Efficient Sample Reuse in EM-based Policy Search

Hirotaka Hachiya†, Jan Peters‡, and Masashi Sugiyama†

†Tokyo Institute of Technology, Tokyo, 152-8552, Japan
{hachiya@sg, sugi@}cs.titech.ac.jp
‡Max-Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany
jan.peters@tuebingen.mpg.de

Abstract. Direct policy search is a promising reinforcement learning framework in particular for controlling in continuous, high-dimensional systems such as anthropomorphic robots. Policy search often requires a large number of samples for obtaining a stable policy update estimator due to its high flexibility. However, this is prohibitive when the sampling cost is expensive. In this paper, we extend an EM-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 a robot learning experiment.

1 Introduction

Model-free reinforcement learning is an important tool for solving real-world Markov decision problems online. However, due to its high flexibility, 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, we need to let the robot hit a ball hundreds of times for obtaining a reliable policy-improvement direction; then this policy update steps need to be repeated many times for finally obtaining a good policy. As a result, robot engineers need to spend a lot of time and effort 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* [16] for which the bias is canceled out asymptotically. However, a naive use of importance sampling significantly increases the variance of the estimator and, therefore, it becomes highly unstable. To mitigate this problem, the *per-decision* importance-weighting technique has been introduced for variance reduction [10]. This technique cleverly 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 for real-world reinforcement learning tasks yet. For more stable and efficient variance reduction, an *adaptive* importance sampling scheme has recently been proposed [4]. This formulates the off-policy value function approximation problem as *covariate shift adaptation*, which is a statistical learning paradigm under non-stationarity: a 'flattening' parameter is introduced to the importance weight for trading the variance reduction with a slight bias increase—the bias-variance trade-off is optimally controlled based on *importance-weighted cross-validation* [15].

Due to the above efforts, reinforcement learning methods based on value function approximation can 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 [18, 17], the *natural policy gradient* method [5, 9], and the *policy search by expectation-maximization* [2, 8]. 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 applied in the direct policy search, they were shown to be unstable [13, 7]. For stabilization purposes, heuristic techniques are often used in practice, e.g., samples with smaller importance weights are not used for learning [6]. However, to the best of our knowledge, systematic treatment of instability issues in the 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 this problem. In particular, we combine the policy search by expectation-maximization [2, 8] with the covariate shift adaptation paradigm, and 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 robot-control experiments.

2 Policy Search Framework

We consider the standard reinforcement learning framework in which an agent interacts with a Markov decision process. In this section, we review how Markov decision problems can be solved using the policy search by expectation-maximization [2]; for Gaussian models, this results in the reward-weighted regression (RWR) algorithm [8].

2.1 Markov Decision Problems

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, $P_{\mathrm{T}}(\boldsymbol{s}'|\boldsymbol{s}, a)$

($\in (0,1]$) is the transition probability-density from state $\boldsymbol{s}$ to next state $\boldsymbol{s}'$ when action a is taken, $P_{\mathrm{I}}(\boldsymbol{s})$ ($\in (0,1]$) is the probability of initial states, $R(\boldsymbol{s},a,\boldsymbol{s}')$ (≥ 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 $\pi(a|\boldsymbol{s};\boldsymbol{\theta})$ ($\in (0,1]$) 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 a *trajectory* of the agent by $d \equiv (\boldsymbol{s}_1, a_1, \boldsymbol{s}_2, a_2, \ldots, \boldsymbol{s}_N, a_N, s_{N+1})$, where N is the length of the trajectory. Let $\mathcal{R}(d)$ be the *return* (i.e., the sum of discounted rewards) along trajectory d:

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

Let $P(d;\boldsymbol{\theta})$ be the probability of occurring trajectory d under $P_{\mathrm{I}}(\boldsymbol{s}_1)$, $P_{\mathrm{T}}(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n, a_n)$, and $\pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta})$:

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

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

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

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

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

2.2 EM-based Policy Search

Naively maximizing $J(\boldsymbol{\theta})$ is hard since $J(\boldsymbol{\theta})$ usually contains high non-linearity. The basic idea of the policy search by expectation-maximization is to iteratively update the policy parameter $\boldsymbol{\theta}$ by maximizing a lower bound of the target cost function [3]. More precisely, let $\boldsymbol{\theta}_L$ be the current policy parameter, where the subscript L indicates the iteration number. Then the use of *Jensen's inequality* allows us to obtain a lower bound of the *normalized expected return* $\log(J(\boldsymbol{\theta})/J(\boldsymbol{\theta}_L))$ as

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

Note that $\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)/J(\boldsymbol{\theta}_L)$ is treated as a probability density function in applying Jensen's inequality; this is why $\mathcal{R}(d)$ is assumed to be non-negative and the expected return is normalized.

