Reward Weighted Regression with Sample Reuse for Direct Policy Search in Reinforcement Learning*

Hirotaka Hachiya
Tokyo Institute of Technology, Japan
hachiya@sg.cs.titech.ac.jp

Jan Peters
Max-Planck Institute for Biological Cybernetics, Germany
jan.peters@tuebingen.mpg.de

Masashi Sugiyama
Tokyo Institute of Technology, Japan
sugi@cs.titech.ac.jp
http://sugiyama-www.cs.titech.ac.jp/~sugi/

Abstract

Direct policy search is a promising reinforcement learning framework in particular for controlling continuous, high-dimensional systems. Policy search often requires a large number of samples for obtaining a stable policy update estimator. However, this is prohibitive when the sampling cost is expensive. In this paper, we extend an expectation-maximization based policy search method so that previously collected samples can be efficiently reused. The usefulness of the proposed method, called *Reward-weighted Regression with sample Reuse* (R^3), is demonstrated through robot learning experiments.

Keywords

reinforcement learning, EM-based policy search, data reuse, adaptive importance sampling, importance-weighted cross-validation, robot control

*This paper is an extended version of our earlier conference paper (Hachiya et al., 2009b).

1 Introduction

Policy search is an important tool for solving real-world Markov decision problems online. However, many data samples are usually required for obtaining good control policies. In practice, the cost of collecting rollout data is often prohibitively expensive and too time-consuming for real-world problems where thousands of trials would require weeks or months of experiments. For example, when a robot learns how to hit a ball in baseball or tennis, robot engineers need to let the robot hit a ball hundreds of times for obtaining reliable policy improvement; then this policy update steps need to be repeated many times for finally obtaining a good policy. In this procedure, robot engineers need to spend long time to "nurse" the vulnerable robot through frequent mechanical maintenance. As in many other real-world reinforcement learning problems, it is therefore highly important to reduce the number of training samples generated by the physical system and instead re-use them efficiently in future updates.

A lot of efforts have been made to *reuse* previously collected samples, in particular in the context of value function approximation. A basic technique for sample reuse is to use *importance sampling* (Sutton & Barto, 1998) for which the bias is canceled out asymptotically. However, a naive use of importance sampling significantly increases the variance of estimators and, therefore, it becomes highly unstable. To mitigate this problem, the *per-decision* importance-weighting technique has been introduced for variance reduction (Precup et al., 2000). This technique efficiently makes use of a property of Markov decision processes and eliminates irrelevant terms in the importance sampling identity. However, the obtained estimator still tends to be unstable and, thus, the importance-sampling paradigm has not been in active use in real-world reinforcement learning tasks yet.

For more significant variance reduction, *adaptive* importance sampling techniques have been applied to reinforcement learning recently (Uchibe & Doya, 2004; Wawrzynski, 2009). The idea of adaptive importance sampling is to trade the variance reduction with a slight bias increase by introducing an 'adaptation' parameter. However, its performance heavily depends on the choice of the adaptation parameter and the optimal parameter values tend to change through the process of learning. Thus, manually selecting a fixed value of the adaptation parameter is not favorable. In order to optimally select the adaptation parameter, *importance-weighted cross-validation* (Sugiyama et al., 2007) was introduced into value function approximation to tune the adaptive parameter so as to minimize the estimated approximation error (Hachiya et al., 2009a).

Due to the above efforts, reinforcement learning methods based on value function approximation can now successfully reuse previously collected samples in a stable manner. However, it is not easy to deal with continuous actions in the value function based policy iteration framework; the *direct policy search* approach is more suitable for learning control policies with continuous actions, e.g., the *policy gradient* method (Williams, 1992; Sutton et al., 2000), the *natural policy gradient* method (Kakade, 2002; Peters et al., 2005) and *policy search by expectation-maximization* (Dayan & Hinton, 1997; Peters & Schaal, 2007). Reusing data samples is even more urgent in policy search approaches as small policy updating steps can result into under-utilization of the data. While plain importance

sampling techniques have also been employed in direct policy search, they were shown to be unstable (Shelton, 2001; Peshkin & Shelton, 2002). For stabilization purposes, heuristic techniques are often used in practice, e.g., samples with smaller importance weights are not used for learning (Uchibe & Doya, 2004; Kober & Peters, 2008). However, to the best of our knowledge, systematic treatment of instability issues in policy search with sample reuse is still an open research topic.

The purpose of this paper is to propose a new framework for systematically addressing the instability problems in direct policy search. In particular, we combine policy search by expectation-maximization (Dayan & Hinton, 1997; Peters & Schaal, 2007) with covariate shift adaptation (Shimodaira, 2000; Sugiyama & Müller, 2005; Sugiyama et al., 2007), which is a statistical learning paradigm under non-stationarity. Within this new framework, we develop an efficient data-reuse algorithm for direct policy learning. The effectiveness of the proposed method, called *Reward-weighted Regression with sample Reuse* (R^3), is demonstrated by simulated robot-control experiments.

The rest of this paper is organized as follows. In Section 2, we formulate the policy search problem in reinforcement learning, and review the reward weighted regression method and importance sampling techniques. In Section 3, we describe our proposed method, R^3, which allows us to efficiently reuse previously collected samples in the reward weighted regression framework. Experimental results are reported in Section 4, demonstrating the effectiveness of the R^3 algorithm in robot control tasks such as one-dimensional ball-balancing, robot-arm ball-balancing, and Acrobot swing-up. Finally, we conclude in Section 5 by summarizing our contributions and describing future work.

2 Policy Search Framework

We consider the standard reinforcement learning framework in which an agent interacts with the environment modeled as a Markov decision problem. In this section, we review how the Markov decision problem is solved using policy search by expectation-maximization (Dayan & Hinton, 1997); for Gaussian models, this results in the reward-weighted regression (RWR) algorithm (Peters & Schaal, 2007).

2.1 Markov Decision Problem

Let us consider a Markov decision problem specified by $(\mathcal{S}, \mathcal{A}, P_{\mathrm{T}}, P_{\mathrm{I}}, R, \gamma)$, where $\mathcal{S}$ is a set of (continuous) states, $\mathcal{A}$ is a set of (continuous) actions, $P_{\mathrm{T}}(\boldsymbol{s}'|\boldsymbol{s}, a)(> 0)$ is the transition probability-density from state $\boldsymbol{s}$ to next state $\boldsymbol{s}'$ when action a is taken, $P_{\mathrm{I}}(\boldsymbol{s})$ (> 0) is the probability density of the initial state, $R(\boldsymbol{s}, a, \boldsymbol{s}')$ (≥ 0) is an immediate reward for transition from $\boldsymbol{s}$ to $\boldsymbol{s}'$ by taking action a, and γ $(\in (0, 1])$ is the discount factor for future rewards. Let $\pi(a|\boldsymbol{s}; \boldsymbol{\theta})$ (> 0) be a stochastic policy with parameter $\boldsymbol{\theta}$, which represents the conditional probability density of taking action a given state $\boldsymbol{s}$.

Let us denote an *episode* of the agent's experience by

$$d \equiv (\boldsymbol{s}_1, a_1, \boldsymbol{s}_2, a_2, \ldots, \boldsymbol{s}_N, a_N, \boldsymbol{s}_{N+1}),$$

where $\boldsymbol{s}_1$ is an initial state selected following $P_{\mathrm{I}}(\boldsymbol{s}_1)$, a_n is an action chosen in state $\boldsymbol{s}_n$ following $\pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta})$, $\boldsymbol{s}_{n+1}$ is a state visited after taking action a_n in state $\boldsymbol{s}_n$ following $P_{\mathrm{T}}(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n,a_n)$, and N is the number of steps in the episode. The probability density $P(d;\boldsymbol{\theta})$ of an episode d occurring is given by

$$P(d;\boldsymbol{\theta}) \equiv P_{\mathrm{I}}(\boldsymbol{s}_1)\prod_{n=1}^{N}\pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta})P_{\mathrm{T}}(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n,a_n). \tag{1}$$

Let $\mathcal{R}(d)$ be the *return* (i.e., the sum of discounted rewards) along episode d:

$$\mathcal{R}(d) \equiv \sum_{n=1}^{N}\gamma^{n-1}R(\boldsymbol{s}_n,a_n,\boldsymbol{s}_{n+1}).$$

The *expected return* is denoted by $J(\boldsymbol{\theta})$:

$$J(\boldsymbol{\theta}) \equiv \int \mathcal{R}(d)P(d;\boldsymbol{\theta})\mathrm{d}d.$$

Note that the expected return is regarded as a function of parameter $\boldsymbol{\theta}$ since the probability density of episodes occurring depends on it.

The goal of reinforcement learning is to find the optimal policy $\boldsymbol{\theta}^*$ that maximizes the expected return $J(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* \equiv \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \tag{2}$$

2.2 Policy Search by Expectation-Maximization

Directly maximizing $J(\boldsymbol{\theta})$ is hard since $J(\boldsymbol{\theta})$ usually contains high non-linearity. The basic idea of policy search by expectation-maximization (EM) is to iteratively update the policy parameter $\boldsymbol{\theta}$ by maximizing a lower bound of the expected return (Dempster et al., 1977).

Let $\boldsymbol{\theta}_L$ be the current policy parameter, where the subscript L indicates the iteration number. By assuming that return $R(d)$ is nonnegative, Jensen's inequality (Bishop, 2006) yields the following lower bound of the log-expected return (see Appendix A for details):

$$\log J(\boldsymbol{\theta}) \geq \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)}{J(\boldsymbol{\theta}_L)}\log\frac{P(d;\boldsymbol{\theta})}{P(d;\boldsymbol{\theta}_L)}\mathrm{d}d + \log J(\boldsymbol{\theta}_L) \equiv \log J_L(\boldsymbol{\theta}).$$

In the EM approach, the parameter $\boldsymbol{\theta}$ is iteratively updated by maximizing the lower bound $J_L(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}_{L+1} \equiv \arg\max_{\boldsymbol{\theta}} J_L(\boldsymbol{\theta}). \tag{3}$$

Since $\log J_L(\boldsymbol{\theta}_L) = \log J(\boldsymbol{\theta}_L)$, the lower bound $J_L(\boldsymbol{\theta})$ is *tight* (i.e., the lower bound touches the target function) at $\boldsymbol{\theta}_L$. Thus monotone non-decrease of the expected return is guaranteed:

$$J(\boldsymbol{\theta}_{L+1}) \geq J(\boldsymbol{\theta}_L).$$

This update is iterated until convergence (see Figure 1).

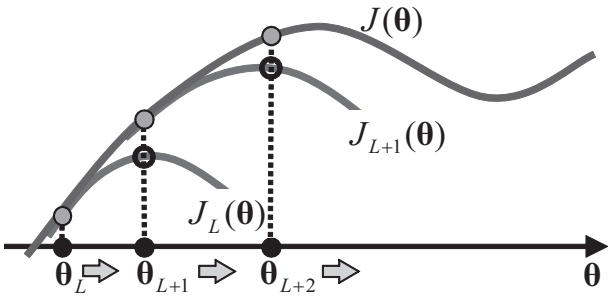


Figure 1: Illustration of policy-parameter update in EM-based policy search. The policy parameter $\boldsymbol{\theta}$ is updated iteratively by maximizing lower bounds $J_L(\boldsymbol{\theta})$. The lower bound $J_L(\boldsymbol{\theta})$ touches $J(\boldsymbol{\theta})$ at $\boldsymbol{\theta}_L$.

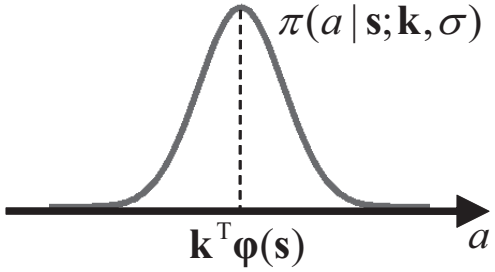


Figure 2: Illustration of the Gaussian policy model. $\boldsymbol{k}^\top \boldsymbol{\phi}(\boldsymbol{s})$ is the mean and σ is the standard deviation of the Gaussian function.

2.3 Reward-Weighted Regression

Let us employ the Gaussian policy model defined as

$$\pi(a|\boldsymbol{s};\boldsymbol{\theta}) = \pi(a|\boldsymbol{s};\boldsymbol{k},\sigma) \equiv \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(a-\boldsymbol{k}^\top\boldsymbol{\phi}(\boldsymbol{s}))^2}{2\sigma^2}\right), \tag{4}$$

where $\boldsymbol{\phi}(\boldsymbol{s}) \equiv (\phi_1(\boldsymbol{s}), \phi_2(\boldsymbol{s}), \ldots, \phi_B(\boldsymbol{s}))^\top$ are fixed basis functions, B is the number of basis functions, and $\boldsymbol{\theta} = (\boldsymbol{k}^\top, \sigma)^\top$ ($\boldsymbol{k} \in \mathcal{R}^B$ and $\sigma > 0$) are policy parameters (see Figure 2).

This model allows us to deal with one-dimensional action a and multi-dimensional state vector $\boldsymbol{s}$—multi-dimensional action vectors may be handled by concatenating one-dimensional models. Note that the Gaussian policy model can be seen as a linear function (with respect to the parameter $\boldsymbol{k}$) contaminated by Gaussian noise ε with mean zero and standard deviation σ:

$$a = \boldsymbol{k}^\top\boldsymbol{\phi}(\boldsymbol{s}) + \varepsilon.$$

The maximizer $\boldsymbol{\theta}_{L+1} = (\boldsymbol{k}_{L+1}^\top, \sigma_{L+1})^\top$ of the lower bound $\log J_L(\boldsymbol{\theta})$ can be analytically

obtained (see Appendix B for details) as

$$\begin{cases} \boldsymbol{k}_{L+1} = \left(\int \mathcal{R}(d) P(d;\boldsymbol{\theta}_L) \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{s}_n) \boldsymbol{\phi}(\boldsymbol{s}_n)^\top \mathrm{d}d \right)^{-1} \\ \qquad\qquad \times \left(\int \mathcal{R}(d) P(d;\boldsymbol{\theta}_L) \frac{1}{N} \sum_{n=1}^{N} a_n \boldsymbol{\phi}(\boldsymbol{s}_n) \mathrm{d}d \right), \\ \sigma_{L+1}^2 = \left(\int \mathcal{R}(d) P(d;\boldsymbol{\theta}_L) \mathrm{d}d \right)^{-1} \left(\int \mathcal{R}(d) P(d;\boldsymbol{\theta}_L) \frac{1}{N} \sum_{n=1}^{N} (a_n - \boldsymbol{k}_{L+1}{}^\top \boldsymbol{\phi}(\boldsymbol{s}_n))^2 \mathrm{d}d \right). \end{cases}$$

EM-based policy search for Gaussian models is called *reward-weighted regression* (Peters & Schaal, 2007).

