# Robust On-Policy Data Collection for Data-Efficient Policy Evaluation

**Rujie Zhong**
University of Edinburgh
Edinburgh, Scotland, UK
329053269@qq.com

**Josiah P. Hanna**
University of Wisconsin–Madison
Madison, WI, United States
jphanna@cs.wisc.edu

**Lukas Schäfer**
University of Edinburgh
Edinburgh, Scotland, UK
l.schaefer@ed.ac.uk

**Stefano V. Albrecht**
University of Edinburgh
Edinburgh, Scotland, UK
s.albrecht@ed.ac.uk

## Abstract

This paper considers how to complement offline reinforcement learning (RL) data with additional data collection for the task of policy evaluation. In policy evaluation, the task is to estimate the expected return of an *evaluation* policy on an environment of interest. Prior work on offline policy evaluation typically only considers a static dataset. We consider a setting where we can collect a small amount of additional data to combine with a potentially larger offline RL dataset. We show that simply running the evaluation policy – on-policy data collection – is sub-optimal for this setting. We then introduce two new data collection strategies for policy evaluation, both of which consider previously collected data when collecting future data so as to reduce distribution shift (or sampling error) in the entire dataset collected. Our empirical results show that compared to on-policy sampling, our strategies produce data with lower sampling error and generally lead to lower mean-squared error in policy evaluation for any total dataset size. We also show that these strategies can start from initial off-policy data, collect additional data, and then use both the initial and new data to produce low mean-squared error policy evaluation without using off-policy corrections.

## 1 Introduction

Sequential decision making tasks such as web-marketing and robot control can often be formalized as Markov decision processes and solved using reinforcement learning (RL). RL algorithms produce policies that map states of the world to actions in order to maximize the expected return, i.e., the sum of a scalar reward signal over time. The sub-problem of policy evaluation in RL is the problem of estimating the expected return of a particular *evaluation policy* using a finite set of data collected by interacting with the task of interest. Policy evaluation allows practitioners to estimate the expected return of a particular policy before extensively deploying it.

In many RL applications, data-efficient policy evaluation is of the upmost importance – we desire the most accurate estimate with minimal collected data. Towards data-efficient policy evaluation, much research has gone into how to most efficiently use a set of already collected data [Precup et al., 2000, Thomas and Brunskill, 2016]. Comparatively less research has considered the question of how to improve data collection for the data-efficiency of policy evaluation. This paper focuses on this latter question.

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2021

In the policy evaluation literature, a widely held assumption is that the optimal data collection strategy is to simply run the evaluation policy [c.f. Sutton and Barto, 1998, Chapter 5]. This strategy, known as on-policy data collection, typically leads to lower variance policy evaluation than off-policy data collection, in which a different policy (called the *behaviour policy*) is executed to collect data. One exception to this assumption is the work of Hanna et al. [2017] who show that off-policy evaluation can be more data-efficient than on-policy evaluation if importance sampling [Precup et al., 2000] is the policy evaluation method, and propose the behavior policy gradient (BPG) algorithm to search for a behavior policy that can increase the probability of rare, high-magnitude returns. However, both on-policy data collection and BPG fail to consider what additional data best supplements the previously collected data. Specifically, on-policy data collection samples each trajectory independently and BPG only attempts to minimize the error of policy evaluation on data collected in the future.

In this paper, we study the problem of how to take previously collected data into account when collecting data for Monte Carlo policy evaluation. We design two data collection strategies* – robust on-policy sampling (ROS) and robust on-policy acting (ROA) – both of which consider previously collected data when collecting new data to move the empirical distribution of the entire dataset closer to the expected on-policy distribution (referred to as lower *sampling error*). We then show across multiple RL domains that data from ROS and ROA generally leads to lower mean-squared error (MSE) policy evaluation than Monte Carlo evaluation with on-policy sampling data collection. We also show that ROS and ROA can be initialized with offline data and collect additional data to be able to use the entire dataset *without off-policy estimation methods*. While both ROS and ROA lower MSE compared to on-policy data collection, ROA generally lowers MSE further while ROS is a simpler strategy to implement. Taken together, our contributed strategies and empirical study demonstrate the promise of taking past data into account when collecting data for policy evaluation in reinforcement learning.

## 2 Related Work

Data collection is a fundamental part of the RL problem. The most widely studied data collection problem is the question of how an agent should explore its environment to learn an optimal policy [Ostrovski et al., 2017, Tang et al., 2017]. Many approaches introduce intrinsic rewards [Oudeyer and Kaplan, 2009, Bellemare et al., 2016, Pathak et al., 2017] or consider uncertainty in their value function [Osband et al., 2016, Zintgraf et al., 2020, Ciosek et al., 2019] to guide policy optimisation. Schäfer et al. [2021] instead propose to learn a separate exploration strategy to collect off-policy data used to optimise a policy. In contrast to these approaches, our work focuses on the question of how an agent should collect data to evaluate a fixed policy. When given a choice of how to collect data for policy evaluation, on-policy data collection is generally preferable to off-policy data collection Sutton and Barto [1998]. One exception is the adaptive importance sampling method introduced by Hanna et al. [2017] which leads to lower MSE policy evaluation than on-policy Monte Carlo sampling. Adaptive importance sampling has also been used in RL for policy gradient learning Ciosek and Whiteson [2017], Bouchard et al. [2016] and temporal difference learning Frank et al. [2008]. Adaptive importance sampling focuses on lowering variance in estimates from future samples while our work aims to lower variance in the estimate computed from both past and future samples.

The strategies we introduce in this paper are motivated by the idea of decreasing sampling error in collected data. Previous work has considered how sampling error can be reduced *after* data collection by re-weighting the obtained samples. For example, Hanna et al. [2021] show how importance sampling with an estimated behavior policy can lower sampling error and lead to more accurate policy evaluation. Similar methods have also been studied for policy evaluation in multi-armed bandits Li et al. [2015], Narita et al. [2019], temporal-difference learning Pavse et al. [2020], and policy gradient RL Hanna et al. [2021]. All of these prior works assume that data is available a priori and ignore the question of how to collect data if it is unavailable.

Finally, the idea of adapting the *sampling distribution*, (i.e., behavior policy) has analogs outside of policy evaluation in Markov decision processes. O'Hagan [1987] identifies flaws in Monte Carlo sampling that motivate taking past samples into account. Rasmussen and Ghahramani [2003] use

*We provide an open-source implementation of this work here: `https://github.com/uoe-agents/robust_onpolicy_data_collection`

Gaussian processes to represent uncertainty in an expectation to be evaluated and use this uncertainty to guide future sample generation. However, we are unaware of any adaptations of these ideas for RL.