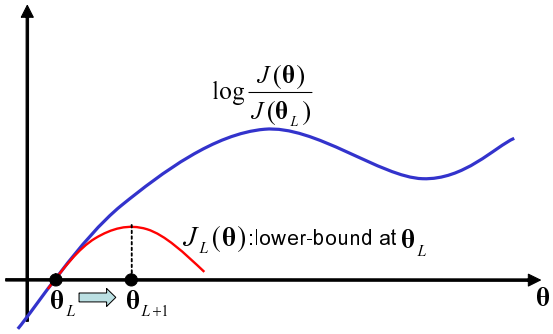


Fig. 1. Relation between the normalized expected return $\log \frac{J(\boldsymbol{\theta})}{J(\boldsymbol{\theta}_L)}$ and its lower bound $J_L(\boldsymbol{\theta})$. At $\boldsymbol{\theta}_L$, the lower bound touches $\log \frac{J(\boldsymbol{\theta})}{J(\boldsymbol{\theta}_L)}$ with zero value.

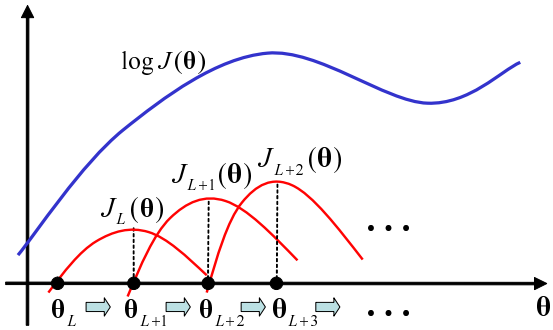


Fig. 2. The policy parameter $\boldsymbol{\theta}$ is updated iteratively by maximizing lower bounds.

The expectation-maximization (EM) approach iteratively updates the parameter $\boldsymbol{\theta}$ 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 $J_L(\boldsymbol{\theta}_L) = 0$, 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 (un)normalized expected return is guaranteed (Fig.1):

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

This update is iterated until convergence (see Fig.2).

2.3 Reward-weighted Regression

Let us employ the Gaussian policy model defined as

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

where $\boldsymbol{k} = (k_1, k_2, \ldots, k_b)^\top$ and σ are policy parameters, b is the number of basis functions, and $\boldsymbol{\phi}(\boldsymbol{s}) = (\boldsymbol{\phi}_1(\boldsymbol{s}), \boldsymbol{\phi}_2(\boldsymbol{s}), \ldots, \boldsymbol{\phi}_b(\boldsymbol{s}))^\top$ are fixed basis functions. 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. The maximizer $\boldsymbol{\theta}_{L+1}$ of the lower bound $J_L(\boldsymbol{\theta})$ satisfies the following equation:

$$\begin{aligned}\left.\frac{\partial}{\partial\boldsymbol{\theta}}J_L(\boldsymbol{\theta})\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{L+1}} &= \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)}{J(\boldsymbol{\theta}_L)}\left.\frac{\partial}{\partial\boldsymbol{\theta}}\log P(d;\boldsymbol{\theta})\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{L+1}}\mathrm{d}d \\ &= \int \frac{\mathcal{R}(d)P(d;\boldsymbol{\theta}_L)}{J(\boldsymbol{\theta}_L)}\sum_{n=1}^{N}\left.\frac{\partial}{\partial\boldsymbol{\theta}}\log\pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta})\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{L+1}}\mathrm{d}d = \boldsymbol{0}. \quad (5)\end{aligned}$$

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

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

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

$$\begin{aligned}\boldsymbol{k}_{L+1} &= \Big(\int \mathcal{R}(d)P(d;\boldsymbol{\theta}_L)\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{s}_n)\boldsymbol{\phi}(\boldsymbol{s}_n)^\top\Big)^{-1} \\ &\quad\times\Big(\int \mathcal{R}(d)P(d;\boldsymbol{\theta}_L)\sum_{n=1}^{N}a_n\boldsymbol{\phi}(\boldsymbol{s}_n)\Big), \\ \sigma_{L+1}^2 &= \Big(N\int \mathcal{R}(d)P(d;\boldsymbol{\theta}_L)\Big)^{-1} \\ &\quad\times\Big(\int \mathcal{R}(d)P(d;\boldsymbol{\theta}_L)\sum_{n=1}^{N}(a_n-\boldsymbol{k}_{L+1}{}^\top\boldsymbol{\phi}(\boldsymbol{s}_n))^2\Big).\end{aligned}$$

The EM-based policy search for Gaussian models is called *reward-weighted regression* [8].