2.4 Learning from Episodic Data Samples

Suppose a dataset consisting of M episodes with N steps is available for each RWR iteration, where episodic samples at the Lth iteration are generated as follows. Initially, the agent starts from a randomly selected state $\boldsymbol{s}_1$ following the initial-state probability density $P_{\mathrm{I}}(\boldsymbol{s}_1)$ and chooses an action based on the policy $\pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta}_L)$. Then the agent makes a transition following $P_{\mathrm{T}}(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n, a_n)$ and receives a reward r_n $(= R(\boldsymbol{s}_n, a_n, \boldsymbol{s}_{n+1}))$. This transition is repeated N times for M episodes—hence, the training data $\mathcal{D}^L$ gathered at the Lth iteration is expressed as $\mathcal{D}^L \equiv \{d_m^L\}_{m=1}^M$, where each episodic sample d_m^L consists of a set of 4-tuple elements as

$$d_m^L \equiv \{(\boldsymbol{s}_{m,n}^L, a_{m,n}^L, r_{m,n}^L, \boldsymbol{s}_{m,n+1}^L)\}_{n=1}^N.$$

Thus the RWR solution $\boldsymbol{\theta}_{L+1} \equiv (\boldsymbol{k}_{L+1}^\top, \sigma_{L+1})^\top$ can be approximated using the Lth training data $\mathcal{D}^L$ as $\widehat{\boldsymbol{\theta}}_{L+1} \equiv (\widehat{\boldsymbol{k}}_{L+1}^\top, \widehat{\sigma}_{L+1})^\top$, where

$$\begin{cases} \widehat{\boldsymbol{k}}_{L+1} = \left(\frac{1}{M} \sum_{m=1}^{M} \mathcal{R}(d_m^L) \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^L) \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^L)^\top \right)^{-1} \\ \qquad\qquad \times \left(\frac{1}{M} \sum_{m=1}^{M} \mathcal{R}(d_m^L) \frac{1}{N} \sum_{n=1}^{N} a_{m,n}^L \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^L) \right), \\ \widehat{\sigma}_{L+1}^2 = \left(\frac{1}{M} \sum_{m=1}^{M} \mathcal{R}(d_m^L) \right)^{-1} \left(\frac{1}{M} \sum_{m=1}^{M} \mathcal{R}(d_m^L) \frac{1}{N} \sum_{n=1}^{N} (a_{m,n}^L - \widehat{\boldsymbol{k}}_{L+1}^\top \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^L))^2 \right). \end{cases} \quad (5)$$

Since the expectation is simply replaced by the sample average, the above solution may be *consistent*, i.e., as the number of episodes M goes to infinity, the solution converges to the optimal value in probability. A pseudo code of RWR using episodic data samples is summarized in Figure 3.

Algorithm 2.1: RWR$(l, M, N, \boldsymbol{\phi}, \widehat{\boldsymbol{\theta}}_1, \epsilon)$

$//l$ Number of iterations
$//M$ Number of episodes collected at each iteration
$//N$ Number of steps in each episode
$//\boldsymbol{\phi}$ Basis functions, $\boldsymbol{\phi}(s) = (\phi_1(s), \phi_2(s), \ldots, \phi_B(s))^\top$
$//\widehat{\boldsymbol{\theta}}_1$ Initial policy parameter, $\widehat{\boldsymbol{\theta}}_1 = (\widehat{\boldsymbol{k}}_1^\top, \widehat{\sigma}_1)^\top$
$//\epsilon$ Stopping criterion

$L \leftarrow 0$
repeat
$L \leftarrow L + 1$

// Collect data samples using current policy π
$\mathcal{D}^L \leftarrow \textsc{DataSampling}(\widehat{\boldsymbol{\theta}}_L, M, N)$

// Update policy parameter $\widehat{\boldsymbol{\theta}}_L$ using episodic data $\mathcal{D}^L$

$$\widehat{\boldsymbol{k}}_{L+1} \leftarrow \left(\frac{1}{M} \sum_{m=1}^{M} \mathcal{R}(d_m^L) \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^L) \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^L)^\top \right)^{-1} \times \left(\frac{1}{M} \sum_{m=1}^{M} \mathcal{R}(d_m^L) \frac{1}{N} \sum_{n=1}^{N} a_{m,n}^L \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^L) \right)$$

$$\widehat{\sigma}_{L+1}^2 \leftarrow \left(\frac{1}{M} \sum_{m=1}^{M} \mathcal{R}(d_m^L) \right)^{-1} \left(\frac{1}{M} \sum_{m=1}^{M} \mathcal{R}(d_m^L) \frac{1}{N} \sum_{n=1}^{N} (a_{m,n}^L - \widehat{\boldsymbol{k}}_{L+1}^\top \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^L))^2 \right)$$

$\widehat{\boldsymbol{\theta}}_{L+1} \leftarrow (\widehat{\boldsymbol{k}}_{L+1}^\top, \widehat{\sigma}_{L+1})$

until $\|\widehat{\boldsymbol{\theta}}_{L+1} - \widehat{\boldsymbol{\theta}}_L\| \leq \epsilon$
return $(\widehat{\boldsymbol{\theta}}_{L+1})$

Figure 3: Pseudo code of reward-weighted regression using episodic data samples. By the DataSampling function, data samples (M episodes and N steps) are collected following current policy $\pi(a|\boldsymbol{s}; \widehat{\boldsymbol{\theta}}_L)$.

2.5 Importance Sampling

When the cost for gathering rollout samples is high, the number M of episodes should be kept small. As a result, the next policy parameter $\widehat{\boldsymbol{\theta}}_{L+1}$ suggested by RWR may not be sufficiently accurate. In order to improve the estimation accuracy, we may reuse the samples collected at the previous iterations $\{\mathcal{D}^l\}_{l=1}^L$.

If the policies remain unchanged by the RWR updates, just using Eq.(5) gives a consistent estimator. This estimator is denoted by $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{NIW}} \equiv (\widehat{\boldsymbol{k}}_{L+1}^{\mathrm{NIW}\top}, \widehat{\sigma}_{L+1}^{\mathrm{NIW}})^\top$, where

$$\begin{cases} \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{NIW}} = \left(\dfrac{1}{L}\displaystyle\sum_{l=1}^{L}\dfrac{1}{M}\sum_{m=1}^{M}\mathcal{R}(d_m^l)\dfrac{1}{N}\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)^\top\right)^{-1} \\ \qquad\qquad \times\left(\dfrac{1}{L}\displaystyle\sum_{l=1}^{L}\dfrac{1}{M}\sum_{m=1}^{M}\mathcal{R}(d_m^l)\dfrac{1}{N}\sum_{n=1}^{N}a_{m,n}^l\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\right), \\ (\widehat{\sigma}_{L+1}^{\mathrm{NIW}})^2 = \left(\dfrac{1}{L}\displaystyle\sum_{l=1}^{L}\dfrac{1}{M}\sum_{m=1}^{M}\mathcal{R}(d_m^l)\right)^{-1} \\ \qquad\qquad \times\left(\dfrac{1}{L}\displaystyle\sum_{l=1}^{L}\dfrac{1}{M}\sum_{m=1}^{M}\mathcal{R}(d_m^l)\dfrac{1}{N}\sum_{n=1}^{N}\left(a_{m,n}^l - \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{NIW}\top}\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\right)^2\right). \end{cases} \tag{6}$$

The superscript 'NIW' stands for 'No Importance Weight'. However, since policies are updated in each RWR iteration, $\{\mathcal{D}^l\}_{l=1}^L$ generally follow different distributions induced by different policies and, therefore, the naive use of Eq.(5) will result in an inconsistent estimator.

Importance sampling (Bishop, 2006) can be used for coping with this problem. The basic idea of importance sampling is to weight the samples drawn from a different distribution to match the target distribution. More specifically, from i.i.d. (*independent and identically distributed*) samples $\{d_m\}_{m=1}^M$ following $P(d;\boldsymbol{\theta}_l)$, the expectation of a function $g(d)$ over another probability density function $P(d;\boldsymbol{\theta}_L)$ can be estimated in a consistent manner by the *importance-weighted average*:

$$\begin{aligned} \frac{1}{M}\sum_{m=1}^{M} g(d_m)\frac{P(d_m;\boldsymbol{\theta}_L)}{P(d_m;\boldsymbol{\theta}_l)} &\xrightarrow{M\to\infty} \mathop{\mathbb{E}}_{P(d;\boldsymbol{\theta}_l)}\left[g(d)\frac{P(d;\boldsymbol{\theta}_L)}{P(d;\boldsymbol{\theta}_l)}\right] \\ &= \int g(d)\frac{P(d;\boldsymbol{\theta}_L)}{P(d;\boldsymbol{\theta}_l)}P(d;\boldsymbol{\theta}_l)\mathrm{d}d = \mathop{\mathbb{E}}_{P(d;\boldsymbol{\theta}_L)}\left[g(d)\right]. \end{aligned}$$

The ratio of two densities $P(d;\boldsymbol{\theta}_L)/P(d;\boldsymbol{\theta}_l)$ is called the *importance weight* for an episode d.

2.6 Episodic Importance Sampling in RWR

This importance sampling technique can be employed in RWR for obtaining a consistent estimator $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{EIW}} \equiv (\widehat{\boldsymbol{k}}_{L+1}^{\mathrm{EIW}\top}, \widehat{\sigma}_{L+1}^{\mathrm{EIW}})^\top$, where

$$\begin{cases} \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{EIW}} = \left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\mathcal{R}(d_m^l)w_{L,l}(d_m^l)\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)^\top\right)^{-1} \\ \qquad \times \left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\mathcal{R}(d_m^l)w_{L,l}(d_m^l)\frac{1}{N}\sum_{n=1}^{N}a_{m,n}^l\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\right), \\ (\widehat{\sigma}_{L+1}^{\mathrm{EIW}})^2 = \left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\mathcal{R}(d_m^l)w_{L,l}(d_m^l)\right)^{-1} \\ \qquad \times \left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\mathcal{R}(d_m^l)w_{L,l}(d_m^l)\frac{1}{N}\sum_{n=1}^{N}\left(a_{m,n}^l - \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{EIW}\top}\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\right)^2\right). \end{cases} \tag{7}$$

Here, $w_{L,l}(d)$ denotes the importance weight defined by

$$w_{L,l}(d) \equiv \frac{P(d;\boldsymbol{\theta}_L)}{P(d;\boldsymbol{\theta}_l)}.$$

The superscript 'EIW' stands for 'Episodic Importance Weight'. According to Eq.(1), the two probability densities $P(d;\boldsymbol{\theta}_L)$ and $P(d;\boldsymbol{\theta}_l)$ both contain $P_{\mathrm{I}}(\boldsymbol{s}_1)$ and $\{P_{\mathrm{T}}(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n,a_n)\}_{n=1}^N$, which are often unknown. However, since they cancel out in the importance weight, we can compute the importance weight without the knowledge of $P_{\mathrm{I}}(\boldsymbol{s})$ and $P_{\mathrm{T}}(\boldsymbol{s}'|\boldsymbol{s},a)$ as

$$w_{L,l}(d) = \frac{\prod_{n=1}^{N}\pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta}_L)}{\prod_{n=1}^{N}\pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta}_l)}.$$

Although the importance-weighted estimator $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{EIW}}$ is guaranteed to be consistent, it tends to have a larger variance (Shimodaira, 2000; Sugiyama & Müller, 2005; Sugiyama et al., 2007). Therefore, the importance-weighted estimator tends to be unstable when the number of episodes M is rather small.

3 Adaptive Importance Sampling for Stable Policy Search

In this section, we propose a new policy search method called *Reward-weighted Regression with sample Reuse* (R^3) for effective sample reuse.