# 3 Preliminaries

## 3.1 Notation

We assume the environment is an episodic *Markov decision process* with state set $\mathcal{S}$, action set $\mathcal{A}$, transition function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, discount factor $\gamma$, and initial state distribution $d_0$ Puterman [2014]. For simplicity, we assume that $\mathcal{S}$ and $\mathcal{A}$ are finite though all methods discussed are applicable to infinite-sized state and action spaces and our empirical analysis considers both settings. We assume that the transition and reward functions are unknown. A policy, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, is a function mapping states and actions to probabilities. We use $\pi(a|s) \coloneqq \pi(s, a)$ to denote the conditional probability of action $a$ given state $s$ and $P(s'|s, a) \coloneqq P(s, a, s')$ to denote the conditional probability of state $s'$ given state $s$ and action $a$.

Let $h \coloneqq (s_0, a_0, r_0, s_1, \dots, s_{l-1}, a_{l-1}, r_{l-1})$ be a possible *trajectory* and $g(h) \coloneqq \sum_{t=0}^{l-1} \gamma^t r_t$ be the *discounted return* of $h$. Any policy induces a distribution over trajectories, $\Pr(h|\pi)$. We define the *value* of a policy, $v(\pi) \coloneqq \sum_h \Pr(h|\pi) g(h)$, as the expected discounted return when sampling a trajectory with policy $\pi$.

## 3.2 Policy Evaluation

In the policy evaluation problem, we are given an *evaluation policy*, $\pi_e$, for which we would like to estimate $v(\pi_e)$. Conceptually, algorithms for policy evaluation involve two steps: collecting data (or receiving previously collected data) and producing an estimate. We assume that data is collected by running a policy which we call the *behavior policy*. If the behavior policy is the same as the evaluation policy data collection is *on-policy*; otherwise it is *off-policy*. Whether on-policy or off-policy, we assume the data collection process produces a set of trajectories, $D \coloneqq \{H_i\}_{i=1}^n$. The final value estimate is then computed by a policy evaluation method (PE) that maps the set of trajectories to a scalar-valued estimate of $v(\pi_e)$. Our goal is policy evaluation with low *mean squared error* (MSE), which is formally defined as:

$$\mathrm{MSE}\Big[\mathrm{PE}\Big] \coloneqq \mathbf{E}\left[\left(\mathrm{PE}(D) - v(\pi_e)\right)^2\right], \tag{1}$$

where the expectation is taken with respect to the collected data, $D$. MSE is the most common metric in the policy evaluation literature [e.g. Thomas and Brunskill, 2016, Hanna et al., 2019].

## 3.3 Monte Carlo Policy Evaluation

Perhaps the most fundamental, model-free policy evaluation method is the *on-policy Monte-Carlo* (MC) estimator. As an on-policy method, the Monte Carlo estimator collects data by sampling trajectories with $\pi_e$, which is known as on-policy sampling. The estimate of $v(\pi_e)$ is then the mean return:

$$\mathrm{MC}(D) \coloneqq \frac{1}{n}\sum_{i=1}^{n} g(H_i) = \sum_h \Pr(h|D) g(h), \tag{2}$$

where $\Pr(h|D)$ denotes the empirical probability of $h$, i.e. how often $h$ appears in $D$. This estimator is unbiased and strongly consistent given mild assumptions Sen and Singer [1993]. However, this method can have high variance as on-policy sampling may require many trajectories for the empirical trajectory distribution $\Pr(h|D)$ to accurately approximate $\Pr(h|\pi_e)$. Since on-policy sampling collects each trajectory i.i.d., it relies on the law of large numbers for an accurate weighting on each possible return. We refer to the discrepancy between $\Pr(h|D)$ and $\Pr(h|\pi_e)$ as *sampling error*.

# 4 Data Completion Problem

Our starting point for this work is the question, "given an available dataset of trajectories, how should additional trajectories be collected for minimal MSE policy evaluation of a fixed evaluation policy?"

This question amounts to asking what behavior policy should be used to collect additional trajectories. The answer to this question depends on the policy evaluation estimator used to produce the value estimate. In this paper we focus on the sample-average Monte Carlo estimator without importance sampling which may be the simplest method for policy evaluation.

Given our choice of Monte Carlo estimation, on-policy data collection may seem like the only generally viable choice for accurate policy evaluation. However, when including previously collected trajectories in the policy evaluation data, on-policy future data collection for the Monte Carlo estimator may produce an estimate that is biased in expectation – even if the older trajectories were collected on-policy as well. Consider a one-step MDP with one state, two actions $a_0$ and $a_1$. The return following $a_0$ is 2 and the return following $a_1$ is 4. The evaluation policy puts equal probability on both actions and suppose that, after sampling 3 trajectories, we have observed $\{a_0, 2\}$ twice and $\{a_1, 4\}$ once. If we collect an additional trajectory with $\pi_e$ the expected value of the Monte Carlo estimate is: $\frac{1}{4}(2 + 2 + 4 + 2\pi_e(a_0) + 4\pi_e(a_1)) = \frac{11}{4} = 2.75$. The true value, $v(\pi_e) = 3$ and thus, *conditioned on prior data*, the Monte Carlo estimate is biased in expectation. If instead we choose the behavior policy such that $\pi_b(a_1) = 1$ then the expected value of the Monte Carlo estimate becomes: $\frac{1}{4}(2+2+4+4) = \frac{12}{4} = 3$. By considering the previously collected data when choosing the behavior policy for collecting additional data the final estimate is less biased in expectation. This example highlights that adapting the behavior policy to consider previously collected data can lead to more accurate policy evaluation.

We call this problem – how to choose a behavior policy that depends on what data is already collected – the dataset completion problem (DCP). DCP has a resemblance to the behavior policy search problem introduced by Hanna et al. [2017]. The crucial difference is that Hanna et al. only considered how to minimize MSE using the data newly collected by the chosen behavior policy. We are interested in minimizing MSE using the newly collected data *and* the data already collected. In the next section, we will introduce two behavior policy adaptation strategies that improve upon on-policy sampling for the Monte Carlo estimator.

# 5 Robust On-Policy Data Collection

In this section, we introduce two data collection strategies that adapt the data collecting behavior policy online so as to minimize sampling error in the Monte Carlo estimator. These strategies are also able to start with data collected from a different policy than $\pi_e$ and collect additional data to reduce sampling error in the entire dataset, allowing policy evaluation without using off-policy corrections.

To reduce sampling error, our goal is to reduce the discrepancy between the evaluation policy $\pi_e$ and the *empirical policy* $\pi_D$, which is the maximum-likelihood policy under collected data $D$. As the distribution of the collected data $D$ could be decomposed as $\Pr(h|D) \approx \Pr(h|\pi_D)$, we are more likely estimating the policy value of $\pi_D$ when performing Monte Carlo estimation, and thus may get large error if there is a large difference between $\pi_D$ and $\pi_e$. In the next two sub-sections, we will introduce two concrete strategies for reducing this discrepancy (sampling error). In particular, we assume that the evaluation policy is from a class of policies parameterized by a vector $\boldsymbol{\theta}$ and we write $\boldsymbol{\theta}_e$ to denote the evaluation policy's parameter values, and $\pi_{\boldsymbol{\theta}}$ to denote a policy with parameters $\boldsymbol{\theta}$.