2.4 Learning from Episodic Data Samples

Suppose a dataset consisting of M episodes with N steps is available for each RWR iteration, where each episodic sample is 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 $\pi(a_n|\boldsymbol{s}_n;\boldsymbol{\theta}_L)$ at the L-th iteration. 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 L-th iteration is expressed as $\mathcal{D}^L \equiv \{d_m^L\}_{m=1}^M$, where each episodic sample d_m^L is given by

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

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

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

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 could 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.(6) gives an *efficient* estimator[1] $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{NIW}} \equiv (\widehat{\boldsymbol{k}}_{L+1}^{\mathrm{NIW}\top}, \widehat{\sigma}_{L+1}^{\mathrm{NIW}})^\top$, where

$$\begin{aligned} \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{NIW}} &= \Big(\sum_{l=1}^L \sum_{m=1}^M \mathcal{R}(d_m^l) \sum_{n=1}^N \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l) \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)^\top \Big)^{-1} \\ &\qquad \times \Big(\sum_{l=1}^L \sum_{m=1}^M \mathcal{R}(d_m^l) \sum_{n=1}^N a_{m,n}^l \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l) \Big), \\ (\widehat{\sigma}_{L+1}^{\mathrm{NIW}})^2 &= \Big(N \sum_{l=1}^L \sum_{m=1}^M \mathcal{R}(d_m^l) \Big)^{-1} \\ &\qquad \times \Big(\sum_{l=1}^L \sum_{m=1}^M \mathcal{R}(d_m^l) \sum_{n=1}^N \Big(a_{m,n}^l - \widehat{\boldsymbol{k}}_{L+1}^{\mathrm{NIW}\top} \boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l) \Big)^2 \Big). \end{aligned}$$

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 for different policies and, therefore, the naive use of Eq.(6) will result in a biased estimator.

[1] As the number of episodes goes to infinity, the estimated parameter converges to the optimal value and its variance achieves the Cramér-Rao lower bound [11].

Importance sampling 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 some function $g(d)$ over another probability density function $P(d;\boldsymbol{\theta}_L)$ can be consistently estimated by the *importance-weighted average*:

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

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

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

$$\begin{aligned}\widehat{\boldsymbol{k}}_{L+1}^{\mathrm{IW}} &= \Big(\sum_{l=1}^{L}\sum_{m=1}^{M}\mathcal{R}(d_m^l)w_{L,l}(d_m^l)\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)^\top\Big)^{-1}\\ &\quad\times\Big(\sum_{l=1}^{L}\sum_{m=1}^{M}\mathcal{R}(d_m^l)w_{L,l}(d_m^l)\sum_{n=1}^{N}a_{m,n}^l\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\Big),\\ (\widehat{\sigma}_{L+1}^{\mathrm{IW}})^2 &= \Big(N\sum_{l=1}^{L}\sum_{m=1}^{M}\mathcal{R}(d_m^l)w_{L,l}(d_m^l)\Big)^{-1}\\ &\quad\times\Big(\sum_{l=1}^{L}\sum_{m=1}^{M}\mathcal{R}(d_m^l)w_{L,l}(d_m^l)\sum_{n=1}^{N}\Big(a_{m,n}^l-\widehat{\boldsymbol{k}}_{L+1}^{\mathrm{IW}\top}\boldsymbol{\phi}(\boldsymbol{s}_{m,n}^l)\Big)^2\Big).\end{aligned}$$

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 'IW' abbreviates '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 $\{P_{\mathrm{T}}(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n,a_n)\}_{n=1}^N$. 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 is guaranteed to be consistent, it is not *efficient* even asymptotically. Therefore, the importance-weighted estimator tends to be unstable when the number of samples is rather small. The purpose of this paper is to propose a new framework for systematically addressing instability problems in the direct policy search.

3 Adaptive Importance Weight for Stable Policy Search

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

3.1 Bias-Variance Trade-off and Adaptive Importance Weight

When data samples $\{\mathcal{D}_l\}_{l=1}^{L}$ follow different distributions, the NIW estimator is biased even asymptotically. On the other hand, the IW estimator is asymptotically unbiased [14]. Thus, IW would generally have a smaller bias than NIW. However, IW has a larger variance than NIW, so there is the trade-off between the bias and variance.