3.1 Per-Decision Importance-Weight

In Precup et al. (2000), a more effective importance weighting technique called the *per-decision importance-weight* (PIW) method was proposed. A crucial observation in PIW

is that the reward at the nth step does not depend on future state-action transitions after the nth step. Then an episodic importance weight can be decomposed into step-wise importance weights. For instance, the expected return $J(\boldsymbol{\theta}_L)$ can be expressed (see Appendix C for details) as

$$J(\boldsymbol{\theta}_L) = \int \Big(\sum_{n=1}^{N} \gamma^{n-1} r_n\Big) w_{L,l}(d) P(d;\boldsymbol{\theta}_l)\mathrm{d}d = \int \Big(\sum_{n=1}^{N} \gamma^{n-1} r_n w_{L,l}^n(d)\Big) P(d;\boldsymbol{\theta}_l)\mathrm{d}d,$$

where $w_{L,l}^n(d)$ is an n-step importance weight defined as

$$w_{L,l}^n(d) = \frac{\prod_{n'=1}^{n} \pi(a_{n'}|\boldsymbol{s}_{n'};\boldsymbol{\theta}_L)}{\prod_{n'=1}^{n} \pi(a_{n'}|\boldsymbol{s}_{n'};\boldsymbol{\theta}_l)}.$$

Here, we apply the PIW idea to episodic importance sampling in RWR and develop a more stable algorithm. The policy update formula in the sample-reuse RWR contains nested sums over N steps. For example, if $\mathcal{R}(d)$ is expanded in Eq.(7), we have

$$\sum_{n=1}^{N}\sum_{n'=1}^{N} \gamma^{n-1} r_n \phi(\boldsymbol{s}_{n'})\phi(\boldsymbol{s}_{n'}).$$

The summand $\gamma^{n-1} r_n \phi(\boldsymbol{s}_{n'})\phi(\boldsymbol{s}_{n'})$ does not depend on future state-action pairs after the $\widetilde{n}$th step, where

$$\widetilde{n} \equiv \max(n, n').$$

Thus, the episodic importance weight for $\gamma^{n-1} r_n \phi(\boldsymbol{s}_{n'})\phi(\boldsymbol{s}_{n'})$ can be simplified to the stepwise importance weights. Then Eq.(7) is simplified without loss of generality as

$$\begin{cases} \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{PIW}} = \left(\dfrac{1}{L}\displaystyle\sum_{l=1}^{L}\dfrac{1}{M}\sum_{m=1}^{M}\dfrac{1}{N}\Big(\sum_{n=1}^{N}\sum_{n'=1}^{N}\gamma^{n-1} r_n \boldsymbol{\phi}(\boldsymbol{s}_{m,n'}^l)\boldsymbol{\phi}(\boldsymbol{s}_{m,n'}^l)^\top w_{L,l}^{\widetilde{n}}(d_m^l)\Big)\right)^{-1} \\ \qquad \times \left(\dfrac{1}{L}\displaystyle\sum_{l=1}^{L}\dfrac{1}{M}\sum_{m=1}^{M}\dfrac{1}{N}\Big(\sum_{n=1}^{N}\sum_{n'=1}^{N}\gamma^{n-1} r_n a_{m,n'}^l \boldsymbol{\phi}(\boldsymbol{s}_{m,n'}^l) w_{L,l}^{\widetilde{n}}(d_m^l)\Big)\right), \\ (\widehat{\sigma}_{L+1}^{\mathrm{PIW}})^2 = \left(\dfrac{1}{L}\displaystyle\sum_{l=1}^{L}\dfrac{1}{M}\sum_{m=1}^{M}\Big(\sum_{n=1}^{N}\gamma^{n-1} r_n w_{L,l}^n(d_m^l)\Big)\right)^{-1} \\ \quad \times \left(\dfrac{1}{L}\displaystyle\sum_{l=1}^{L}\dfrac{1}{M}\sum_{m=1}^{M}\dfrac{1}{N}\Big(\sum_{n=1}^{N}\sum_{n'=1}^{N}\gamma^{n-1} r_n \Big(a_{m,n'}^l - \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{PIW}\top}\boldsymbol{\phi}(\boldsymbol{s}_{m,n'}^l)\Big)^2 w_{L,l}^{\widetilde{n}}(d_m^l)\Big)\right). \end{cases} \tag{8}$$

This PIW estimator $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{PIW}} \equiv (\widehat{\boldsymbol{k}}_{L+1}^{\mathrm{PIW}\top}, \widehat{\sigma}_{L+1}^{\mathrm{PIW}})^\top$ is consistent and potentially more stable than the plain EIW estimator $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{EIW}}$.

3.2 Adaptive Importance Weight

To more actively control the stability of the PIW estimator, we propose to use *adaptive importance weighting*—an importance weight $w_{L,l}^{\widetilde{n}}(d)$ is 'flattened' by *flattening parameter* ν ($\in [0,1]$) as $\left(w_{L,l}^{\widetilde{n}}(d)\right)^{\nu}$, i.e., the νth power of the importance weight. Then we have $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}} \equiv (\widehat{\boldsymbol{k}}_{L+1}^{\mathrm{AIW}\top}, \widehat{\sigma}_{L+1}^{\mathrm{AIW}})^{\top}$, where

$$\begin{cases} \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{AIW}} = \left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\left(\sum_{n=1}^{N}\sum_{n'=1}^{N}\gamma^{n-1}r_n\boldsymbol{\phi}(\boldsymbol{s}_{m,n'}^l)\boldsymbol{\phi}(\boldsymbol{s}_{m,n'}^l)^{\top}\left(w_{L,l}^{\widetilde{n}}(d_m^l)\right)^{\nu}\right)\right)^{-1} \\ \qquad \times\left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\left(\sum_{n=1}^{N}\sum_{n'=1}^{N}\gamma^{n-1}r_n a_{m,n'}^l\boldsymbol{\phi}(\boldsymbol{s}_{m,n'}^l)\left(w_{L,l}^{\widetilde{n}}(d_m^l)\right)^{\nu}\right)\right), \\ (\widehat{\sigma}_{L+1}^{\mathrm{AIW}})^2 = \left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\left(\sum_{n=1}^{N}\gamma^{n-1}r_n\left(w_{L,l}^{n}(d_m^l)\right)^{\nu}\right)\right)^{-1} \\ \quad\times\left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\left(\sum_{n=1}^{N}\sum_{n'=1}^{N}\gamma^{n-1}r_n\left(a_{m,n'}^l-\widehat{\boldsymbol{k}}_{L+1}^{\mathrm{AIW}\top}\boldsymbol{\phi}(\boldsymbol{s}_{m,n'}^l)\right)^2\left(w_{L,l}^{\widetilde{n}}(d_m^l)\right)^{\nu}\right)\right). \end{cases} \tag{9}$$

'AIW' stands for 'Adaptive Importance Weight'. When $\nu = 0$, AIW is reduced to NIW. Therefore, it is relatively stable, but is not consistent. On the other hand, when $\nu = 1$, AIW is reduced to PIW. Therefore, it is consistent, but is rather unstable. In practice, an intermediate ν often produces a better estimator. Note that the value of the flattening parameter can be different for each $\mathcal{D}^l$. However, for simplicity, we employ a single common value ν.

3.3 Automatic Selection of Flattening Parameter

The flattening parameter allows us to control the trade-off between consistency and stability. Here, we show how the value of the flattening parameter can be optimally chosen using data samples.

The goal of policy search is to find the optimal policy that maximizes the expected return $J(\boldsymbol{\theta})$. Therefore, the optimal flattening parameter value ν_L^* at the Lth iteration is given by

$$\nu_L^* \equiv \arg\max_{\nu} J(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu)). \tag{10}$$

Directly obtaining ν_L^* requires to compute the expected return $J(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu))$ for each candidate of ν. To this end, we need to collect data samples following $\pi(a|\boldsymbol{s};\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu))$ for each ν, which is prohibitively expensive. In order to reuse samples generated by previous policies, we propose to use a variation of cross-validation based on *importance-weighted cross-validation* (IWCV) (Sugiyama et al., 2007).

The basic idea of IWCV is to split the training dataset $\mathcal{D}^{1:L} \equiv \{\mathcal{D}^l\}_{l=1}^L$ into an 'estimation part' and a 'validation part'. Then the policy parameter $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu)$ is learned from the estimation part and its expected return $J(\widehat{\boldsymbol{\theta}}^{\mathrm{AIW}}(\nu))$ is approximated using the

importance-weighted loss for the validation part. As we pointed out in Section 2.6, importance weighting tends to be unstable when the number M of episodes is small. So we use per-decision importance weighting for cross-validation. Below, we explain in more detail how we apply IWCV to the selection of the flattening parameter ν in the current context.

Let us divide the training dataset $\mathcal{D}^{1:L} = \{\mathcal{D}^l\}_{l=1}^L$ into K disjoint subsets $\{\mathcal{D}_k^{1:L}\}_{k=1}^K$ of the same size[1], where each $\mathcal{D}_k^{1:L}$ contains M/K episodic samples from every $\mathcal{D}^l$. Let $\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}(\nu)$ be the policy parameter learned from $\{\mathcal{D}_{k'}^{1:L}\}_{k' \neq k}$ (i.e., without $\mathcal{D}_k^{1:L}$) by AIW estimation. We estimate the expected return of $\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}(\nu)$ using the PIW estimator from $\mathcal{D}_k^{1:L}$ as

$$\widehat{J}_{\mathrm{IWCV}}^k(\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}(\nu)) \equiv \frac{1}{\eta} \sum_{d^l \in \mathcal{D}_k^{1:L}} \sum_{n=1}^{N} \gamma^{n-1} r_n^l w_{L,l}^n(d^l), \tag{11}$$

where η is a normalization constant. An ordinary choice of η would be $\eta = LM/K$, but we use a 'stable' variant $\eta \equiv \sum_{d^l \in \mathcal{D}_k^{1:L}} w_{L,l}^n(d^l)$ (Precup et al., 2000).

The above procedure is repeated for all $k = 1, 2, \ldots, K$ and the average score is computed:

$$\widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu)) \equiv \frac{1}{K} \sum_{k=1}^{K} \widehat{J}_{\mathrm{IWCV}}^k(\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}(\nu)).$$

This is the K-fold IWCV estimate of $J(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu))$, which was shown to be almost unbiased (Sugiyama et al., 2007).

We compute this K-fold IWCV score for each candidate value of the flattening parameter ν and choose the one that maximizes the IWCV score:

$$\widehat{\nu}_{\mathrm{IWCV}} \equiv \arg\max_{\nu} \widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu)).$$

This IWCV scheme can also be used for choosing the basis functions $\boldsymbol{\phi}(\boldsymbol{s})$ in the Gaussian policy model (4).

Note that when the importance weights $w_{L,l}^{\widetilde{n}}$ are all one (i.e., no importance weighting), the above IWCV procedure is reduced to the ordinary CV procedure. The use of IWCV is essential here since the target policy $\pi(a|s; \widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu))$ is usually different from the previous policies used for collecting the data samples $\mathcal{D}^{1:L}$, so the expected return estimated using ordinary CV, $\widehat{J}_{\mathrm{CV}}(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu))$, would be heavily biased.

3.4 Algorithm

So far, we introduced the AIW method to control the stability of policy-parameter update, and IWCV to automatically choose the flattening parameter based on the estimated expected return. Here we show how these two methods are combined and implemented in

[1]For simplicity, we assume that M is divisible by K, i.e., M/K is an integer. We use $K = 5$ in the experiments.

a single algorithm. We call the proposed method *Reward weighted Regression with sample Reuse* (R^3). Figure 4 depicts the pseudo code of R^3.

In each iteration ($L = 1, 2, \ldots$), episodic data samples $\mathcal{D}^L$ are collected following the current policy $\pi(a|s; \boldsymbol{\theta}_L^{\mathrm{AIW}})$, the flattening parameter is chosen so as to maximize the expected return $\widehat{J}_{\mathrm{IWCV}}(\nu)$ estimated by IWCV using $\{\mathcal{D}^l\}_{l=1}^L$, and then the policy parameter is updated by Eq.(9) using $\{\mathcal{D}^l\}_{l=1}^L$.

4 Experiments

In this section, we evaluate the performance of the proposed method through experiments.

4.1 Numerical Examples

First, we illustrate how the proposed method behaves on a one-dimensional ball-balancing simulation illustrated in Figure 5.

The goal is to control the angle of the seesaw so that the ball is brought to the middle of the seesaw. The action space $\mathcal{A}$ consists of the angle α ($\in (-\pi/4, \pi/4)$) [rad] of the seesaw, which is one-dimensional and continuous. The state space $\mathcal{S}$ is also continuous and a state vector $\boldsymbol{s} = (x, \dot{x})^\top$ consists of the position x [m] of the ball on the seesaw (the middle of the seesaw is the origin and the left/right side of the origin corresponds the negative/positive direction), and the velocity $\dot{x}$ [m/s] of the ball (moving to the left/right corresponds to the negative/positive velocity).

The position x and velocity $\dot{x}$ are modeled by the following equations:

$$
\begin{aligned}
x_{t+1} &= x_t + \dot{x}_{t+1}\Delta t, \\
\dot{x}_{t+1} &= \dot{x}_t + \Delta t \left(-\frac{f}{m}\dot{x}_t - 9.8\sin(a_t) \right),
\end{aligned}
$$

where $f = 0.01$ is the friction coefficient, $m = 3$ [kg] is the mass of the ball, a_t [rad] is the action chosen at time t, and $\Delta t = 0.05$ [s] is the duration of a time step. We assume that if an action a_t is chosen, the seesaw angle will be adjusted to the desired angle in a single time-step (this simulation is for illustration purposes; more realistic experiments will be shown in Section 4.2).