## 5.1 Robust On-policy Sampling

Our first strategy – **R**obust **O**n-Policy **S**ampling (ROS) – reduces sampling error by adapting the behavior policy $\pi_b$ for future data collection, which – compared to $\pi_e$ – has higher probabilities for under-sampled actions and lower probabilities for over-sampled actions, i.e., $(\pi_b - \pi_e)(\pi_D - \pi_e) \leq 0$. If we perform one-step gradient ascent on the log-likelihood evaluated at $\boldsymbol{\theta}_e$, we obtain a new policy that is closer to $\pi_D$. ROS therefore updates the behavior policy in the *opposite* direction with a step of gradient *descent* on the log-likelihood evaluate at $\boldsymbol{\theta}_e$. Pseudocode for ROS is shown in Algorithm 1. Importantly, we can use an incremental implementation to store and compute the gradient. In this way, we maintain linear per-time-step computation while considering all historical data in the behavior policy update.

While we will show empirically that ROS reduces sampling error, we highlight two limitations that motivate the second strategy we introduce. Firstly, choosing under-sampled actions cannot always reduce sampling error. Suppose we have the evaluation policy with $\pi_e(a_1|s) = 0.9$ and

**Algorithm 1** Robust On-policy Sampling.

**Input:** evaluation policy $\pi_e$, step size $\alpha$
Initialize $\nabla \leftarrow 0$
**for** step $i$ **do**
  Observe $s$ from the environment
  $\boldsymbol{\theta}_b \leftarrow \boldsymbol{\theta}_e - \alpha\nabla$
  Choose $a \sim \pi_{\boldsymbol{\theta}_b}(s)$
  $\nabla \leftarrow \frac{i}{i+1}\nabla + \frac{1}{i+1}\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_e}$
**end for**

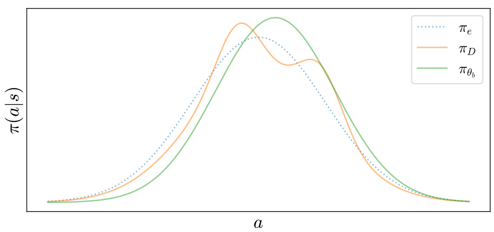



Figure 1: Distribution of evaluation policy $\pi_e$, empirical policy $\pi_D$ and ROS behavior policy $\pi_{\boldsymbol{\theta}_b}$. This figure gives an example where ROS fails to increase the probability of all under-sampled actions because it is constrained by the functional form of the policy class.

$\pi_e(a_2|s) = 0.1$. If we have only collected one-step data of action $a_1$ in state $s$, the empirical policy can be denoted as $\pi_D(a_1|s) = 1$ and $\pi_D(a_2|s) = 0$. At this point, action $a_2$ is a under-sampled action for state $s$. However, if we choose action $a_2$, the empirical policy will become $\pi_D(a_1|s) = 0.5$ and $\pi_D(a_2|s) = 0.5$, which will significantly change the empirical policy and enlarge sampling error. This phenomenon (referred to as *over-correction*) could easily occur when using ROS with large step sizes.

Second, in continuous action domains, the parameterization of policy class $\pi_{\boldsymbol{\theta}}$ may be too inflexible to adequately reduce sampling error. For example, a common distribution used in these domains is the normal distribution where the policy is defined as $\pi(a|s) := \mathcal{N}(a, \mu, \sigma^2)$, where $\mu$ and $\sigma^2$ are the mean and variance, respectively. In this case the ROS behavior policy $\pi_{\boldsymbol{\theta}_b}$ can only output a normal distribution as it shares the same structure as the evaluation policy. As an example, we may obtain the ROS behavior policy $\pi_{\boldsymbol{\theta}_b}$ with $\pi_e$ and $\pi_D$ as shown in Figure 1. This behavior policy fails to increase the probability of some under-sampled actions.

### 5.2 Robust On-policy Acting

To mitigate both problems of ROS, we propose **R**obust **O**n-policy **A**cting (ROA) that aims to identify and choose the action that can minimize the magnitude of the gradient of the log-likelihood, as shown in Algorithm 2. This strategy is based on the observation that when there is no sampling error, the gradient of the log-likelihood should be zero when evaluated at $\boldsymbol{\theta}_e$. As the empirical policy $\pi_D$ is the maximum likelihood policy of collected data $D$, its parameters can be obtained by $\boldsymbol{\theta}_D := \text{argmax}_{\boldsymbol{\theta}} \sum_{s,a \in D} \log \pi_{\boldsymbol{\theta}}(a|s)$. When there is no sampling error, $\boldsymbol{\theta}_D = \boldsymbol{\theta}_e$ and the log-likelihood has zero-gradient when evaluated at $\boldsymbol{\theta}_e$. This observation provides a goal for adapting the behavior policy to reduce sampling error: taking actions that reduce the magnitude of the gradient of the log-likelihood of actions in $D$.

Deterministicly choosing the gradient minimizing action can fail to reduce sampling error when done in a state that has not been previously visited. To reduce sampling error, the algorithm should only consider the historical data in states similar to the current state. To mitigate this issue, ROA reserves a probability of $1 - \rho$ to sample actions from $\pi_e$ (line 6), so as to collect data for identifying the minimum-sampling-error action precisely.

For computational tractability, ROA only considers an $m$-action subset for choosing the minimum-gradient action in continuous action domains. This finite subset is obtained by computing the inverse

**Algorithm 2** Robust On-policy Acting.

1: **Input:** evaluation policy $\pi_e$, correction probability $\rho$, potential action number $m$
2: Initialize $\nabla \leftarrow 0$
3: **for** step $i$ **do**
4:     Observe $s$ from the environment
5:     Sample $u$ from $\mathcal{U}(0, 1)$
6:     **if** $u < \rho$ **then**
7:         **if** $\mathcal{A}$ is a finite set **then**
8:             $\widetilde{\mathcal{A}} \leftarrow \mathcal{A}$
9:         **else**
10:             $\widetilde{\mathcal{A}} \leftarrow \left\{ F^{-1}(\frac{i}{m+1}) \right\}_{i=1}^{m}$ where $F^{-1}$ is the inverse CDF of evaluation distribution $\pi_e(s)$.
11:         **end if**
12:         $a = \operatorname*{argmin}_{a' \in \widetilde{\mathcal{A}}} \| \frac{i}{i+1} \nabla + \frac{1}{i+1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a'|s)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_e} \|$
13:     **else**
14:         Choose $a \sim \pi_e(s)$
15:     **end if**
16:     $\nabla \leftarrow \frac{i}{i+1} \nabla + \frac{1}{i+1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_e}$
17: **end for**

cumulative distribution function (CDF) of $\pi_e$ and choosing actions at uniform percentiles (line 10). Such a subset considers a wide range of actions while focusing on actions more likely to be represented in the data.