In order to appropriately control the bias-variance trade-off, we introduce an *adaptive importance weighting* technique [14]. For a *flattening parameter* ν ($\in[0,1]$), the importance weight $w_{L,l}(d)$ is 'flattened' as $w_{L,l}^{\nu}(d)$ ($w_{L,l}(d)$ to the power of ν). 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

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

'AIW' stands for 'Adaptive Importance Weight'. When $\nu = 0$, AIW is reduced to NIW. On the other hand, when $\nu = 1$, AIW is reduced to IW. In practice, an intermediate ν often produces a better estimator since the bias-variance trade-off can be better controlled.

In the above AIW method, the flattening parameter value ν can be different for each $\mathcal{D}^l$. However, for simplicity, we employ a single common value ν.

3.2 Flattening Parameter Selection

The performance of AIW depends on the choice of the flattening parameter ν, which allows us to control the bias-variance trade-off. Here, we show how the value of the flattening parameter can be optimally chosen using samples.

The goal of the 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 L-th iteration is given by

$$
\nu_L^* \equiv \arg\max_{\nu} J(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu)). \tag{8}
$$

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 called *importance-weighted cross-validation* (IWCV) [15].

The basic idea of IWCV is to split the training dataset $\mathcal{D}^{1:L} = \{\mathcal{D}^l\}_{l=1}^L$ into an 'estimation part' and a 'validation part'. Then the 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 validation part. Below, we explain in more detail how we apply IWCV to the selection of the flattening parameter ν in the current context. For simplicity, we assume that M is divisible by K, i.e., M/K is an integer.

Let us randomly divide the training dataset $\mathcal{D}^{1:L} = \{\mathcal{D}^l\}_{l=1}^L$ into K (we use $K = 5$ in the experiments) disjoint subsets $\{\mathcal{D}_k^{1:L}\}_{k=1}^K$ of the same size, 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. We estimate the expected return of $\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}(\nu)$ using $\mathcal{D}_k^{1:L}$ as

$$\widehat{J}_{\mathrm{IWCV}}^k(\widehat{\boldsymbol{\theta}}_{L+1,k}^{\mathrm{AIW}}(\nu)) \equiv \frac{1}{\eta} \sum_{d^l \in \mathcal{D}_k^{1:L}} \mathcal{R}(d^l) w_{L,l}(d^l),$$

where d^l denotes an episodic sample from $\mathcal{D}^l$ and η is a normalization constant. An ordinary choice of η would be $\eta = LM/K$, but we use a 'stable' variant $\eta \equiv \sum_{d^l \in \mathcal{D}_k^{1:L}} w_{L,l}(d^l)$ [10].

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 is shown to be unbiased [15].

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:

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

Note that the above IWCV scheme can be used also for choosing the type and number of basis functions $\{\phi_i(\boldsymbol{s})\}_{i=1}^b$ in the Gaussian policy model (4).

3.3 Numerical Examples

Here, we illustrate how the proposed method behaves using a one-dimensional ball-balancing simulation illustrated in Fig.3. The goal is to control the angle

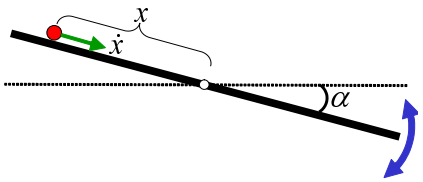


Fig. 3. Illustration of one-dimensional ball-balancing simulation. The goal is to control the angle α of the plate so that the ball is kept in the middle of plate.

of the plate so that the ball is kept in the middle of the plate. The action space $\mathcal{A}$ consists of the angle α ($\in (-\pi/4, \pi/4)$) [rad] of the plate 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 distance x [m] between the centers of mass of the ball and the plate, and the velocity $\dot{x}$ [m/s] of the ball. The distance x and velocity $\dot{x}$ can be modeled using the following equations:

$$x_{t+1} = x_t + \dot{x}_{t+1}\Delta t,$$
$$\dot{x}_{t+1} = \dot{x}_t + \Delta t\Big(-\frac{f}{m}\dot{x}_t - 9.8\sin(a_t)\Big),$$

where $f = 0.5$ 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 plate angle will be adjusted with a single time-step (this simulation is for illustration purposes; more realistic experiments will be shown in the next section). The reward function $R(s, a, s')$ is a quadratic function defined as

$$R'(\boldsymbol{s}, a, \boldsymbol{s}') = -\boldsymbol{s}'^\top \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix} \boldsymbol{s}' - 0.1a^2.$$