The reward function $R(s, a, s')$ is a bell-shaped function defined as

$$
R(\boldsymbol{s}, a, \boldsymbol{s}') = \exp\left(-\frac{x'^2 + 0.5\dot{x}'^2 + 0.1a^2}{2} \right).
$$

This reward function indicates that the agent will receive the maximum reward (i.e., one) when the ball stops at the middle of the seesaw ($\boldsymbol{s} = \boldsymbol{0}$ and $a = 0$). We use 10 Gaussian kernels as basis functions $\boldsymbol{\phi}(\boldsymbol{s})$ of the Gaussian policy model (4):

$$
\boldsymbol{\phi}_b(\boldsymbol{s}) = \exp\left(-\frac{\|\boldsymbol{s} - \boldsymbol{c}_b\|^2}{2\sigma_{\mathrm{basis}}^2} \right), \tag{12}
$$

Algorithm 3.1: $\mathrm{R}^3(K, M, N, \boldsymbol{\phi}, \widehat{\boldsymbol{\theta}}_1^{\mathrm{AIW}}, \epsilon)$

$//K$ Number of folds
$//M$ Number of episodes collected at each iteration
$//N$ Number of steps in each episode
$//\boldsymbol{\phi}$ Basis functions, $\boldsymbol{\phi}(s) = (\phi_1(s), \phi_2(s), \ldots, \phi_B(s))^\top$
$//\widehat{\boldsymbol{\theta}}_1^{\mathrm{AIW}}$ Initial policy parameter, $\widehat{\boldsymbol{\theta}}_1^{\mathrm{AIW}} = (\widehat{\boldsymbol{k}}_1^{\mathrm{AIW}\top}, \widehat{\sigma}_1^{\mathrm{AIW}})^\top$
$//\epsilon$ Stopping criterion

$L \leftarrow 0$
repeat
  $L \leftarrow L + 1$
  // Collect data samples (M episodes and N steps)
  // using current policy $\pi(a|s; \widehat{\boldsymbol{\theta}}_L^{\mathrm{AIW}})$
  $\mathcal{D}^L \leftarrow \textsc{DataSampling}(\widehat{\boldsymbol{\theta}}_L^{\mathrm{AIW}}, M, N)$
  // Choose flattening parameter $\widehat{\nu}_{\mathrm{IWCV}}$ that maximizes $\widehat{J}_{\mathrm{IWCV}}(\nu)$
  for $\nu \leftarrow 0, 0.1, \ldots, 1$
    for $k \leftarrow 1, 2, \ldots, K$
      // Learn parameter $\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}$ from $K-1$ groups of data samples $\{\mathcal{D}_{k'}^{1:L}\}_{k' \neq k}$
      $\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}} \leftarrow \textsc{PolicyParameterUpdate}(\{\mathcal{D}_{k'}^{1:L}\}_{k' \neq k}, \nu, \boldsymbol{\phi})$
      // Estimate expected return from kth group of data samples $\{\mathcal{D}_{k'}^{1:L}\}_{k'=k}$
      $\widehat{J}_{\mathrm{IWCV}}^k(\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}(\nu)) = \frac{1}{\eta} \sum_{d^l \in \mathcal{D}_k^{1:L}} \sum_{n=1}^N \gamma^{n-1} r_n^l w_{L,l}^n(d^l)$
    // Compute the mean of expected return
    $\widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu)) \leftarrow \frac{1}{K} \sum_{k=1}^K \widehat{J}_{\mathrm{IWCV}}^k(\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}(\nu))$
  $\widehat{\nu}_{\mathrm{IWCV}} \leftarrow \arg\max_{\nu} \widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu))$
  // Update policy parameter $\widehat{\boldsymbol{\theta}}_L^{\mathrm{AIW}}$ by Eq.(9)
  $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}} \leftarrow \textsc{PolicyParameterUpdate}(\{\mathcal{D}^l\}_{l=1}^L, \widehat{\nu}_{\mathrm{IWCV}}, \boldsymbol{\phi})$
until $\|\widehat{\boldsymbol{\theta}}_{L+1} - \widehat{\boldsymbol{\theta}}_L\| \leq \epsilon$
return $(\widehat{\boldsymbol{\theta}}_{L+1})$

Figure 4: Pseudo code of the proposed R^3 method (Reward-weighted Regression with sample Reuse). By the DataSampling function, data samples (M episodes and N steps) are collected following current policy $\pi(a|\boldsymbol{s}; \widehat{\boldsymbol{\theta}}_L^{\mathrm{AIW}})$. By the PolicyParameterUpdate function, policy parameters are updated using datasets $\{\mathcal{D}^l\}_{l=1}^L$, flattening parameter $\widehat{\nu}_{\mathrm{IWCV}}$, and basis functions $\boldsymbol{\phi}$ by Eq.(9).

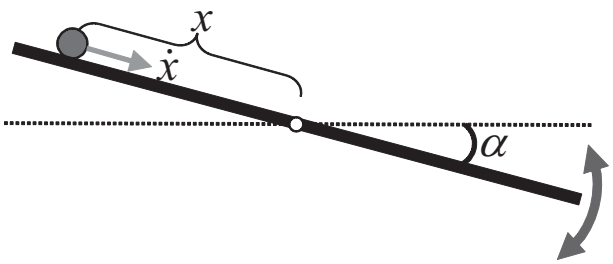


Figure 5: Illustration of one-dimensional ball-balancing simulation. The goal is to control the angle α of the seesaw so that the ball is moved to and kept at the middle of the seesaw.

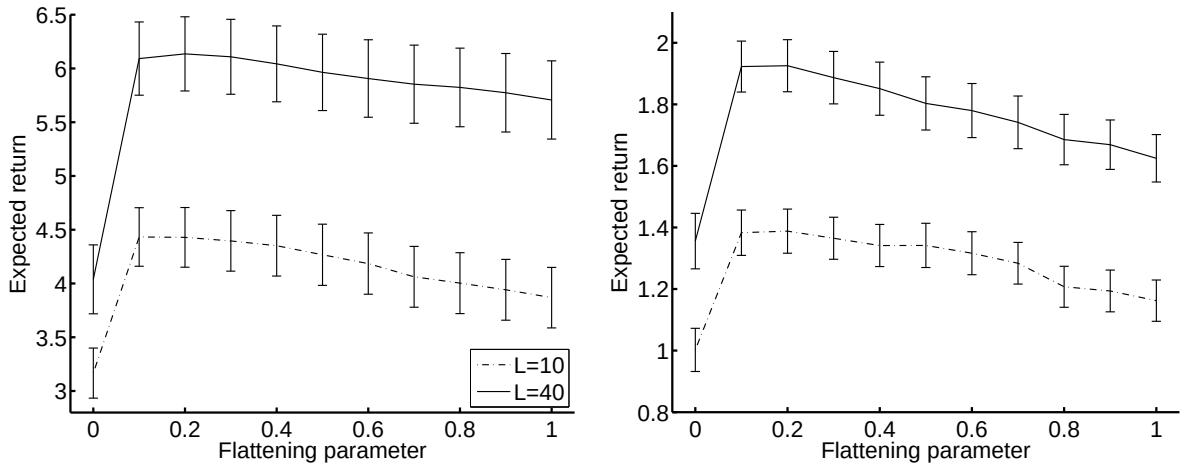
where $\boldsymbol{c}_b$ is a randomly located kernel center and $\sigma_{\mathrm{basis}} = 1$ is the Gaussian width.

The initial policy-parameter $\widehat{\boldsymbol{\theta}}_1^{\mathrm{AIW}}$ is set randomly as $\widehat{\boldsymbol{k}}_1^{\mathrm{AIW}} = (\beta_1, \beta_2, \ldots, \beta_{10})^\top$ and $\widehat{\sigma}_1^{\mathrm{AIW}} = 0.5$, where $\beta_1, \beta_2, \ldots, \beta_{10}$ are independently drawn from the uniform distribution on $[-0.2, 0.2]$. At every iteration, the agent collects samples $\mathcal{D}^L$ following the current policy $\pi(a|\boldsymbol{s}; \widehat{\boldsymbol{\theta}}_L^{\mathrm{AIW}})$, and then the policy parameter $\widehat{\boldsymbol{\theta}}_L^{\mathrm{AIW}}$ is updated using all samples $\{\mathcal{D}^l\}_{l=1}^L$. This is repeated for 50 iterations.

Our original motivation for introducing R^3 was to reduce the number of samples for saving the sampling cost. To investigate this, we set the number of episodes and the number of steps to small values: $M = 5$ and $N = 20$. The discount factor is set to $\gamma = 0.99$.

First, let us illustrate how the flattening parameter ν influences the policy-parameter update. For this purpose, we compute the expected return for the estimated parameter $\widehat{\boldsymbol{\theta}}_{10}^{\mathrm{AIW}}(\nu)$ and $\widehat{\boldsymbol{\theta}}_{40}^{\mathrm{AIW}}(\nu)$ at the 10th and 40th iterations from 50 test episodic data. Figure 6(a) depicts $J(\widehat{\boldsymbol{\theta}}_{10}^{\mathrm{AIW}}(\nu))$ and $J(\widehat{\boldsymbol{\theta}}_{40}^{\mathrm{AIW}}(\nu))$ averaged over 50 trials as a function of the flattening parameter. The graph overall shows that as the iteration progresses from 10th to 40th, the average expected returns become larger and neither NIW ($\nu = 0$) nor PIW ($\nu = 1$) is the best—intermediate values of ν (say, $0.1 \leq \nu \leq 0.2$) perform better than NIW and PIW on average. Thus, given that ν is chosen optimally, AIW can outperform PIW and NIW. Note that, although the amount of performance improvement over PIW and NIW gained by tuning the flattening parameter in AIW seems subtle in this one-iteration simulation, accumulation of this small gain over iterations can cause significant performance improvement as will be demonstrated below.

Next, we illustrate how IWCV behaves. Figure 6(b) depicts the expected returns estimated by 5-fold IWCV at the 10th and 40th iterations, $\widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}_{10}^{\mathrm{AIW}}(\nu))$ and $\widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}_{40}^{\mathrm{AIW}}(\nu))$, averaged over 50 trials. We can only access $\{\mathcal{D}^l\}_{l=1}^{10}$ and $\{\mathcal{D}^l\}_{l=1}^{40}$ in IWCV, which are collected through iterations. Thus, the total number of episodes used in IWCV is only 50 and 200 at the 10th and 40th iterations, respectively. The graph shows that IWCV well captures the trends of the true expected return for both cases ($L = 10$ and 40). Note that the orders of magnitude of the values in Figure 6(a) and Figure 6(b) are different due to the normalization constant η (see Eq. (11)) which is computed by the



(a) True expected return

(b) 5-fold IWCV estimate

Figure 6: True expected return $J(\widehat{\boldsymbol{\theta}}^{\mathrm{AIW}}_{L+1}(\nu))$ and its 5-fold IWCV estimate $\widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}^{\mathrm{AIW}}_{L+1}(\nu))$ averaged over 50 trials as a function of the flattening parameter ν in the one-dimensional ball-balancing task. The error bars indicate 1/10 of standard deviation.

sum of importance weights. However, this does not cause a problem in model selection as long as relative profiles of the curves are similar, which is achieved well in this experiment.

Finally, we illustrate how R^3 performs. Figure 7 depicts the expected returns averaged over 50 trials as a function of the number of iterations. The expected return at each trial is computed from 50 test episodic data for this evaluation. We compare seven scenarios: episodic REINFORCE (Williams, 1992; Peters & Schaal, 2006) with learning rate 0.1 or 0.3, ordinary RWR (no sample-reuse), sample-reuse RWR with fixed flattening parameter values: $\nu = 0$ (NIW), $\nu = 0.5$, or $\nu = 1$ (PIW), and R^3 where $\nu \in \{0, 0.1, 0.2, \ldots, 1\}$ is adaptively chosen by 5-fold IWCV at each iteration. The graph shows that R^3 outperforms REINFORCE, ordinary RWR, and all other sample-reuse RWR schemes with fixed flattening parameter values.

The performance of REINFORCE depends on the learning rate; when the learning rate is 0.005, policies are improved in a stable way. However, the speed of policy improvement tends to be slow since the amount of policy update at each step is rather small. when the learning rate is 0.1, policy improvement tends to be unstable since the number of data samples collected at each iteration is too small ($M = 5$) to take a large policy-update step in each iteration. Similarly, the policy update tends to be unstable for no-sample-reuse RWR. Consequently, the performance of updated policies is saturated after the 22nd iteration. When ν is fixed to 0 (NIW) or 0.5 through sample-reuse RWR iterations, the performance is improved in a stable manner without decay unlike ordinary RWR. However, the speed of performance improvement tends to be slower than R^3. When ν is fixed to 1 (PIW), the performance is not much improved over iterations. This indicates that the instability of the estimators severely degrades the performance. Furthermore, increasing the total number of episodes as the iteration progresses does not contribute to

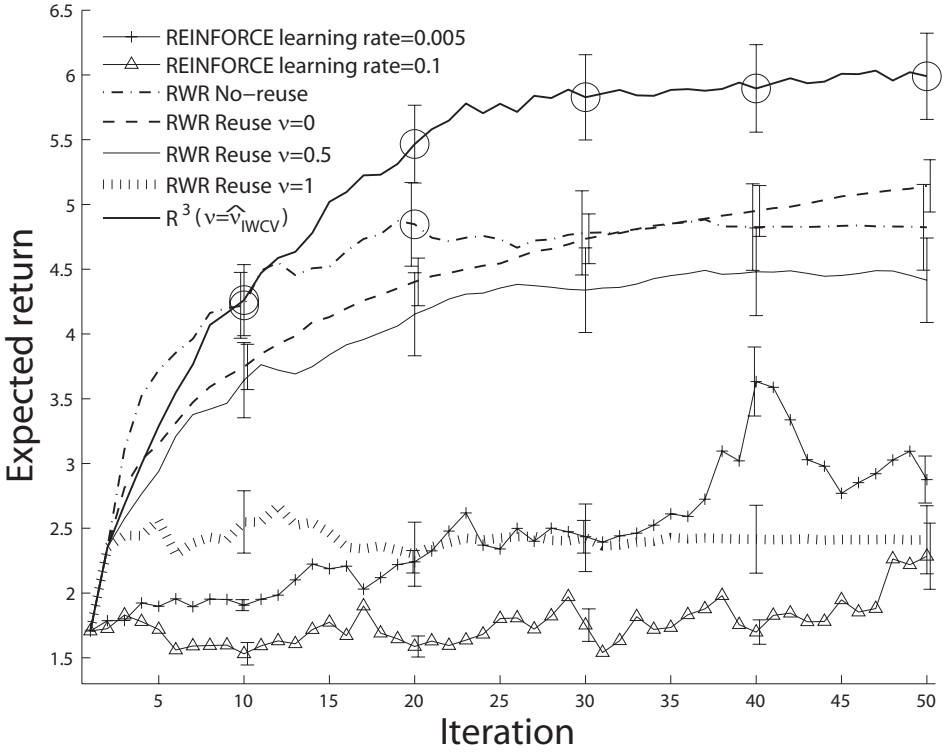
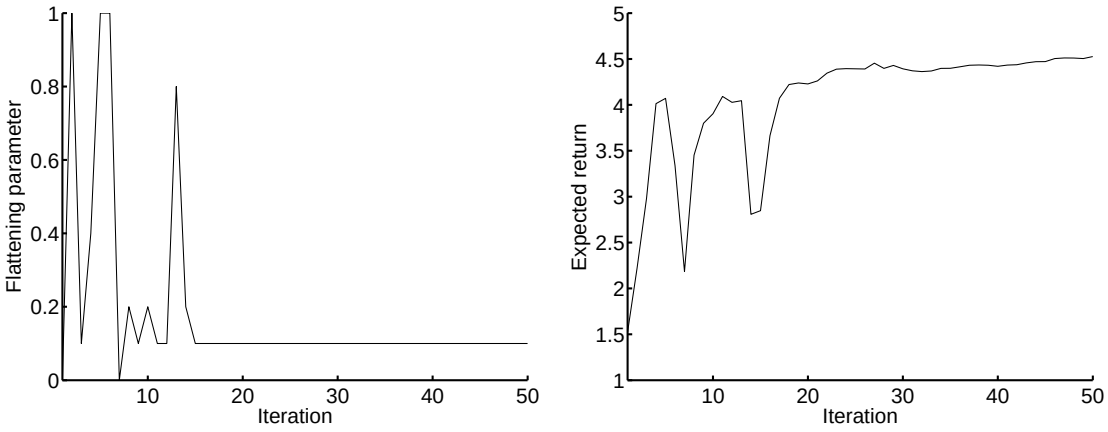


Figure 7: The expected returns averaged over 50 trials as a function of iterations in the one-dimensional ball-balancing task. We compare seven scenarios: episodic REINFORCE with learning rate 0.005 or 0.1, ordinary RWR (no sample-reuse), sample-reuse RWR with fixed flattening parameter values: $\nu = 0$ (NIW), $\nu = 0.5$, or $\nu = 1$ (PIW), and R^3 where $\nu \in \{0, 0.1, 0.2, \ldots, 1\}$ is adaptively chosen by 5-fold IWCV at each iteration. The symbol '$\circ$' indicates the fact that the method is the best or comparable to the best one in terms of the expected return by the *t-test* at the significance level 5%, performed at every 10 iterations. The error bars indicate 1/10 of standard deviation.

the performance improvement since the number of episodic samples taken from the same policy is still kept small (only $M = 5$). On the other hand, the proposed R^3 method achieves stable and fast policy improvement throughout iterations by adaptively turning the flattening parameter using IWCV. As a result, the performance of R^3 is significantly better than the other methods (the symbol '$\circ$' indicates the fact that the corresponding method is the best or comparable to the best one in terms of the mean performance by the *t-test* at the significance level 5%).