## 6 Empirical Study

We next conduct an empirical study of our proposed strategies for the dataset completion problem. In this empirical study, our goal is to answer the following questions:

1. Do ROS and ROA collect data with lower sampling error?
2. Do ROS and ROA lower the MSE of policy evaluation when starting with or without off-policy data?
3. How is the MSE of ROS and ROA affected by different evaluation policies, hyper-parameter and environment settings?

In particular, we use KL-divergence $D_{\text{KL}}(\pi_D \parallel \pi_e)$ of the empirical policy $\pi_D$ and the evaluation policy $\pi_e$ to measure sampling error. Similar measurement has also been applied in offline RL [Jaques et al., 2019], and the complete definition is presented in Appendix G

We conduct policy evaluation experiments in four domains covering discrete and continuous state and action spaces: A multi-armed bandit problem [Sutton and Barto, 1998], Gridworld [Hanna et al., 2017], CartPole and Continuous CartPole [Brockman et al., 2016]. Since these domains are widely used, we omit their description here; full details are presented in Appendix A. Our primary point of comparison is to on-policy sampling (OS) with Monte Carlo estimation. We also compare to Behavior Policy Gradient (BPG) which is designed to find a minimum variance behavior policy for ordinary importance sampling [Hanna et al., 2017].

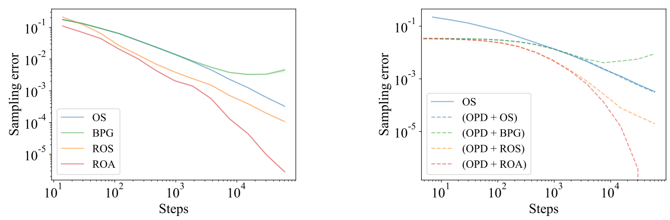
We obtain the evaluation policy, $\pi_e$, in each domain by improving an initial random policy with REINFORCE [Williams, 1992] and stopping once the policy has improved but is still far from convergence. We obtain the true value $v(\pi_e)$ and the average episode steps $\overline{T}$ in each domain with $10^6$ Monte Carlo roll-outs. Additional details are found in Appendix B.

### 6.1 Policy Evaluation without Initial Data

We first start from the setting where we have no available initial data and collect data from scratch for policy evaluation. For each domain, we follow different data collection strategies to collect data with $2^{13}\overline{T}$ steps, which is equivalent to around $2^{13}$ trajectories. Note that we specify the number of steps

| Policy Evaluation | MultiBandit | GridWorld | CartPole | CartPoleContinuous |
|---|---|---|---|---|
| OS - MC | 4.01e-05 $\pm$ 3.89e-06 | 2.68e-04 $\pm$ 2.42e-05 | 2.61e-05 $\pm$ 2.40e-06 | 4.44e-05 $\pm$ 4.01e-06 |
| BPG - OIS | 3.52e-05 $\pm$ 3.03e-06 | 2.81e-04 $\pm$ 2.63e-05 | 1.20e-05 $\pm$ 1.11e-06 | 3.62e-05 $\pm$ 3.54e-06 |
| ROS - MC | 2.93e-05 $\pm$ 2.79e-06 | 1.36e-05 $\pm$ 1.33e-06 | 1.01e-05 $\pm$ 9.79e-07 | 3.29e-05 $\pm$ 3.01e-06 |
| ROA - MC | 2.83e-05 $\pm$ 3.03e-06 | 2.10e-07 $\pm$ 2.39e-08 | 3.74e-06 $\pm$ 3.34e-07 | 1.02e-05 $\pm$ 1.06e-06 |

Table 1: Final MSE of policy evaluation without initial data. These results give the MSE for policy evaluation at the end of data collection, averaged over 200 trials with one standard error.




(a) Without initial data

(b) With initial data

Figure 2: Sampling error ($D_{\mathrm{KL}}\left(\pi_D \parallel \pi_e\right)$) curves of data collection in the GridWorld domain. Each strategy is followed to collect data with $2^{13}\overline{T}$ steps, and all results are averaged over 200 trials with shading indicating one standard error intervals. (2a) and (2b) show the sampling error curves of data collection without and with initial data, respectively. Axes in these figures are log-scaled.

for any evaluation, as it is a fairer control variable than the number of trajectories to compare different strategies for data-efficient policy evaluation. The hyper-parameter settings for these experiments are presented in Appendix E.

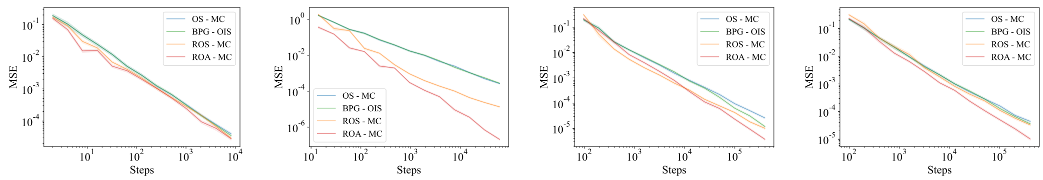
We first verify that ROS and ROA reduce sampling error compared to on-policy sampling. Figure 2a shows sampling error in the GridWorld domain, and we observe that both ROS and ROA can collect data with less sampling error than other strategies. This pattern can also hold for other domains in our study, which can be found in Appendix G. We also measure sampling error using the magnitude of the log-likelihood gradient and present results in Appendix H and these gradient curves are generally consistent with the sampling error curves. This result supports ROA identifying the minimum-gradient action as the next action to take to reduce sampling error.

Ultimately, we aim to reduce sampling error for lower MSE policy evaluation. In our next experiment, we compare different data collection strategies for policy evaluation. Figure 3 shows that both ROS and ROA can generally lead to lower MSE compared to both OS and BPG. However, the small gap between ROS and OS in Figure 3d shows the limited ability of ROS to generalize in continuous action domains. The numerical results of the final MSE are reported in Table 1. In order to evaluate the stability of different strategies, we also present the median and inter-quartile range of the squared error (SE) of the policy evaluation for these experiments in Appendix J, and results show that ROS and ROA will not introduce extra error range and can also enable estimation with lower median of SE. Other policy evaluation methods such as weighted regression importance sampling [Hanna et al., 2021] and fitted q-evaluation [Le et al., 2019] are also conducted on this collected data, and the results can be found in Appendix I, showing that these off-policy evaluation methods could also benefit from this lower-sampling-error data.