This reward function indicates that the agent will receive the maximum reward (i.e., zero) when the ball stops at the center of the plate ($\boldsymbol{s} = \boldsymbol{0}$). However, the above reward function takes negative values and therefore violates the non-negativity assumption imposed in the derivation of the EM algorithm (see Section 2). To cope with this problem, when the policy search is carried out, we convert the return $\mathcal{R}(d)$ as

$$\mathcal{R}'(d) = -\frac{1}{0.00001 + \mathcal{R}(d)}. \tag{9}$$

We use the Gaussian policy model with $\boldsymbol{\phi}(\boldsymbol{s}) = \boldsymbol{s}$. The discount factor was set to $\gamma = 0.95$.

First, let us illustrate how the flattening parameter ν influences the policy-parameter update. For this purpose, we compute $\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu)$ only from $\mathcal{D}^{L-1}$. The target policy parameter $\widehat{\boldsymbol{\theta}}_L = (\widehat{\boldsymbol{k}}_L^\top, \widehat{\sigma}_L)^\top$ is fixed to $\widehat{\boldsymbol{k}}_L = (0.5, 0.5)^\top$ and $\widehat{\sigma}_L =$

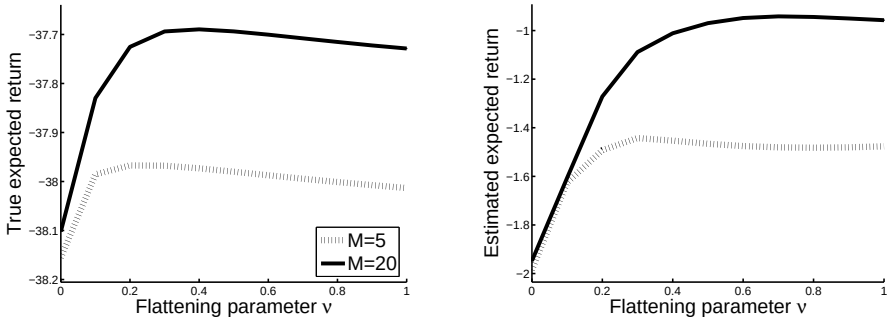
0.2. The sample-generation policy parameter $\widehat{\boldsymbol{\theta}}_{L-1}$ is chosen randomly as $\widehat{\boldsymbol{k}}_{L-1} = \widehat{\boldsymbol{k}}_L - (\beta_1, \beta_2)^\top$ and $\widehat{\sigma}_{L-1} = \widehat{\sigma}_L$, where β_1 and β_2 independently follow the uniform distribution on $[0, 0.2]$. Fig.4(a) depicts the true expected return $J(\widehat{\boldsymbol{\theta}}^{\mathrm{AIW}}_{L+1}(\nu))$ averaged over 20 trials as a function of the flattening parameter ν for $M = 5$ and 20.

The graph overall shows that when the number M of episodes is larger, the average expected return is also larger. However, the performance of NIW ($\nu = 0$) is not improved significantly when M is increased. The reason for this phenomenon is that increasing the number of samples does not contribute to reducing the estimation bias in NIW. The amount of variance reduction in NIW is limited as the variance is relatively small even when M is not large. On the other hand, the performance of IW ($\nu = 1$) is significantly improved as M is increased. This is because IW tends to have a large variance when M is small and increasing M highly contributes to reducing the variance. The graph also shows that neither NIW nor IW is the best in this simulation—intermediate values of ν (say, $0.2 \leq \nu \leq 0.4$) perform better than NIW and IW. Thus, given that ν is chosen optimally, AIW can outperform IW and NIW. Note that, although the amount of performance improvement by AIW over IW seems subtle in this one-step experiment, accumulation of this small gain through iterations actually causes big performance improvement (Fig.6).

Next, we illustrate how IWCV behaves. Fig.4(b) depicts the expected return $\widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}^{\mathrm{AIW}}_{L+1}(\nu))$ estimated by 5-fold IWCV, averaged over 20 trials as a function of the flattening parameter ν. The graphs show that IWCV roughly captures the trend of the true expected return for both cases ($M = 5$ and 20). Note that the orders of magnitude of the values in Fig.4(a) and in Fig.4(b) are different due to the importance weights. However, this does not cause a problem in model selection as long as relative profiles of the curves are similar.

