309.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation, 12*(1), 219–245.

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., & Klopf, A. H. (1995). Advantage updating applied to a differential game. In G. Tesauro, D.S. Touretzky, & T.K. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 353–360). Cambridge, MA: MIT Press.

Imafuku, K. (1999). *Singularities of nonlinear control systems designed by Hamilton-Jacobi equations*. Unpublished doctoral dissertation, Nara Institute of Science and Technology.

Kretchmar, R. M., Young, P. M., Anderson, C. W., Hittle, D. C., Anderson, M. L., & Delnero, C. C. (2001). Robust reinforcement learning control with static and dynamic stability. *International Journal of Robust and Nonlinear Control, 11*, 1469–1500.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning* (pp.157–163). San Francisco: Morgan Kaufmann.

Littman, M. L. (2001). Value-function reinforcement learning in Markov games. *Journal of Cognitive Systems Research, 2*, 55–66.

Morimoto, J., & Atkeson, C. G. (2003). Minimax differential dynamic programming: An application to robust biped walking. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems, 15* (pp. 1563–1570). Cambridge, MA: MIT Press.

Morimoto, J., & Doya, K. (2000). Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. In *Proceedings of Seventeenth International Conference on Machine Learning* (pp. 623–630). San Francisco: Morgan Kaufmann.

Morimoto, J., & Doya, K. (2001a). Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems, 36*, 37–51.

Morimoto, J., & Doya, K. (2001b). Robust reinforcement learning. In T.K. Leen, T. G., Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems, 13* (pp. 1061–1067). Cambridge, MA: MIT Press.

Neuneier, R., & Mihatsch, O. (1998). Risk sensitive reinforcement learning. In M.S. Kearns, S A Solla, & D. A. Cohn (Eds.), *Advances in neural information processing systems, 11* (pp. 1031–1037). Cambridge, MA: MIT Press.

Nilim, A., & Ghaoui, L. E. (2004). Robustness in Markov decision problems with uncertain transition matrices. In S. Thrun, L. Saul, & B. Schölkopf, (Eds.), *Advances in neural information processing systems, 16*. Cambridge, MA: MIT Press.

Singh, S. P., Barto, A. G., Grupen, R., & Connolly, C. (1994). Robust reinforcement learning in motion planning. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems, 6* (pp. 655–662). San Francisco: Morgan Kaufmann.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning 8*, 257–277.

Weiland, S. (1989). Linear quadratic games, $H_\infty$, and the Riccati equation. In *Proceedings of the Workshop on the Riccati Equation in Control, Systems, and Signals* (pp. 156–159) Como, Italy.

Yoshioka, T., & Ishii, S. (1998). Strategy acquisition for the game "othello" based on reinforcement learning. In S. Usui & T. Omori (Eds.), *International Conference on Neural Information Processing* (pp. 841–844). London: ISO Press.

Zhou, K., Doyle, J. C., & Glover, K. (1996). *Robust optimal control*. Englewood Cliffs, NJ: Prentice Hall.

Received May 13, 2004; accepted July 2, 2004.