### 6.2 Policy Evaluation with Initial Data

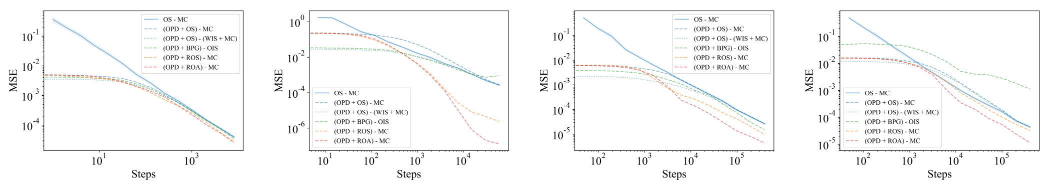
We then conduct experiments in the setting where a small amount of off-policy data (OPD) is already available, and we follow a strategy to collect additional data for policy evaluation.

To simulate this scenario, we first follow a behavior policy that is slightly different from the evaluation policy to collect 100 trajectories as OPD (more details see Appendix C). We then follow different strategies respectively to continue collecting data with $2^{13}\overline{T}$ steps. All strategies except OS will consider this OPD as the historical data and adjust their behavior policies according to the corresponding algorithms. The hyper-parameter settings for these experiments are presented in Appendix E.




(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

Figure 3: Mean squared error (MSE) of different policy evaluation without initial data. Policy evaluation is conducted on the data collection from each strategy, and these curves show the MSE of estimations and the true policy value. Lower MSE denotes more accurate policy evaluation. Steps, axes, trials and intervals are the same as Figure 2.




(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

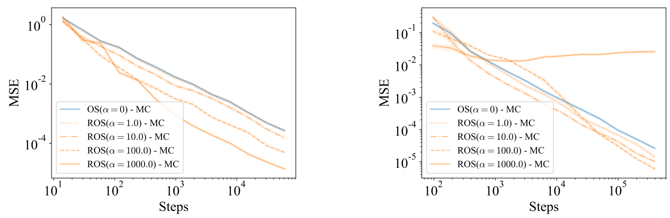
Figure 4: Mean squared error (MSE) of different policy evaluation with initial data. In particular, the estimation at step 0 is based on only the off-policy data (OPD), while there is no data available for OS *only* to make estimations and thus no MSE presented. Steps, axes, trials and intervals are the same as in Figure 3.

This newly collected data will be combined with OPD for policy evaluation. Sampling error and log-likelihood gradient are computed for each combined data, and ROS and ROA can reduce both of them faster than OS, which can be found in Appendix G and H. The MSE curves of policy evaluation are shown in Figure 4. In particular, OS can be regarded as a baseline that only uses new data for policy evaluation. Additionally, we also consider using an off-policy estimator – weighted importance sampling (WIS) – for OPD and MC for new OS data, and make combined estimations with the sum weighted by their sizes of data. It is shown from these figures that although OIS and WIS can generally make more accurate estimation for OPD than MC, ROS and ROA can collect additional data that can quickly reduce sampling error and produce lower MSE MC estimation compared to other data collection strategies. It is worth noting that in Figure 4d, it is hard for OS to correct the bias brought by OPD, while ROS and ROA are able to correct the off-policy distribution to on-policy distribution, and thus make estimation without any off-policy estimator. The numerical results of the final MSE can be found in Appendix F.

### 6.3 Hyper-parameter Sensitivity

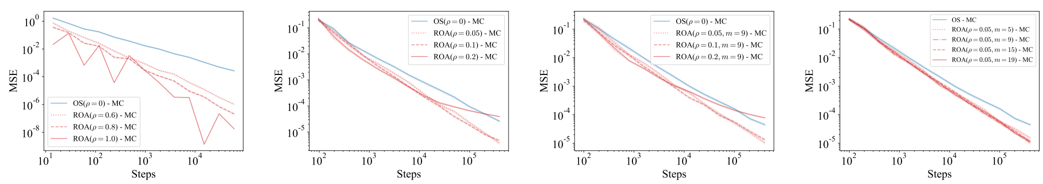
In this sub-section, we explore the effects of hyper-parameter settings on the performance of ROS and ROA, based on the settings of policy evaluation without initial data.

ROS only involves one hyper-parameter, step size $\alpha$, which controls how much the ROS behavior policy reduces its probabilities for over-sampled actions, and increases those of under-sampled actions. By taking GridWorld and CartPole as examples, we show the MSE curves of ROS with different step size $\alpha$ in Figures 5. In particular, OS can be seen as ROS with $\alpha = 0$. It can be observed from Figure 5a that ROS with large step size may produce higher MSE when collecting small sizes of data, which reflects its problem of over-correction. As it collects more data, ROS with larger $\alpha$ can generally enable lower MSE because the norm of the gradient reduces dramatically, and thus requires large $\alpha$ to make significant updates. However, in the domain with a continuous state space (Figure 5b), ROS with extremely large $\alpha$ $(1000)$ cannot reduce MSE with more data, because performing gradient descent with such a large step size on a neural network's parameters can make the behavior policy diverge.




(a) GridWorld (b) CartPole

Figure 5: MSE curves of ROS with different step size $\alpha$. These figures covers the MSE curves of ROS with $\alpha \in \{1, 10, 100, 1000\}$. Steps, axes, trials and intervals are the same as Figure 3.




(a) GridWorld (b) CartPole (c) CartPoleContinuous (d) CartPoleContinuous

Figure 6: MSE curves of ROA with different correction probability $\rho$ and potential action number $m$. (6a) (6b) (6c) consider ROA with $\rho$ in $\{0.6, 0.8, 1.0\}$ for tabular domains and $\{0.05, 0.1, 0.2\}$ for non-tabular domains, and (6d) considers $m \in \{5, 9, 15, 19\}$ for domains with continuous action space. Steps, axes, trials and intervals are the same as Figure 3.

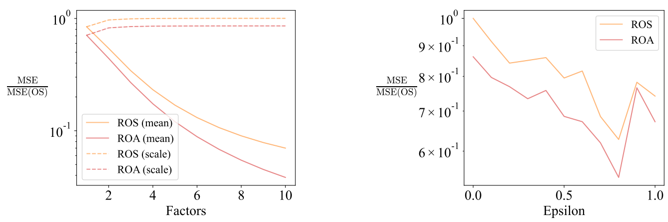
On the other hand, ROA involves two hyper-parameters, correction probability $\rho$ and potential action number $m$. By taking GridWorld, CartPole, CartPoleContinuous as examples, we show the MSE curves with different $\rho$ and $m$ in Figure 6. It can be observed that ROA can further lower the MSE with increasing $\rho$ in the pure tabular domain (Figure 6a), showing it mitigates the problem of over-correction. However, this pattern cannot hold in non-tabular domains (Figure 6b and 6c), which is because ROA with large $\rho$ cannot collect enough on-policy data to identify the minimum-sampling-error actions precisely. Another hyper-parameter $m$ is the number of actions that ROA can choose from and thus also affects the precision of ROA identifying minimum-sampling-error actions. In Figure 6d, ROA could slightly lower the MSE with $m$ increasing from 5 to 9, but could hardly further improve with $m$ further increasing.