Fig.5 depicts the expected return obtained by NIW ($\nu = 0$), IW ($\nu = 1$), and AIW+IWCV ($\nu \in \{0, 0.1, 0.2, \ldots, 1\}$ is selected in each trial using 5-fold IWCV) averaged over 20 trials as a function of the number M of episodes. This result indicates that the performance of NIW ($\nu = 0$) does not improve significantly when M is increased, implying that the estimation bias in NIW is crucial. On the other hand, the performance of IW ($\nu = 1$) is highly improved when M is increased. Thus, consistency of IW would be useful in this simulation. The proposed method, AIW+IWCV, tends to outperform both NIW and IW.

Finally, we illustrate how R^3 behaves in three different scenarios; through RWR iterations, ν is fixed to 0 (NIW), ν is fixed to 1 (IW), and ν is chosen by IWCV (R^3). Starting from a randomly-chosen initial policy-parameter $\widehat{\boldsymbol{\theta}}_1$, we let the agent collect samples $\mathcal{D}^L$ following the current policy $\pi(a|\boldsymbol{s}; \widehat{\boldsymbol{\theta}}_L)$, and then the policy parameter is updated using all samples $\{\mathcal{D}^l\}_{l=1}^L$. This is repeated over iterations. Fig.6 depicts the return averaged over 20 trials as a function of the number of RWR iterations; the number of episodes is $M = 5$ or 20, while the number of steps is fixed to $N = 20$.



(a) True expected return (b) 5-fold IWCV estimate

Fig. 4. True expected return $J(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu))$ and 5-fold IWCV estimate $\widehat{J}_{\mathrm{IWCV}}(\widehat{\boldsymbol{\theta}}_{L+1}^{\mathrm{AIW}}(\nu))$ averaged over 20 trials as a function of the flattening parameter ν in one-dimensional ball-balancing task.

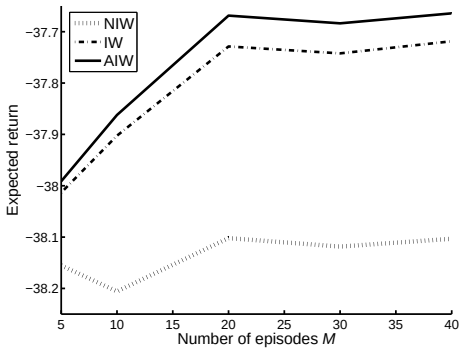
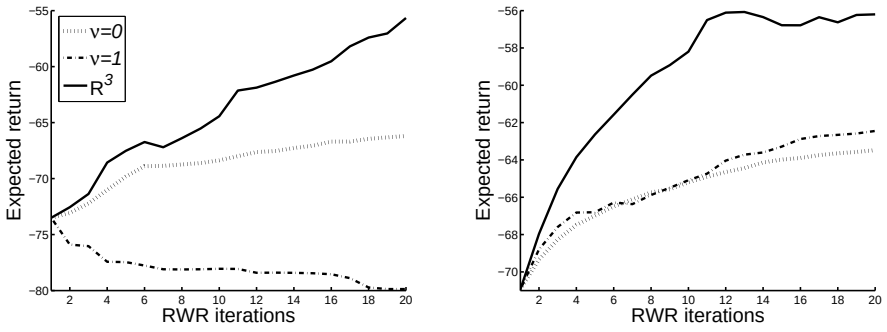


Fig. 5. Expected return obtained by NIW ($\nu = 0$), IW ($\nu = 1$), and AIW (ν is chosen by IWCV) averaged over 20 trials as a function of the number of episodes in one-dimensional ball-balancing task.



(a) $M = 5$ (b) $M = 20$

Fig. 6. The performance of policies learned in three scenarios: $\nu = 0$ (NIW), $\nu = 1$ (IW), and ν is chosen by IWCV (R^3) in one-dimensional ball-balancing task. The performance is measured by the expected return averaged over 20 trials.

The graphs show that R^3 tends to outperform the non-adaptive schemes. When ν is fixed to 0 (NIW), the performance of policy is stable for both cases. However, performance improvement is much slower than that for R^3. On the other hand, when ν is fixed to 1 (IW), the behavior depends on the number of samples; it works as well as NIW for $M = 20$, but the performance is poor and is not improved over iterations for $M = 5$ due to the high estimation variance.

Our original motivation to introduce R^3 was to reduce the number of samples for saving the sampling cost. When the sampling cost is high, it is preferable to keep M small, e.g., $M = 5$. In this case, the IW methods ($\nu = 1$) performs very poorly due to the huge estimation variance. On the other hand, the proposed R^3 method is stable even in the small sample case and its performance tends to be better than the naive RWR method ($\nu = 0$).