Figure 8(a) and Figure 8(b) depict an example of the flattening parameter value used in the R^3 iterations and its corresponding expected return. The graphs show that the value of the flattening parameter is rather large and changes strongly in earlier iterations. On the other hand, once the expected return converges (at around the 25th iteration), the flattening parameter value decreases to a lower value, 0.1, and tends to be stable. This would be a natural outcome because after the policy converges, the amount of policy



(a) Flattening parameter value (b) Expected return

Figure 8: An example of the behavior of the flattening parameter values in R^3 and its corresponding expected return as a function of iterations in the one-dimensional ball-balancing task.

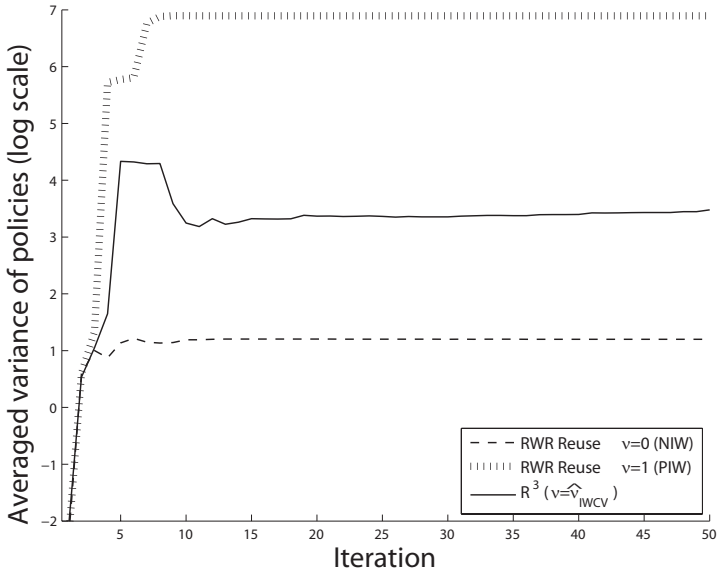


Figure 9: Averaged variance of the mean $\widehat{\boldsymbol{k}}_L^\top \boldsymbol{\phi}(\boldsymbol{s})$ of learned Gaussian policy over 50 trials as a function of iterations—the variance computed at each state is averaged over the set of states $\boldsymbol{s} = (x, \dot{x})^\top \in \{-5, -4.5, \ldots, 4.5, 5\} \times \{-5, -4.5, \ldots, 4.5, 5\}$.

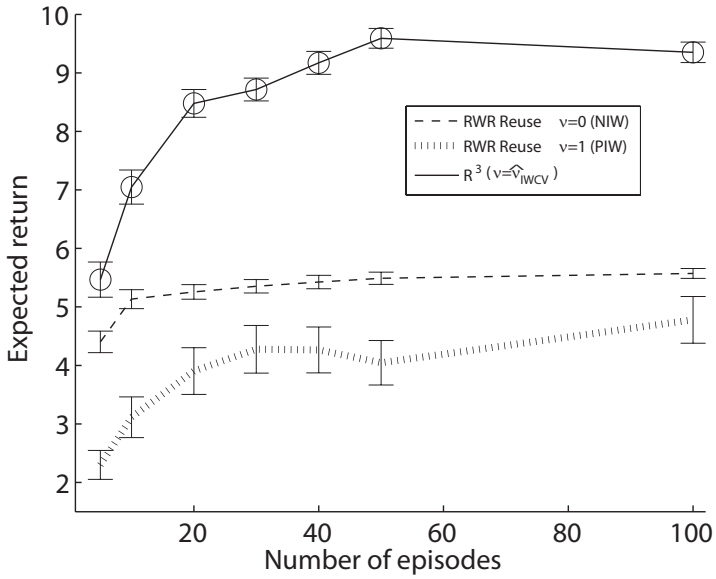


Figure 10: The expected returns at the 20th iteration averaged over 50 trials as a function of the number of episodes in the one-dimensional ball-balancing task. We consider three sample-reuse RWR for performance comparison: sample reuse RWR with fixed $\nu = 0, 1$, and R^3 where ν is chosen by IWCV. The symbol '$\circ$' indicates the fact that the method is the best or comparable to the best one in terms of the expected return by the *t-test* at the significance level 5%. The error bars indicate 1/10 of standard deviation.

update usually becomes small. Then data samples collected after convergence have similar distributions and, thus, can be used without importance weights.

Figure 9 depicts the variance of the mean $\widehat{\boldsymbol{k}}_L^\top \boldsymbol{\phi}(\boldsymbol{s})$ of the Gaussian policy model over 50 trials as a function of iterations—the variance at each state $\boldsymbol{s} = (x, \dot{x})^\top \in \{-5, -4.5, \ldots, 4.5, 5\} \times \{-5, -4.5, \ldots, 4.5, 5\}$ is computed and is averaged over all states.

The graph shows that the variance of learned policies by PIW increases significantly in the beginning of the learning process (from the 4th to 9th iteration) compared to NIW and R^3—the variance of PIW is about 500000 and 2600 times larger than the variance of NIW and R^3 at the 20th iteration, respectively. This indicates that policy update by PIW is unstable, i.e., policies are strongly changed due to the numerical instability of importance weights. On the other hand, policy update by NIW is highly stable because importance weights are not used. However, the direction of policy update may not be appropriate due to the inconsistency of NIW policy update. Indeed, the performance of policies learned by NIW is not largely improved over iterations (see Figure 7). The variance of learned policies by R^3 is in between the ones of PIW and NIW. This indicates that the consistency of PIW is slightly compromised to obtain more stable policies by adjusting flattening parameter in R^3. As a result, the performance of resulting policies is largely improved over iterations (see Figure 7).

Finally, we investigate how the number M of episodes affects the performance of

sample-reuse RWR methods. Figure 10 depicts the expected return at the 20th iteration as a function of the number M of episodes. The graph shows that the performance of PIW (ν is fixed to 1) gradually improves as the number of episodes increases. This indicates that the consistent property of the PIW estimator tends to take effect as the number of episodes increases. However, a huge number of episodes would be necessary to make PIW competitive to other methods. The performance of NIW (ν is fixed to 0) is saturated after M reaches 10. This indicates that the inconsistent nature of NIW cannot be overcome just by increasing the number M of episodes. On the other hand, our proposed method R^3 tends to perform better as the number M increases. We conjecture that the flattening parameter ν selected by IWCV can be more stable and previously collected data can be used more efficiently when the number M of episodes is large.

4.2 Robot-arm Ball-balancing Task

Next, we evaluate the performance of our proposed method R^3 in more challenging problems, i.e., a ball-balancing task using a Barrett WAM$^{\mathrm{TM}}$ robot arm and an Acrobot swing-up task.

We employ the *Simulation Laboratory (SL) simulator* (Schaal, 2009) for the ball-balancing experiment, which is known to be highly realistic. The 7-degree-of-freedom Barrett WAM$^{\mathrm{TM}}$ arm is mounted on the ceiling upside down and is equipped with a circular tray (the radius is 0.24 [m]) at the end-effector (see Figure 11). The goal is to control the joints of the robot so that the ball is brought to the middle of the tray, similarly to the toy ball-balancing task in Section 4.1. However, unlike before, the angle of the tray cannot be directly controlled; this is a typical restriction in the real joint-motion planning based on the feedback from the environment (e.g., the state of the ball). Thus, achieving the goal is much harder than the toy setup.

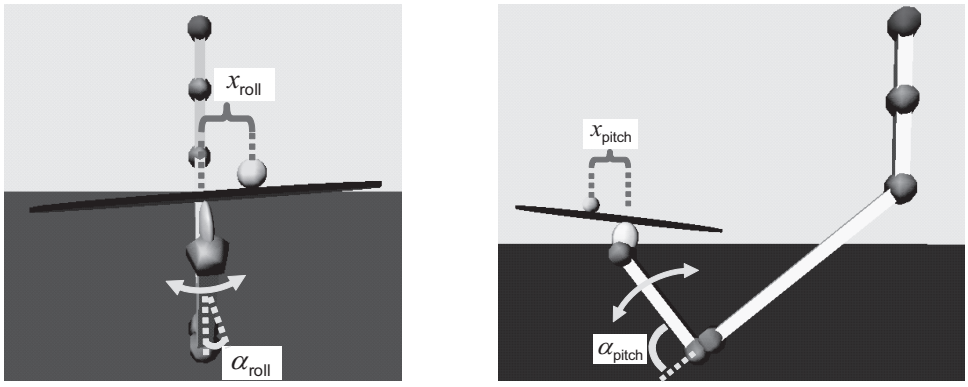
To simplify the problem, we control only two joints: the wrist angle α_{roll} and the elbow angle α_{pitch}. All the remaining joints are fixed. Control of the wrist and elbow angles would roughly correspond to changing the *roll* and *pitch* angles of the tray, but not directly.

We design two separate control subsystems, each of which is in charge of roll- and pitch-angle control. Each subsystem has its own policy parameter $\boldsymbol{\theta}_*$, state space $\mathcal{S}_*$, and action space $\mathcal{A}_*$, where '*' corresponds to 'roll' or 'pitch'. The state space $\mathcal{S}_*$ is continuous and consists of $(x_*, \dot{x}_*)$, where x_* [m] is the position of the ball on the tray along each axis and $\dot{x}_*$ [m/s] is the velocity of the ball. The action space $\mathcal{A}_*$ is continuous and corresponds to the target angle of the joint a_* [rad]. The reward function is defined as

$$R_*(\boldsymbol{s}_*, a_*, \boldsymbol{s}'_*) = \exp\left(-\frac{5(x'_*)^2 + (\dot{x}'_*)^2 + a_*^2}{2(0.24/2)^2}\right),$$

where the number 0.24 in the denominator comes from the radius of the tray (i.e., 0.24 [m]).

Below, we explain how the control system is designed in more detail. As illustrated in Figure 12, our control system has two feedback loops for trajectory planning using



(a) The angle of wrist (b) The angle of elbow

Figure 11: Realistic ball-balancing task using a Barrett WAM$^{\text{TM}}$ arm simulator. Two joints of the robots are controlled so as to keep the ball in the middle of the tray.

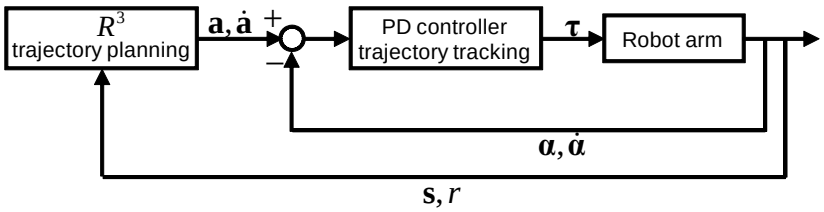


Figure 12: The block diagram of the Barrett WAM$^{\text{TM}}$ robot-arm control system for ball balancing. The control system has two feedback loops, i.e., joint-trajectory planning by R^3 and trajectory tracking by a high-gain proportional-derivative (PD) controller.

an R^3 controller and trajectory tracking using a high-gain *proportional-derivative* (PD) controller (Siciliano & Khatib, 2008). The R^3 controller outputs the target joint-angle obtained by current policy $\pi(a|\boldsymbol{s};\boldsymbol{\theta}_L)$ at every 0.2 [s]. 9 Gaussian kernels are used as basis functions $\phi(\boldsymbol{s})$ (see Eq.(12)) with the kernel centers $\{c_b\}_{b=1}^9$ located over the state-space at