### 6.4 Environment and Policy Sensitivity

We then explore how different environment settings and evaluation policies can affect the improvement of ROS and ROA to OS. In this sub-section, we take the MultiBandit domain as the example for simplicity, while similar experimental results in GridWorld can be found in Appendix K. In particular, we choose $\alpha = 1000$ for ROS and $\rho = 1.0$ for ROA for the following experiments.

We first create MultiBandit environments with different randomness by multiplying the means and scales of the reward by different factors. OS, ROS and ROA are followed to collect data with $1000\overline{T}$ steps and MC estimation are performed on this data, respectively. The relative MSEs are shown in Figure 7a, we can observe that 1) as the factors for means increase, MC estimation could suffer from larger error for the wrong action proportions and thus higher improvement can be brought by ROS and ROA (lower sampling error); 2) as the factors for scales increase, the environment contains more noise, which makes it harder to make accurate estimations even with low-sampling-error data.

To evaluate the improvement for evaluation policies with different randomness, we create parameterized epsilon-greedy policies with epsilon from 0 to 1. In particular, the epsilon-greedy policy with 0 epsilon represents the deterministic optimal policy, and that with 1 epsilon represents the uniformly random policy. Complete details about the creation of these policies can be found in Appendix D. We then perform the same policy evaluation as above and present the relative MSEs in Figure 7b. It can be observed that as the policies contain more stochasticity (with larger epsilon), ROS and




(a) Improvement with different environment settings.

(b) Improvement with different evaluation policies.

Figure 7: Improvement of ROS and ROA to OS with different settings in MultiBandit. (7a) shows improvement curves with changing reward mean factors and reward scale factors where higher factors represent greater randomness of the environment. (7b) shows the improvement curves with changing epsilon-greedy policies where higher epsilon represent greater randomness of $\pi_e$. Results in these figures are averaged over $500$ trials.

ROA generally improve upon OS by a larger margin, except when the policies are close to uniformly random.

## 7 Discussion and Future Work

This work has shown across multiple domains that ROS and ROA are more data-efficient than OS for accurate policy evaluation, which benefits from the consideration of previously collected data. To the best of our knowledge, these methods are the first data collection methods for policy evaluation that consider how to combine future data with the data that has already been collected. The strong performance of ROS and ROA suggests that this general approach for data collection is a direction that should be explored more in the policy evaluation literature.

In our experiments, ROA generally led to more accurate policy evaluation compared to ROS. On the other hand, ROS is simpler to implement than ROA, has one fewer hyper-parameters, and avoids the need to find a minimum over the action space. These characteristics give a trade-off for practitioners interested in using these methods. Fortunately, choosing either method lowered MSE compared to on-policy sampling.

In the future, we would like to investigate improved data collection for off-policy estimators. ROS and ROA reduce sampling error, producing data that matches the expected on-policy distribution and allowing them to use the mean return as the value estimate. An alternative could be to use data collection strategies that perform wider exploration and then rely on off-policy estimators to correct the discrepancy between the empirical distribution and the expected on-policy distribution. In preliminary results we have observed ROS and ROA reducing the MSE of weighted regression importance sampling Hanna et al. [2021] and fitted q-evaluation Le et al. [2019] compared to on-policy sampling. However, novel data collection strategies specifically designed for off-policy estimators may further increase data efficiency.

We are also interested in increasing the data efficiency of policy improvement methods with ROS and ROA. While exploration for policy improvement is typically framed as uncovering new, high-rewarding action sequences, ROS and ROA take actions to boost data efficiency. Integrating these two types of exploration for policy improvement is an interesting direction for future research.

## 8 Conclusion

In this paper we have introduced two novel strategies for data collection for policy evaluation in reinforcement learning. Our strategies – robust on-policy sampling (ROS) and robust on-policy acting (ROA) – consider previously collected data when selecting actions to reduce sampling error in the entire collected dataset. We show empirically that ROS and ROA produce lower MSE policy value estimates than on-policy sampling without the need for off-policy corrections such as importance sampling.

## References


Doina Precup, Richard S. Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 759–766, 2000.

Philip S. Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Josiah P. Hanna, Philip S. Thomas, Peter Stone, and Scott Niekum. Data-efficient policy evaluation through behavior policy search. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Georg Ostrovski, Marc G Bellemare, A ä ron Oord, and R é mi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *31st Conference on Neural Information Processing Systems (NIPS)*, volume 30, pages 1–18, 2017.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, volume 2017, 2017.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Luisa M. Zintgraf, K. Shiarlis, Maximilian Igl, Sebastian Schulze, Y. Gal, Katja Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Lukas Schäfer, Filippos Christianos, Josiah Hanna, and Stefano V. Albrecht. Decoupling exploration and exploitation in reinforcement learning. In *ICML Workshop on Unsupervised Reinforcement Learning (URL)*, 2021.

Kamil Ciosek and Shimon Whiteson. OFFER: Off-environment reinforcement learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Online learning to sample. *arXiv preprint arXiv:1506.09016*, 2016.

Jordan Frank, Shie Mannor, and Doina Precup. Reinforcement learning in the presence of rare events. In *Proceedings of the 25th International Conference on Machine Learning*, pages 336–343. ACM, 2008.

Josiah P. Hanna, Scott Niekum, and Peter Stone. Importance Sampling in Reinforcement Learning with an Estimated Behavior Policy. *Machine Learning (MLJ)*, 110(6):1267–1317, May 2021.

Lihong Li, Rémi Munos, and Csaba Szepesvári. Toward minimax off-policy value estimation. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.

Yusuke Narita, Shota Yasui, and Kohei Yata. Efficient counterfactual learning from bandit feedback. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Brahma S. Pavse, Ishan Durugkar, Josiah P. Hanna, and Peter Stone. Reducing sampling error in batch temporal difference learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Anthony O'Hagan. Monte carlo is fundamentally unsound. *The Statistician*, pages 247–249, 1987.

Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian monte carlo. *Advances in neural information processing systems*, pages 505–512, 2003.

Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Josiah Hanna, Scott Niekum, and Peter Stone. Importance sampling policy evaluation with an estimated behavior policy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2605–2613. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/hanna19a.html`.