4 Applications

In this section, we evaluate the performance of our proposed method R^3 in a more complicated environment, i.e., a physically realistic ball-balancing task using a Barrett WAM^{TM} robot-arm simulator.

The 7 degree-of-freedom Barrett WAM^{TM} arm is mounted on the ceiling upside down and has a circular tray attached at the end-effector (see Fig.7). The goal is to control the robot so that the ball is always brought to the middle of the tray, similarly to the toy ball-balancing task in the previous section. However, unlike before, the angle of the tray cannot be directly controlled here as this is not feasible in a realistic scenario. Thus, achieving the goal is much harder.

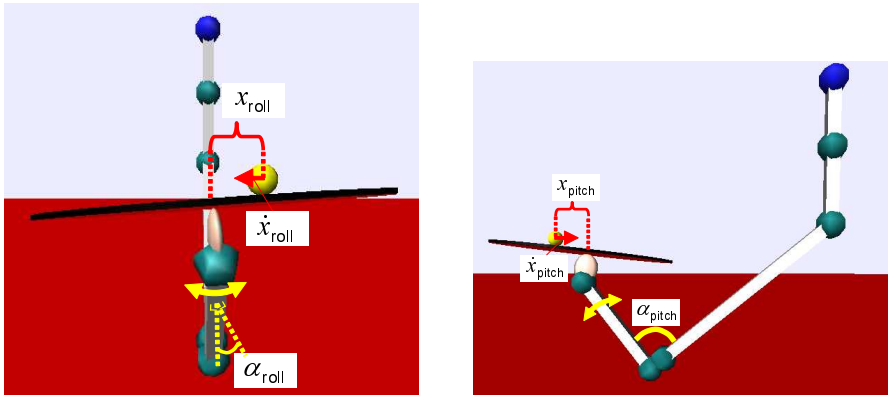
The initial pose of the robot is set so that the tray is slightly tilted and the ball is rolling on the tray (see Fig.7). To simplify the problem, we control only two degrees of freedom: the wrist angle α_{roll} and the elbow angle α_{pitch}; all the remaining joints are fixed. Control of the wrist and elbow angles roughly corresponds to changing the *roll* and *pitch* of the tray, but not directly.

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

$$R_*(\boldsymbol{s}, a, \boldsymbol{s}') = -(x'_*)^2 - (\dot{x}'_*)^2 - a_*^2.$$

Since this reward function is negative, the return is converted to a non-negative one by Eq.(9) when the policy search is carried out.

We explain how the control system is designed in more detail. As described in Fig.8, the control system has two feedback loops: joint-angle control using a *proportional and derivative (PD) controller* and motion control using an R^3 controller. The PD controller outputs motor torque to achieve the desired joint



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

Fig. 7. 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.

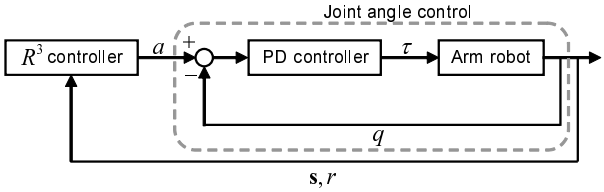


Fig. 8. The architecture of the Barrett WAM$^{\text{TM}}$ robot-arm control system for ball balancing. The control system has two feedback loops, i.e., joint angle control by a PD controller and motion control by an R^3 controller.

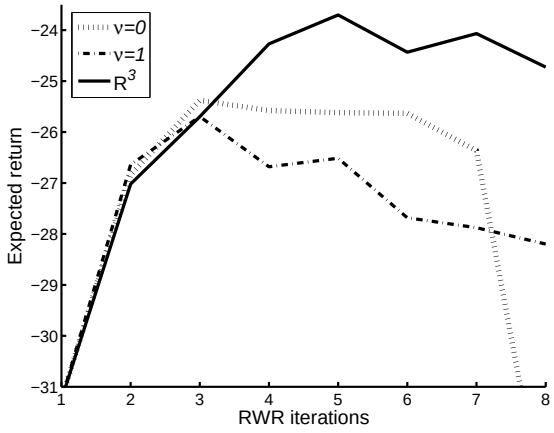


Fig. 9. The performance of learned policies when $\nu = 0$ (NIW), $\nu = 1$ (IW), and ν is chosen by IWCV (R^3) in ball balancing using a simulated Barrett WAM$^{\text{TM}}$ arm system. The performance is measured by the expected return averaged over 10 trials.