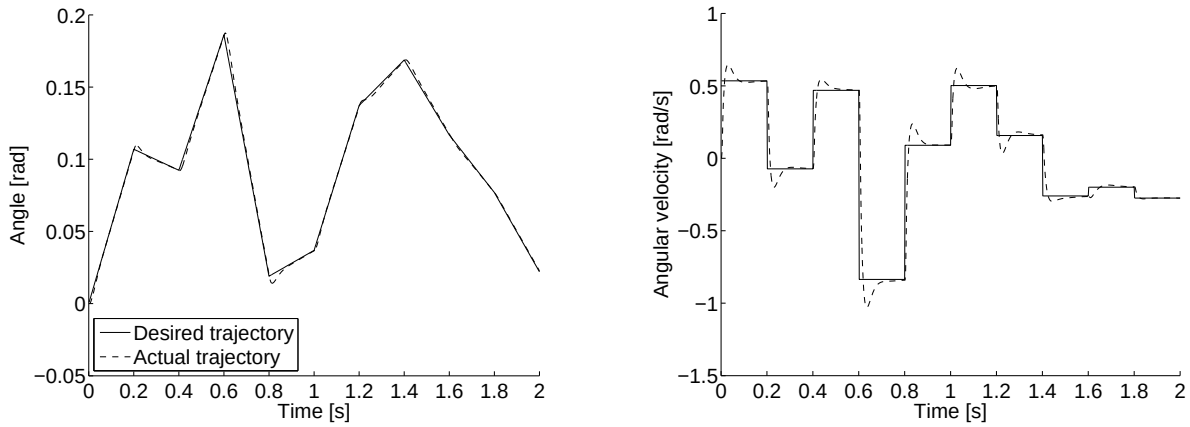
$$\begin{aligned}(x_*, \dot{x}_*) \in \{&(-0.2, -0.4), (-0.2, 0), (-0.1, 0.4),\\ &(0, -0.4), (0, 0), (0, 0.4),\\ &(0.1, -0.4), (0.2, 0), (0.2, 0.4)\}.\end{aligned}$$

The Gaussian width is set to $\sigma_{\text{basis}} = 0.1$. Based on the discrete-time target angles obtained by R^3, the desired joint-trajectory in the continuous time domain is linearly interpolated as

$$a_{n,t,*} = a_{n,*} + t\ \dot{a}_{n,*},$$

where t is the time from the last output $a_{n,*}$ of R^3 at the nth step. $\dot{a}_{n,*}$ is the angular velocity computed by

$$\dot{a}_{n,*} = \frac{a_{n,*} - a_{n-1,*}}{0.2},$$



(a) Trajectory of angle

(b) Trajectory of angular velocity

Figure 13: An example of desired and actual trajectories of the wrist joint in the realistic ball-balancing task. The target joint-angle is determined by a random policy at every 0.2 [s], and then a linearly-interpolated angle and constant velocity are tracked using the proportional-derivative (PD) controller in the cycle of 0.002 [s].

where $a_{0,*}$ is the initial angle of a joint. The angular velocity is assumed to be constant during the cycle of trajectory planning (i.e., 0.2 [s]).

On the other hand, the PD controller converts desired joint-trajectories to motor torques as

$$\boldsymbol{\tau}_{n,t} = \boldsymbol{k}_{\mathrm{p}}(\boldsymbol{a}_{n,t} - \boldsymbol{\alpha}_{n,t})^\top + \boldsymbol{k}_{\mathrm{d}}(\dot{\boldsymbol{a}}_n - \dot{\boldsymbol{\alpha}}_{n,t})^\top,$$

where $\boldsymbol{\tau}$ is the 2-dimensional vector consisting of the torque applied to the wrist and elbow joints. $\boldsymbol{a} = (a_{\mathrm{pitch}}, a_{\mathrm{roll}})^\top$ and $\dot{\boldsymbol{a}} = (\dot{a}_{\mathrm{pitch}}, \dot{a}_{\mathrm{roll}})^\top$ are the 2-dimensional vectors consisting of the desired angles and velocities. $\boldsymbol{\alpha} = (\alpha_{\mathrm{pitch}}, \alpha_{\mathrm{roll}})^\top$ and $\dot{\boldsymbol{\alpha}} = (\dot{\alpha}_{\mathrm{pitch}}, \dot{\alpha}_{\mathrm{roll}})^\top$ are the 2-dimensional vectors consisting of the current joint-angle and velocities. $\boldsymbol{k}_{\mathrm{p}}$ and $\boldsymbol{k}_{\mathrm{d}}$ are the 2-dimensional vectors consisting of the proportional and derivative gains, respectively. Since the control cycle of the Barrett WAM$^{\mathrm{TM}}$ arm is 0.002 [s], the PD controller is applied 100 times (i.e., $t = 0.002, 0.004, \ldots, 0.198, 0.2$) in each R^3 cycle.

Figure 13 depicts a desired trajectory of the wrist joint generated by a random policy and an actual trajectory obtained using the high-gain PD controller described above. The graph shows that the desired trajectory is followed by the robot reasonably well.

The policy parameter $\boldsymbol{\theta}_L$ is learned through the R^3 iterations. The initial policy parameters $\boldsymbol{\theta}_1 = (\boldsymbol{k}_1^\top, \sigma_1)^\top$ are set manually as

$$\boldsymbol{k}_* = (-0.5, -0.5, 0, -0.5, 0, 0, 0, 0, 0),$$

and $\sigma_* = 0.1$ so that various pairs of state and action can be safely explored in the first iteration. The initial position of the ball is randomly selected as $x_* \in [-0.05, 0.05]$ and the angle of elbow α_{pitch} is offset by $\frac{\pi}{2}$. The dataset collected in each iteration consists of 10 episodes with 20 steps. The duration of an episode is 4 [s] and the sampling cycle by R^3 is 0.2 [s].

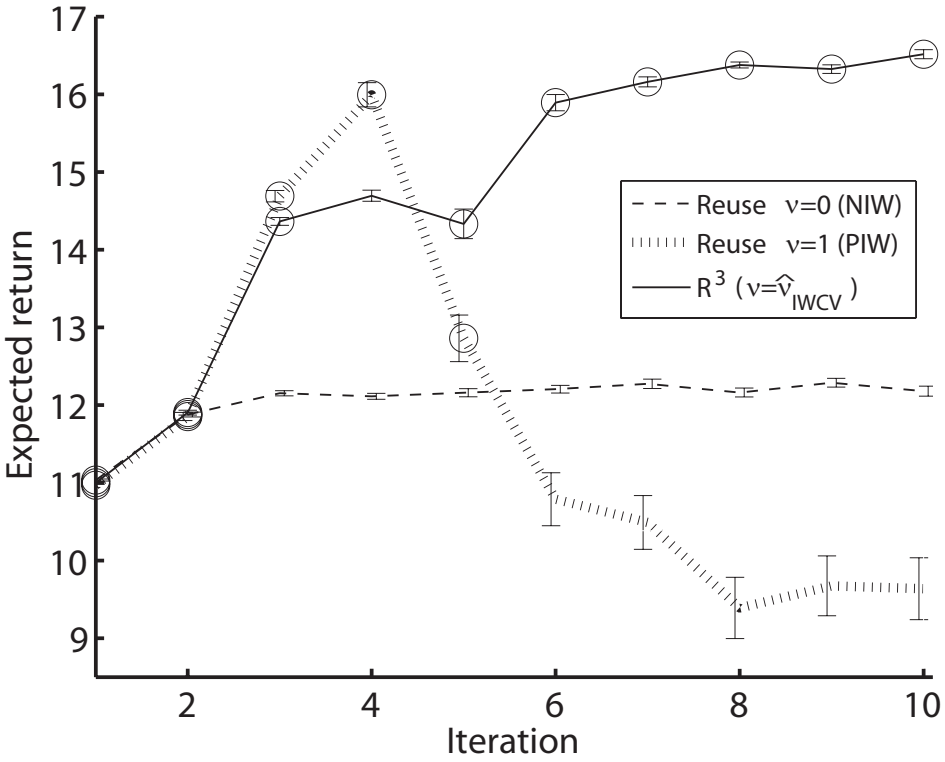


Figure 14: The performance of learned policies when $\nu = 0$ (NIW), $\nu = 1$ (PIW), and ν is chosen by IWCV (R^3) in ball balancing using a simulated Barrett WAM$^{\mathrm{TM}}$ arm system. The performance is measured by the return averaged over 10 trials. The symbol '$\circ$' indicates the fact that the method is the best or comparable to the best one in terms of the expected return by the *t-test* at the significance level 5%, performed at every iteration. The error bars indicate 1/10 of standard deviation.

We consider three scenarios here: sample reuse with fixed $\nu = 0$ (NIW), $\nu = 1$ (PIW), and the proposed R^3 (ν is chosen by IWCV from $\{0, 0.25, 0.5, 0.75, 1\}$ at every iteration). The discount factor is set to $\gamma = 0.99$. Figure 14 depicts the averaged expected return over 10 trials as a function of the number of RWR iterations. The expected return at each trial is computed from 20 test episodic data for this evaluation. The graph shows that R^3 nicely improves the performance over iterations. On the other hand, the performance using fixed $\nu = 0$ is saturated after the 3rd iteration. The performance using fixed $\nu = 1$ was improved much in the beginning but suddenly goes down at 5th iteration. This shows that large change of policies would cause the severe instability of sample reuse with fixed $\nu = 1$.

Figure 15 depicts examples of trajectories of the wrist angle α_{roll}, the elbow angle α_{pitch}, resulting ball movement x_*, and reward r_*, when following policies learned by NIW ($\nu = 0$) and R^3 (ν is chosen by IWCV) at the 10th iteration. When following the policy learned by NIW, the ball goes through the middle of the tray $(x_{\mathrm{roll}}, x_{\mathrm{pitch}}) = (0, 0)$. On

Table 1: Parameter setting of the Acrobot swing-up problem.

Parameters	Values
Length of link 1 (l_1)	0.3 [m]
Length of link 2 (l_2)	0.6 [m]
Mass of link 1 (m_1)	6.4 [kg]
Mass of link 2 (m_2)	11.1 [kg]
Centroid of link 1 (c_1)	0.15 [m]
Centroid of link 2 (c_2)	0.3 [m]
Moment of inertia of link 1 (I_1)	0.048
Moment of inertia of link 2 (I_2)	0.33

the other hand, the policy learned by R^3 successfully guides the ball to the middle of the tray along the roll axis though the movement along the pitch axis is almost the same as NIW.

4.3 Acrobot Swing-up Task

We next evaluate our proposed method on a highly nonlinear control problem of Acrobot swing-up using a simulator based on *open dynamics engine* (Smith, 2005).

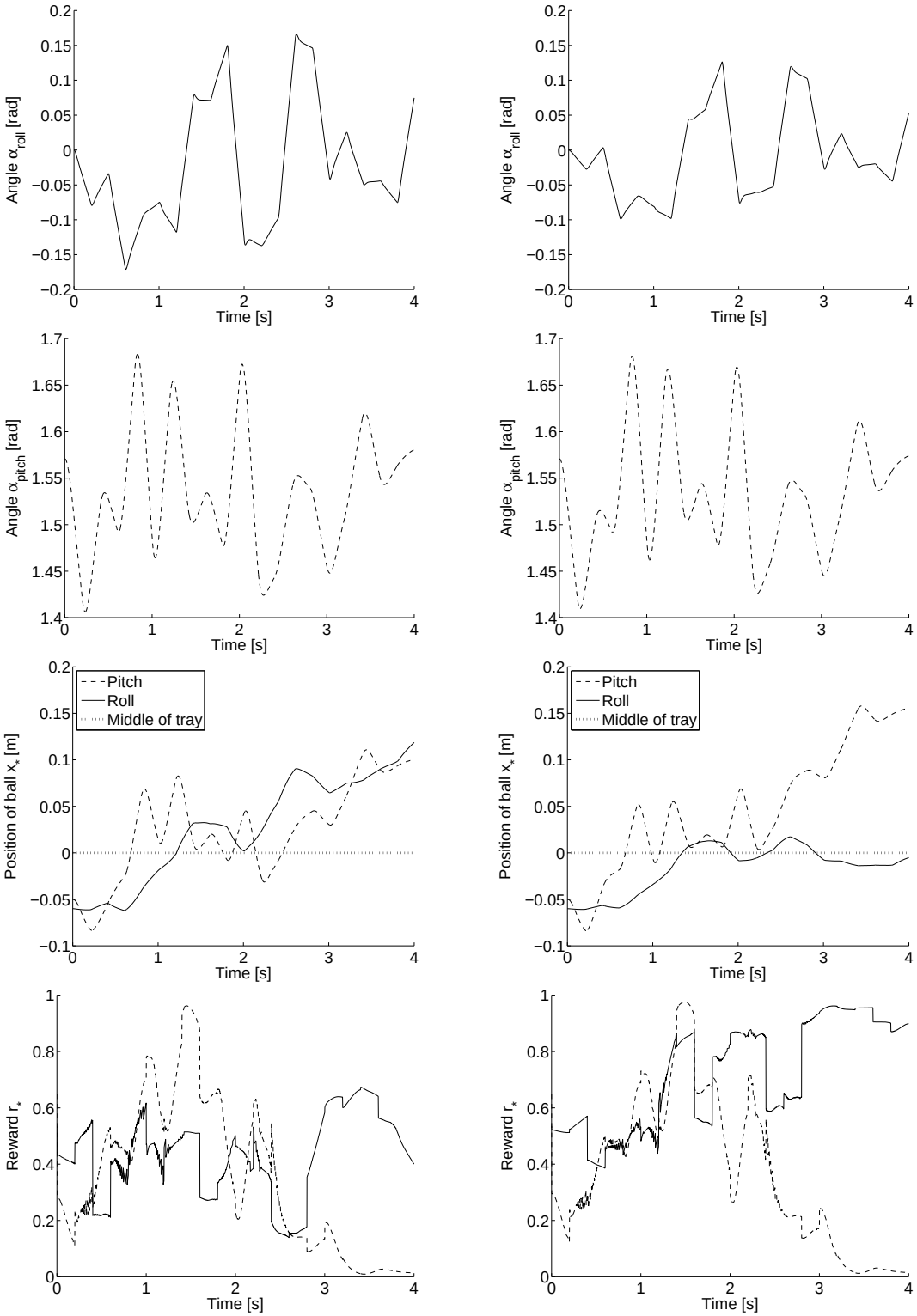
The Acrobot is a well-known under-actuated robotic system where among two joints (shoulder and elbow), only the elbow joint can be controlled; the shoulder joint freely rotates in response to external forces such as inertia force and gravity. The robot is set upside down as illustrated in Figure 16(a). The goal is to control the elbow joint so that the entire body is swung up to the top. This task is known to be very hard because the non-actuated shoulder joint has to be controlled indirectly through the manipulation of the elbow joint, and the system dynamics of each joint is highly nonlinear (Spong, 1995). The detailed settings of the robot are described in Figure 16(b) and Table 1.