Pranab K. Sen and Julio M. Singer. *Large Sample Methods in Statistics: An Introduction with Applications*. Chapman & Hall, 1993.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

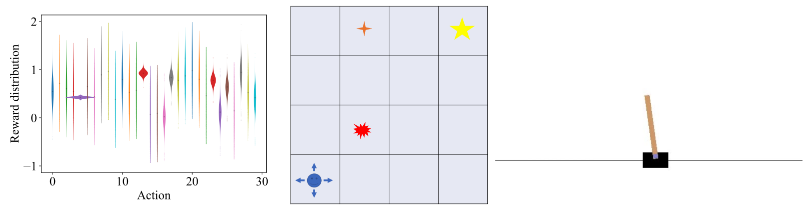
Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pages 3703–3712. PMLR, 2019.

# A Experiment Domains

In this project, we evaluate data collection strategies in 4 domains:

- **Discrete states and discrete actions:** Multi-armed bandit problem, 4x4 GridWorld.
- **Continuous states and discrete actions:** CartPole.
- **Continuous states and continuous actions:** CartPoleContinuous.




(a) MultiBandit (b) GridWorld (c) CartPole(Continuous)

Figure 8: Domains. (8a) shows the normal distribution of reward from each action in a 30-armed bandit problem, with black dots indicating the mean rewards and shading indicating the normal distributions. (8b) shows a $4 \times 4$ GridWorld with the agent starting from the bottom-left corner and the goal state in the top-right corner. (8c) shows a cart with a pole and its goal is to keep the pole from falling over.

Multi-armed Bandit problem [Sutton and Barto, 1998] is a special RL problem that only contains one state, and when the agent takes one action, it will terminate. The rewards from different actions follows different normal distributions, as shown in Figure 8a. The black dot and the shading represent the mean and distribution of reward of each action, respectively.

Gridworld [Hanna et al., 2017] is a domain with $4 \times 4$ states, shown in Figure 8b. The agent starts from $(0, 0)$ and has the action space {left, right, up, down}. The agent will get $+10$, $+1$, $-10$ rewards in terminal state $(3, 3)$, trick state $(1, 3)$ and trap state $(1, 1)$, respectively, and $-1$ rewards in all other states. The maximum number of steps is $100$ and $\gamma = 1$ in this domain.

CartPole [Brockman et al., 2016] is a continuous control domain where the agent can control the cart to go left or right to keep the pole from falling over, as shown in Figure 8c. The agent will get $+1$ reward every timestep, and terminates once the pole is more than 15 degrees from vertical or the cart moves out of the picture. The maximum number of steps is $200$ and $\gamma = 0.99$ in this domain. CartPoleContinuous is the same domain except that the agent can control the force (from $-1$ to $1$) to go left or right.

# B Pre-trained Evaluation Policy

In this paper, the policy model is built for domains with discrete action space and continuous action space as Equation 3 and 4, respectively.

$$\pi_{\boldsymbol{\theta}}(a|s) = \text{softmax}_a \left(\mathbf{w}_a^\top \phi(s)\right) \tag{3}$$

$$\pi_{\boldsymbol{\theta}}(a|s) = \mathcal{N}\left(a, \mathbf{w}_\mu^\top \phi(s), \left(\mathbf{w}_\sigma^\top \phi(s)\right)^2\right) \tag{4}$$

where $\phi$ is a parameter-free one-hot encoding operator for domains with discrete state space and a feed-forward neural network for those with continuous state space. Besides, we use $\boldsymbol{\theta}$ to denote all parameters of the policy model. Specifically, in this paper, when $\phi$ is a neural network, it is constructed with one batch normalization layer as the first layer, followed by two hidden layers, both of which have 64 hidden states and use ReLU as the activation function.

For all domains, we use REINFORCE [Williams, 1992] to train the policy model, and choose a policy snapshot during training as the evaluation policy, which has higher returns than the uniformly random policy, but is still far from convergence. To obtain the true policy value, we use on-policy sampling to collect $n = 10^6$ trajectories and compute the Monte Carlo estimation of discounted returns. We use $\mu_g$ and $\sigma_g$ to denote the mean and standard deviation of collected discounted returns, and $\sigma_\mu = \frac{\sigma_g}{\sqrt{n}}$ is the estimated standard deviation for $\mu_g$, representing the error bar of the true policy value, as shown in Table 2. To compare across different domains, we normalize the policy value to 1 by normalizing all the rewards in the experiments as $\widetilde{R} = \frac{R}{\mu_g}$.

| Domain | $\overline{T}$ | $\mu_g \pm \sigma_\mu$ | $\widetilde{\mu_g} \pm \widetilde{\sigma_\mu}$ |
|---|---|---|---|
| MultiBandit | 1.00 | 0.78 ± 4.40e-04 | 1 ± 5.61e-04 |
| GridWorld | 7.43 | 2.92 ± 4.63e-03 | 1 ± 1.58e-03 |
| CartPole | 48.48 | 36.05 ± 1.64e-02 | 1 ± 4.55e-04 |
| CartPoleContinuous | 49.56 | 35.35 ± 1.96e-02 | 1 ± 5.56e-04 |

Table 2: Information of prepared evaluation policies in different domains. $\overline{T}$ denotes the average steps of each trajectory. $\mu_g$ denotes the mean of discounted returns, $\sigma_\mu$ denotes the standard deviation of $\mu_g$, and $\widetilde{\mu_g}$, $\widetilde{\sigma_\mu}$ denote their normalized values.

## C Prepared Off-policy Data

To prepare slightly off-policy data for each domain, we create the behavior policy $\pi_b$ based on the evaluation policy $\pi_e$ in the following ways. For domains with discrete action space, the off-policy behavior policy is built as

$$\pi_b(a|s) = (1-\delta)\pi_e(a|s) + \delta\frac{1}{|\mathcal{A}|}$$

where $\delta \in (0, 1]$ controls the probability of randomly choosing an action from the action space, and otherwise sampling an action from the evaluation policy $\pi_e$. For domains with continuous action space, the off-policy behavior policy is built as

$$\pi_b(a|s) = \mathcal{N}\left(a, \mu_e(s), ((1+\delta)\,\sigma_e(s))^2\right)$$

where $\mu_e(s)$ and $\sigma_e(s)$ are the output mean and standard deviation of the evaluation policy $\pi_e$ in state $s$, and $\delta$ ($\delta > 0$) controls the increase of the standard deviation. In this paper, we collect 100 trajectories off-policy data with these behavior policies with $\delta = 0.1$ for all domains.

## D Parameterized Epsilon-greedy Policy

To create parameterized epsilon-greedy policy for discrete action domains, we build the policy model with only one parameter $w$ as:

$$\pi_w(a|s) \sim e^{w\mathbf{1}_{a=a_s^*}}$$

where $\mathbf{1}$ is the indicator function and $a_s^*$ is the optimal action of state $s$. As the epsilon-greedy evaluation policy has epsilon $\epsilon$ probability to choose actions uniformly and $1-\epsilon$ probability to choose the optimal action, it can also be denoted as

$$\pi_e(a|s) = (1-\epsilon)\mathbf{1}_{a=a_s^*} + \frac{1}{|\mathcal{A}|}\epsilon.$$

Therefore, for any specified epsilon $0 < \epsilon < 1$, we can build the policy model $\pi_w$ with $w = \log\left(\frac{|\mathcal{A}|}{\epsilon} - |\mathcal{A}| + 1\right)$.