angle a_* as

$$\tau_* = k_{\mathrm{p}}(\alpha_* - a_*) + k_{\mathrm{d}}\dot{\alpha}_*,$$

where $k_{\mathrm{p}} = 20$ is the proportional gain and $k_{\mathrm{d}} = 1$ is the derivative gain. On the other hand, the R^3 controller outputs the target joint angle obtained by current policy $\pi(a|\boldsymbol{s};\boldsymbol{\theta}_L)$; the policy parameter is learned by R^3 using the Gaussian policy model with linear basis function $\boldsymbol{\phi}(\boldsymbol{s}) = \boldsymbol{s}$.

We use the *Simulation Laboratory (SL) simulator* [12] for experiments, which is known to be highly realistic. The initial policy parameters and state[2] are randomly set as $\boldsymbol{k}_* = (\delta_*, \delta'_*, \delta''_*)^\top$, $\sigma_* = 0.2$, and $\boldsymbol{s}_* = (x_*, \dot{x}_*, \alpha_*)^\top = (\kappa_*, 0, 0.15)^\top$, where $\{\delta_*, \delta'_*, \delta''_*\}$ and κ_* are independently drawn the uniform distributions on $[0, 0.1]$ and $[0.045, 0.075]$, respectively. The dataset collected in each iteration consists of 5 episodes with 400 steps; the duration of an episode is 4 [s] in which[3] the sampling cycle and the R^3 control cycle are both 10 [ms]. We consider three scenarios here: sample reuse with $\nu = 0$ (fixed), $\nu = 1$ (fixed), and the proposed R^3 (ν is chosen by IWCV from $\{0.0, 0.2, 0.4, \ldots, 1.0\}$). The discount factor is set to $\gamma = 0.99$. Fig.9 depicts the averaged return over 10 trials as a function of the number of RWR iterations. The graph shows that R^3 nicely improves the performance. On the other hand, the performance using fixed $\nu = 0$ or $\nu = 1$ is saturated after the 2nd or 3rd iteration.

Overall, the proposed R^3 method is shown to be still promising in a complex robot-control task.

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 the 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.

The proposed idea of using importance sampling techniques in the direct policy search is applicable to other policy search methods such as the *policy gradient* method [18, 17], the *natural-policy gradient* method [5, 9], and *policy search by dynamic programming* [1]. We will develop these variants and evaluate their performance in the future work.

Acknowledgements

HH acknowledges GCOE Computationism as a Foundation for the Sciences, and MS thanks MEXT Grant-in-Aid for Young Scientists (A), 20680007, SCAT, and AOARD.

[2] The angle of elbow α_{pitch} is offset by $\pi/2$.

[3] The time cycle of PD control is 2 [ms]. Thus, PD control is applied 5 times in each R^3 control.

References

1. J. A. Bagnell, S. Kakade, A. Y. Ng, and J. Schneider. Policy search by dynamic programming. In *Neural Information Processing Systems 16*, 2003.
2. Peter Dayan and Geoffrey E. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.
3. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.
4. H. Hachiya, T. Akiyama, M. Sugiyama, and J. Peters. Adaptive importance sampling with automatic model selection in value function approximation. In *Proceedings of the Twenty-Third National Conference on Artificial Intelligence*, 2008.
5. S. Kakade. A natural policy gradient. In *Neural Information Processing Systems 14*, pages 1531–1538, 2002.
6. J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Neural Information Processing Systems 21*, 2008.
7. C. R. Peshkin, L. Shelton. Learning from scarce experience. In *Proceedings of International Conference on Machine Learning*, pages 498–505, 2002.
8. J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the International Conference on Machine Learning*, 2007.
9. J. Peters, S. Vijayakumar, and S. Shaal. Natural actor-critic. In *Proceedings of the 16th European Conference on Machine Learning*, pages 280–291, 2005.
10. D. Precup, R. S. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of International Conference on Machine Learning*, pages 759–766, 2000.
11. C. R. Rao. *Linear Statistical Inference and Its Applications*. Wiley, 1973.
12. S. Schaal. *The SL Simulation and Real-Time Control Software Package*. University of Southern California, 2007.
13. C. R. Shelton. Policy improvement for POMDPs using normalized importance sampling. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 496–503, 2001.
14. H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.
15. M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(May):985–1005, 2007.
16. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.
17. R. S. Sutton, M. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.
18. R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.