The state space $\mathcal{S}$ is continuous and consists of $\boldsymbol{s} = (\alpha_1, \alpha_2, \dot{\alpha}_1, \dot{\alpha}_2)^\top$, where $\alpha_1 \in [-3/2\pi, 1/2\pi]$ [rad] and $\alpha_2 \in [-4/5\pi, 4/5\pi]$ [rad] are the angles of the shoulder and elbow joints, respectively, and $\dot{\alpha}_*$ [rad/s] is the angular velocity of each joint. The action space $\mathcal{A}$ is continuous and corresponds to the target angle a [rad] of the elbow joint. The reward function is defined as

$$R(\boldsymbol{s}, a, \boldsymbol{s}') = \frac{1}{2} I_1 \dot{\alpha}_1'^2 + m_1 g c_1 (1 + \sin \alpha_1'),$$

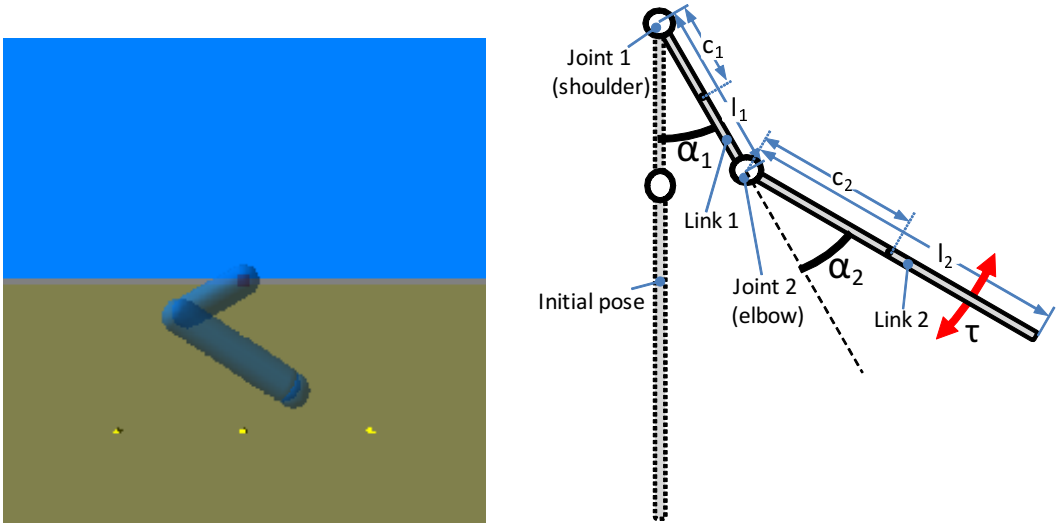
where this corresponds to the sum of kinetic and potential energies of the link 1 to evaluate the policy to swing up the entire body.

Similarly to the ball-balancing task, we design the feedback control system consisting of R^3 and the PD controllers. The R^3 controller outputs the target angle a of the elbow joint by current policy $\pi(a|s, \boldsymbol{\theta}_L)$ at every 0.2 [s]. 135 Gaussian kernels with width $\sigma_{\text{basis}} = 1$



(a) NIW ($\nu = 0$) (b) R^3 (ν is chosen by IWCV)

Figure 15: Typical examples of trajectories of the wrist angle α_{roll}, the elbow angle α_{pitch}, resulting ball movement x_*, and reward r_*, when following policies learned by NIW ($\nu = 0$) and R^3 (ν is chosen by IWCV) at the 10th RWR iteration in the realistic ball-balancing task.



(a) Simulator of Acrobot (b) Structure of Acrobot

Figure 16: Simulator used for the Acrobot control task based on *open-dynamics engine*. The initial pose is set to the bottom equilibrium state. The goal of the task is to swing the entire body up to the top equilibrium state by only controlling the elbow joint.

are used as basis functions with the kernel centers $\{c_b\}_{b=1}^{135}$ located over the state space as

$$(\alpha_1, \alpha_2, \dot{\alpha}_1, \dot{\alpha}_2) \in \{-3/2\pi, -\pi, -\pi/2, 0, \pi/2\} \times \{-\pi/2, 0, \pi/2\} \times \{-10, 0, 10\} \times \{-10, 0, 10\}.$$

Then, the PD controller converts the desired angle a to the torque τ applied to the elbow joint at every 0.005 [s]; the gain parameters are set to $k_{\mathrm{p}} = 100$ and $k_{\mathrm{d}} = 15$, respectively.

The initial state of the robot is set at the bottom equilibrium pose as $\boldsymbol{s} = (-\pi/2, 0, 0, 0)$. The initial policy parameter $\widehat{\boldsymbol{\theta}}_1^{\mathrm{AIW}}$ is set randomly as $\widehat{\boldsymbol{k}}_1^{\mathrm{AIW}} = (\beta_1, \beta_2, \ldots, \beta_{135})^\top$ and $\widehat{\sigma}_1^{\mathrm{AIW}} = 1$, where $\beta_1, \beta_2, \ldots, \beta_{135}$ are independently drawn from the uniform distribution on $[-0.5, 0.5]$. The dataset collected in each iteration consists of 10 episodes with 30 steps; the duration of an episode is 6 [s] and the sampling cycle by R^3 is 0.2 [s].

We consider five scenarios here: sample reuse with fixed $\nu = 0$ (NIW), $\nu = 1$ (PIW), the proposed R^3 (ν is chosen by IWCV from $\{0, 0.25, 0.5, 0.75, 1\}$ at every iteration), and energy-based Σ_2 (Spong, 1995) with high gain parameters ($k_{\mathrm{p}} = 100$ and $k_{\mathrm{d}} = 15$), and with low gain parameters ($k_{\mathrm{p}} = 50$ and $k_{\mathrm{d}} = 8$) as baseline methods. The discount factor is set to $\gamma = 0.99$. Figure 17 depicts the averaged expected return over 10 trials as a function of the number of RWR iterations. The expected return at each trial is computed from 20 test episodic data for this evaluation. The graph shows that R^3 nicely improves the performance over iterations. On the other hand, the speed of performance improvement when ν is fixed to 0 tends to be slow. When ν is fixed to 1, the performance is improved quickly in an initial stage, but its improvement is saturated after the 18th

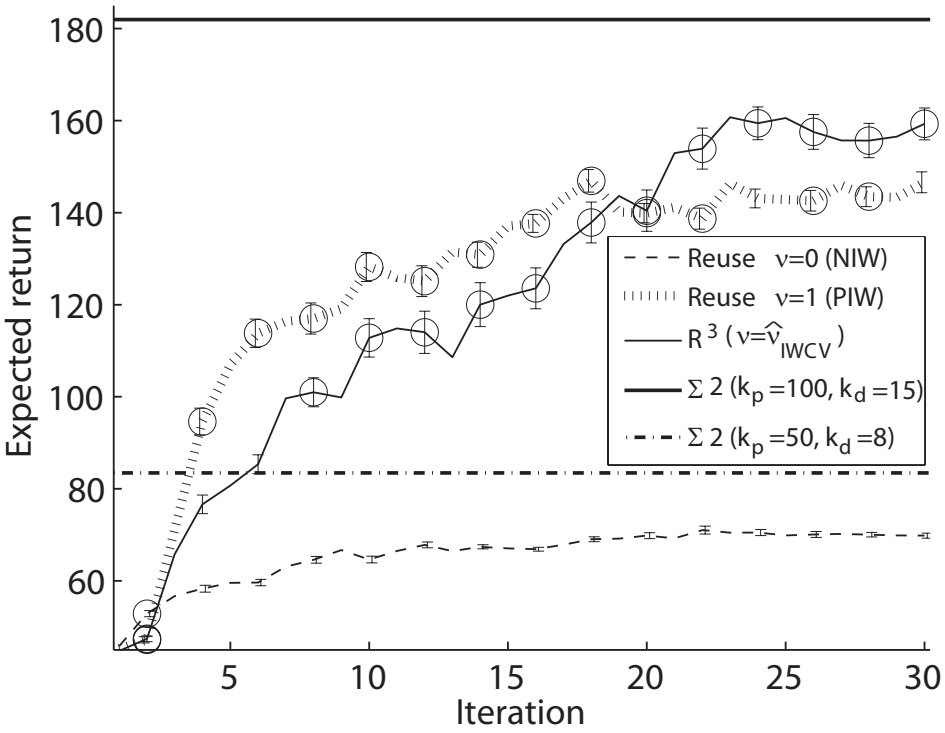


Figure 17: The performance of policies learned by RWR with $\nu = 0$ (NIW), $\nu = 1$ (PIW), and ν chosen by IWCV (R^3), and energy-based Σ_2 with different gain parameters as baseline methods in the Acrobot swing-up task. The performance is measured by the return averaged over 10 trials. The symbol '$\circ$' indicates the fact that the method is the best or comparable to the best one in terms of the expected return by the *t-test* at the significance level 5%, performed at every iteration. The error bars indicate 1/10 of standard deviation.

iteration due to the instability of policy update. The energy-based Σ_2 methods work well when gain parameters are adjusted well. However, good gain parameters depend on the property of the robot in a highly complicated way.

Figure 18 depicts an example of motion learned by R^3 (at the 20th iteration). The images show that the arm is repeatedly swung to the top.

5 Conclusions

In real-world reinforcement learning problems, reducing the number of training samples is highly important as the sampling costs are often much higher than the computational cost. In this paper, we proposed a new framework of direct policy search for efficient sample reuse. To overcome the instability problem caused by importance sampling, we proposed to combine reward-weighted regression with *adaptive* importance sampling techniques. The proposed method, called R^3, was shown to work well in experiments.

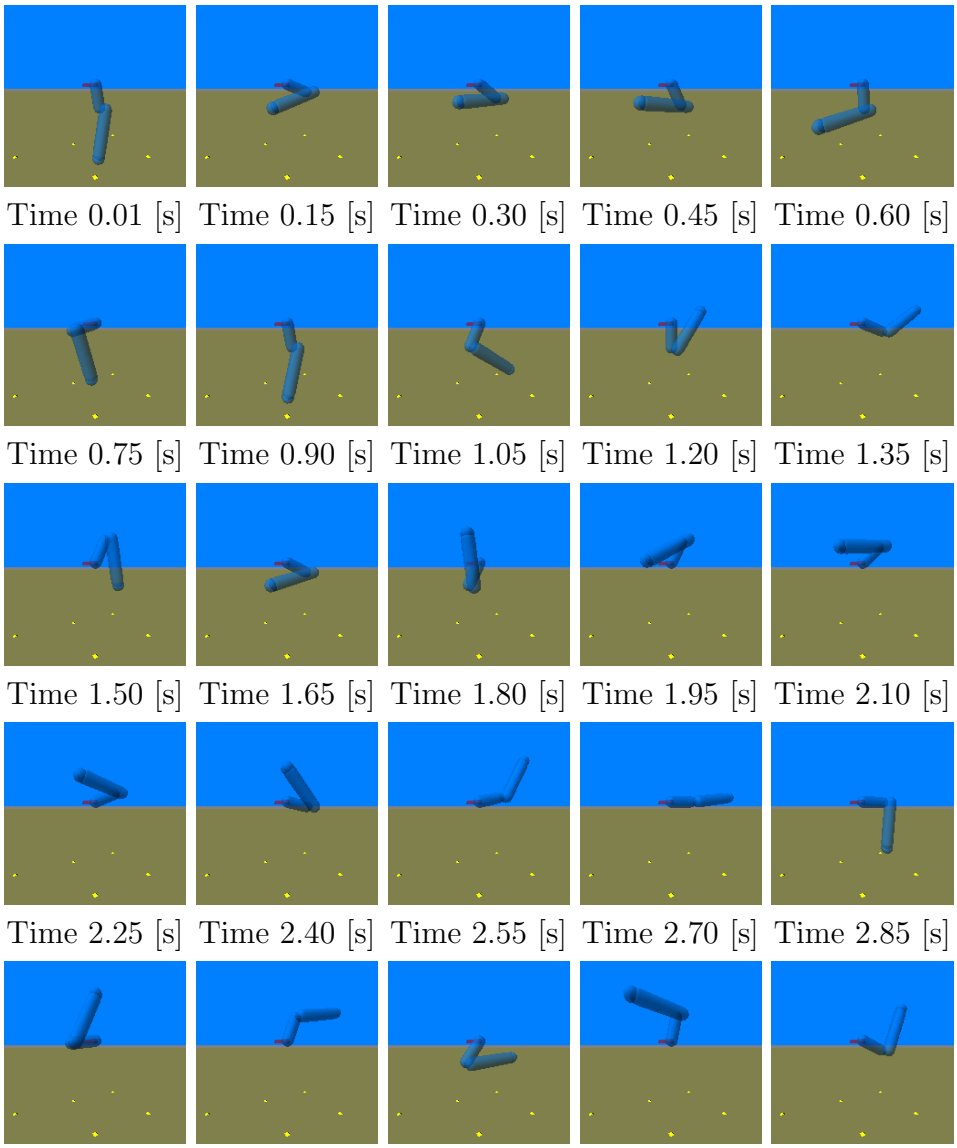


Figure 18: An example of Acrobot motion learned by R^3.

The proposed idea of using importance sampling techniques in direct policy search would be applicable to other policy search methods such as the *policy gradient* method (Williams, 1992; Sutton et al., 2000), the *natural policy gradient* method (Kakade, 2002; Peters et al., 2005), and *policy search by dynamic programming* (Bagnell et al., 2003). Extension along this line would be investigated in the future work.

We proposed to use an adaptive importance weighting technique in which the importance weight is powered by a flattening parameter. However, our proposed data-reuse framework is more general so that it can handle other types of stabilizing techniques. For example, *truncated importance weighting* (TIW) is also a popular variant in which impor-

tance weights larger than some threshold are rounded off to the threshold value (Uchibe & Doya, 2004; Wawrzynski, 2009). To obtain a good performance in data-reuse with TIW, the threshold parameter should be appropriately tuned. Our proposed framework can automatically select the parameter value based on estimated performances. Thus investigating the effect of the sample-reuse idea in other types of importance-weighting techniques would be important future works.

For the selection of the flattening parameter, we proposed to use importance-weighted cross validation (IWCV). However, as we pointed out in Section 2.6, the importance sampling tends to be unstable when only a small number of data samples following the same policy are available. Although we introduced an idea based on *per-decision importance-weight* to stabilize the importance weight in the parameter selection (see Section 3.1), IWCV still suffers from instability. Thus, developing a more stable model selection procedure is an essential research direction, e.g., by introducing an additional adaptive importance weighting into the IWCV-based model selection framework (cf. Sugiyama et al., 2004).

In the ball balancing experiment in Section 4.2, we concatenated two independent Gaussian policy models (see Eq.(4)) to handle the two-dimensional action space. When the synchronization of multiple actions is necessary to achieve the goal, one may employ the *multivariate Gaussian density* as a policy model:

$$\pi^d(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{K},\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{a}-\boldsymbol{K}^\top\boldsymbol{\phi}(\boldsymbol{s}))^\top\Sigma^{-1}(\boldsymbol{a}-\boldsymbol{K}^\top\boldsymbol{\phi}(\boldsymbol{s}))\Big),$$

where d is the dimension of the action space, $\boldsymbol{K}$ is the $B \times d$ parameter matrix, and Σ is the $d \times d$ covariance matrix. Similarly to the univariate Gaussian policy model, the policy parameter $\boldsymbol{\theta}^d_{L+1} = (\boldsymbol{K}^\top_{L+1}, \Sigma_{L+1})^\top$ may be updated analytically by maximizing the lower bound $\log J_L(\boldsymbol{\theta}^d)$. Thus, our proposed R^3 framework would be naturally extended for multivariate Gaussian policy models, which needs to be further investigated in the future work.

In this paper, we focused on the case where rollout samples are collected following the current policy. However, from the viewpoint of *active learning* (Sugiyama, 2006), the best sampling policy would be different from the current policy. Following this idea, an active learning method for better exploration has been developed in the framework of least-squares policy iteration and shown to perform well (Akiyama et al., 2010). Thus, developing active learning strategies in the framework of direct policy learning would also be a promising future direction to be pursued.

We focused on the discrete-time formulation and linearly interpolated the learned system for continuous-time robot control in Sections 4.2 and 4.3. Although this was shown to perform reasonably well, it is an important challenge to extend the current formulation so that continuous time systems can be directly handled. Further investigation along the line of Doya (2000) would be promising.

Acknowledgment

HH was supported by the FIRST program, and MS was supported by MEXT Grant-in-Aid for Young Scientists (A) 20680007, SCAT, and AOARD.

References

Akiyama, T., Hachiya, H., & Sugiyama, M. (2010). Efficient exploration through active learning for value function approximation in reinforcement learning. *Neural Networks*, *23*, 639–648.

Bagnell, J. A., Kakade, S., Ng, A. Y., & Schneider, J. (2003). Policy search by dynamic programming. *Neural Information Processing Systems 16* (pp. 831–838).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Dayan, P., & Hinton, G. E. (1997). Using expectation-maximization for reinforcement learning. *Neural Computation*, *9*, 271–278.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

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

Hachiya, H., Akiyama, T., Sugiyama, M., & Peters, J. (2009a). Adaptive importance sampling for value function approximation in off-policy reinforcement learning. *Neural Networks*, *22*, 1399–1410.

Hachiya, H., Peters, J., & Sugiyama, M. (2009b). Efficient sample reuse in em-based policy search. *Proceedings of the 20th European Conference on Machine Learning* (pp. 469–484).

Kakade, S. (2002). A natural policy gradient. *Neural Information Processing Systems 14* (pp. 1531–1538).

Kober, J., & Peters, J. (2008). Policy search for motor primitives in robotics. *Neural Information Processing Systems 21* (pp. 849–856).

Peshkin, L., & Shelton, C. R. (2002). Learning from scarce experience. *Proceedings of International Conference on Machine Learning* (pp. 498–505).

Peters, J., & Schaal, S. (2006). Policy gradient methods for robotics. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems* (pp. 2219–2225).

Peters, J., & Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. *Proceedings of the International Conference on Machine Learning* (pp. 745–750).

Peters, J., Vijayakumar, S., & Schaal, S. (2005). Natural actor-critic. *Proceedings of the 16th European Conference on Machine Learning* (pp. 280–291).

Precup, D., Sutton, R. S., & Singh, S. (2000). Eligibility traces for off-policy policy evaluation. *Proceedings of International Conference on Machine Learning* (pp. 759–766).

Schaal, S. (2009). *The SL simulation and real-time control software package* (Technical Report). Computer Science and Neuroscience, University of Southern California.

Shelton, C. R. (2001). Policy improvement for POMDPs using normalized importance sampling. *Proceedings of Uncertainty in Artificial Intelligence* (pp. 496–503).

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, *90*, 227–244.

Siciliano, B., & Khatib, O. (Eds.). (2008). *Springer handbook of robotics*. Springer.

Smith, R. (2005). Open dynamic engine. `http://www.ode.org`.

Spong, M. W. (1995). The swing up control problem for the acrobot. *IEEE Controll System Magazine*, 49–55.

Sugiyama, M. (2006). Active learning in approximately linear regression based on conditional expectation of generalization error. *Journal of Machine Learning Research*, *7*, 141–166.

Sugiyama, M., Kawanabe, M., & Müller, K.-R. (2004). Trading variance reduction with unbiasedness: The regularized subspace information criterion for robust model selection in kernel regression. *Neural Computation*, *16*, 1077–1104.

Sugiyama, M., Krauledat, M., & Müller, K.-R. (2007). Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, *8*, 985–1005.

Sugiyama, M., & Müller, K.-R. (2005). Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions*, *23*, 249–279.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. The MIT Press.

Sutton, R. S., McAllester, M., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Neural Information Processing Systems 12* (pp. 1057–1063).

Uchibe, E., & Doya, K. (2004). Competitive-cooperative-concurrent reinforcement learning with importance sampling. *Proceedings of International Conference on Simulation of Adaptive Behavior* (pp. 287–296).

Wawrzynski, P. (2009). Real-time reinforcement learning by sequential actor-critics and experience replay. *Neural Networks*, *22*, 1484–1497.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

A Derivation of Lower-Bound of Expected Return

Let us consider the log of the *normalized expected return* $J(\boldsymbol{\theta})/J(\boldsymbol{\theta}_L)$:

$$\begin{aligned}\log \frac{J(\boldsymbol{\theta})}{J(\boldsymbol{\theta}_L)} &= \log \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta})}{J(\boldsymbol{\theta}_L)} \mathrm{d}d \\ &= \log \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)}{J(\boldsymbol{\theta}_L)} \frac{P(d;\boldsymbol{\theta})}{P(d;\boldsymbol{\theta}_L)} \mathrm{d}d.\end{aligned}$$

By assuming $\mathcal{R}(d)$ to be non-negative and regarding $\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)/J(\boldsymbol{\theta}_L)$ as a probability density function, Jensen's inequality gives a lower-bound of the log normalized expected return:

$$\log \frac{J(\boldsymbol{\theta})}{J(\boldsymbol{\theta}_L)} \geq \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)}{J(\boldsymbol{\theta}_L)} \log \frac{P(d;\boldsymbol{\theta})}{P(d;\boldsymbol{\theta}_L)} \mathrm{d}d.$$

Then we obtain the lower bound of the log expected return as

$$\log J(\boldsymbol{\theta}) \geq \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)}{J(\boldsymbol{\theta}_L)} \log \frac{P(d;\boldsymbol{\theta})}{P(d;\boldsymbol{\theta}_L)} \mathrm{d}d + \log J(\boldsymbol{\theta}_L) \equiv \log J_L(\boldsymbol{\theta}).$$

B Derivation of Maximizer of Lower-Bound

The maximizer $\boldsymbol{\theta}_{L+1}$ of the lower bound $\log J_L(\boldsymbol{\theta})$ satisfies the following equation:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}} \log J_L(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{L+1}} &= \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)}{J(\boldsymbol{\theta}_L)} \frac{\partial}{\partial \boldsymbol{\theta}} \log P(d;\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{L+1}} \mathrm{d}d \\ &= \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)}{J(\boldsymbol{\theta}_L)} \sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{L+1}} \mathrm{d}d = \mathbf{0},\end{aligned}$$

where we used Eq.(1). A useful property of the Gaussian policy model is that the log-derivative of the policy model with respect to the parameters can be analytically computed

as

$$\frac{\partial}{\partial \boldsymbol{k}} \log \pi(a|\boldsymbol{s};\boldsymbol{\theta}) = \frac{a - \boldsymbol{k}^\top \boldsymbol{\phi}(\boldsymbol{s})}{\sigma^2} \boldsymbol{\phi}(\boldsymbol{s}),$$
$$\frac{\partial}{\partial \sigma} \log \pi(a|\boldsymbol{s};\boldsymbol{\theta}) = \frac{(a - \boldsymbol{k}^\top \boldsymbol{\phi}(\boldsymbol{s}))^2 - \sigma^2}{\sigma^3}.$$

Then the maximizer $\boldsymbol{\theta}_{L+1} = (\boldsymbol{k}_{L+1}^\top, \sigma_{L+1})^\top$ can be analytically obtained as

$$\begin{cases} \boldsymbol{k}_{L+1} = \left(\int \mathcal{R}(d) P(d;\boldsymbol{\theta}_L) \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{s}_n) \boldsymbol{\phi}(\boldsymbol{s}_n)^\top \mathrm{d}d \right)^{-1} \\ \qquad\qquad \times \left(\int \mathcal{R}(d) P(d;\boldsymbol{\theta}_L) \frac{1}{N} \sum_{n=1}^{N} a_n \boldsymbol{\phi}(\boldsymbol{s}_n) \mathrm{d}d \right), \\ \sigma_{L+1}^2 = \left(\int \mathcal{R}(d) P(d;\boldsymbol{\theta}_L) \mathrm{d}d \right)^{-1} \left(\int \mathcal{R}(d) P(d;\boldsymbol{\theta}_L) \frac{1}{N} \sum_{n=1}^{N} (a_n - \boldsymbol{k}_{L+1}{}^\top \boldsymbol{\phi}(\boldsymbol{s}_n))^2 \mathrm{d}d \right). \end{cases}$$

C Derivation of Per-Decision Importance Weights

The expected return $J(\boldsymbol{\theta}_L)$ can be expressed with stepwise importance weights $w_{L,l}^n(d)$ as follows:

$$\begin{aligned}
J(\boldsymbol{\theta}_L) &= \int \Big(\sum_{n=1}^{N} \gamma^{n-1} r_n \Big) P(d;\boldsymbol{\theta}_L) \mathrm{d}d \\
&= \int \Big(\sum_{n=1}^{N} \gamma^{n-1} r_n \Big) w_{L,l}(d) P(d;\boldsymbol{\theta}_l) \mathrm{d}d \\
&= \int r_1 \frac{P(d;\boldsymbol{\theta}_L)}{P(d;\boldsymbol{\theta}_l)} P(d;\boldsymbol{\theta}_l) \mathrm{d}d + \int \gamma r_2 \frac{P(d;\boldsymbol{\theta}_L)}{P(d;\boldsymbol{\theta}_l)} P(d;\boldsymbol{\theta}_l) \mathrm{d}d + \cdots \\
&= \iiint r_1 \frac{P(\boldsymbol{s}_1, a_1, \boldsymbol{s}_2;\boldsymbol{\theta}_L)}{P(\boldsymbol{s}_1, a_1, \boldsymbol{s}_2;\boldsymbol{\theta}_l)} P(\boldsymbol{s}_1, a_1, \boldsymbol{s}_2;\boldsymbol{\theta}_l) \mathrm{d}\boldsymbol{s}_1 \mathrm{d}a_1 \mathrm{d}\boldsymbol{s}_2 \\
&+ \iiint\!\!\iint \gamma r_2 \frac{P(\boldsymbol{s}_1, a_1, \boldsymbol{s}_2, a_2, \boldsymbol{s}_3;\boldsymbol{\theta}_L)}{P(\boldsymbol{s}_1, a_1, \boldsymbol{s}_2, a_2, \boldsymbol{s}_3;\boldsymbol{\theta}_l)} P(\boldsymbol{s}_1, a_1, \boldsymbol{s}_2, a_2, \boldsymbol{s}_3;\boldsymbol{\theta}_l) \mathrm{d}\boldsymbol{s}_1 \mathrm{d}a_1 \mathrm{d}\boldsymbol{s}_2 \mathrm{d}a_2 \mathrm{d}\boldsymbol{s}_3 + \cdots \\
&= \iiint r_1 \frac{\pi(a_1|\boldsymbol{s}_1;\boldsymbol{\theta}_L)}{\pi(a_1|\boldsymbol{s}_1;\boldsymbol{\theta}_l)} P(\boldsymbol{s}_1, a_1, \boldsymbol{s}_2;\boldsymbol{\theta}_l) \mathrm{d}\boldsymbol{s}_1 \mathrm{d}a_1 \mathrm{d}\boldsymbol{s}_2 \\
&+ \iiint\!\!\iint \gamma r_2 \frac{\pi(a_1|\boldsymbol{s}_1;\boldsymbol{\theta}_L)\pi(a_2|\boldsymbol{s}_2;\boldsymbol{\theta}_L)}{\pi(a_1|\boldsymbol{s}_1;\boldsymbol{\theta}_l)\pi(a_2|\boldsymbol{s}_2;\boldsymbol{\theta}_l)} P(\boldsymbol{s}_1, a_1, \boldsymbol{s}_2, a_2, \boldsymbol{s}_3;\boldsymbol{\theta}_l) \mathrm{d}\boldsymbol{s}_1 \mathrm{d}a_1 \mathrm{d}\boldsymbol{s}_2 \mathrm{d}a_2 \mathrm{d}\boldsymbol{s}_3 + \cdots \\
&= \int \Big(\sum_{n=1}^{N} \gamma^{n-1} r_n w_{L,l}^n(d) \Big) P(d;\boldsymbol{\theta}_l) \mathrm{d}d.
\end{aligned}$$