## E Hyper-parameter Configuration

The hyper-parameter settings for policy evaluation without and with initial data are shown in Table 3 and 4, respectively. In particular, $k$ and $\alpha$ denote the batch size and step size in BPG, respectively. In experiments with initial data, we use a slightly higher hyper-parameters for ROS and ROA as they have more data available to adjust their behavior policies, and thus could obtain more accurate gradient.

Note that in CartPoleContinuous domain, we can only use small step size for BPG and ROS. That is because with large step size, both of them will update the behavior policy too much and may sample actions with probability density close to 0 under the evaluation policy $\pi_e$. This may cause the gradient to explode when continuing to update the behavior policies. In this case, the behavior policies of BPG and ROS will diverge.

| Domain | BPG - $k$ | BPG - $\alpha$ | ROS - $\alpha$ | ROA - $\rho$ | ROA - $m$ |
|---|---|---|---|---|---|
| MultiBandit | 10 | 0.01 | 1000.0 | 1.0 | N/A |
| GridWorld | 10 | 0.01 | 1000.0 | 0.8 | N/A |
| CartPole | 10 | 5e-05 | 10.0 | 0.05 | N/A |
| CartPoleContinuous | 10 | 1e-06 | 0.1 | 0.05 | 9 |

Table 3: Hyper-parameters for experiments without initial data. Note that the potential action number $m$ in ROA only matters in continuous action domains.

| Domain | BPG - $k$ | BPG - $\alpha$ | ROS - $\alpha$ | ROA - $\rho$ | ROA - $m$ |
|---|---|---|---|---|---|
| MultiBandit | 10 | 0.01 | 10000.0 | 1.0 | N/A |
| GridWorld | 10 | 0.01 | 10000.0 | 1.0 | N/A |
| CartPole | 10 | 5e-05 | 10.0 | 0.1 | N/A |
| CartPoleContinuous | 10 | 1e-06 | 0.1 | 0.1 | 9 |

Table 4: Hyper-parameters for experiments with initial data. Note that the potential action number $m$ in ROA only matters in continuous action domains.

## F Numerical Results of Policy Evaluation with Initial Data

Table 4 shows the numerical results of the final MSE of policy evaluation with initial data. It can be observed that ROA always lowers the MSE further than other baselines.

| Policy Evaluation | MultiBandit | GridWorld | CartPole | CartPoleContinuous |
|---|---|---|---|---|
| OS - MC | 4.01e-05 $\pm$ 3.89e-06 | 2.68e-04 $\pm$ 2.42e-05 | 2.61e-05 $\pm$ 2.40e-06 | 4.44e-05 $\pm$ 4.01e-06 |
| (OPD + OS) - MC | 4.05e-05 $\pm$ 3.95e-06 | 2.80e-04 $\pm$ 2.82e-05 | 2.69e-05 $\pm$ 2.70e-06 | 4.27e-05 $\pm$ 3.72e-06 |
| (OPD + OS) - (WIS + MC) | 4.01e-05 $\pm$ 3.89e-06 | 2.65e-04 $\pm$ 2.40e-05 | 2.59e-05 $\pm$ 2.51e-06 | 4.46e-05 $\pm$ 3.90e-06 |
| (OPD + BPG) - OIS | 3.49e-05 $\pm$ 3.05e-06 | 9.20e-04 $\pm$ 6.27e-04 | 1.52e-05 $\pm$ 1.53e-06 | 1.12e-03 $\pm$ 9.58e-04 |
| (OPD + ROS) - MC | 2.79e-05 $\pm$ 2.83e-06 | 2.35e-06 $\pm$ 2.42e-07 | 9.60e-06 $\pm$ 9.35e-07 | 3.27e-05 $\pm$ 3.21e-06 |
| (OPD + ROA) - MC | 2.56e-05 $\pm$ 2.64e-06 | 1.31e-07 $\pm$ 5.43e-09 | 4.38e-06 $\pm$ 4.12e-07 | 1.14e-05 $\pm$ 1.25e-06 |

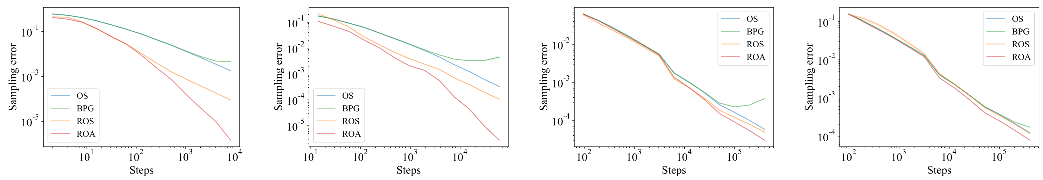
Table 5: Final MSE of policy evaluation without initial data. Trials, error ranges are the same as Figure 1.

## G KL-divergence of Data Collection

In this paper, we measure sampling error of the collected data $D$ by using KL-divergence of the empirical policy $\pi_D$ and the evaluation policy $\pi_e$ as:

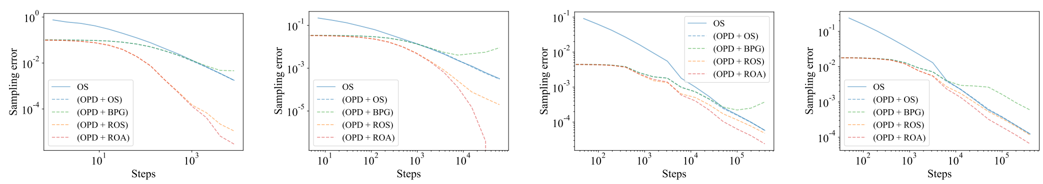
$$\begin{aligned} D_{\text{KL}}\left(\pi_D \parallel \pi_e\right) &= \mathbb{E}\left[\log \pi_D(a|s) - \log \pi_e(a|s) \mid a \sim \pi_D(a|s)\right] \\ &\approx \sum_{s,a \in D} \log \pi_D(a|s) - \log \pi_e(a|s) \end{aligned} \tag{5}$$

The sampling error curves of data collection without and with initial data are shown in Figure 9 and 10, respectively. We observe from these experiments that both ROA and ROS can collect data with lower sampling error than OS and BPG, and ROA can reduce sampling error the most.




(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

Figure 9: Sampling error ($D_{\mathrm{KL}}\left(\pi_D \parallel \pi_e\right)$) of data collection without initial data. Each strategy is followed to collect data with $2^{13}\overline{T}$ steps, and all results are averaged over 200 trials with shading indicating one standard error intervals. Axes in these figures are log-scaled.

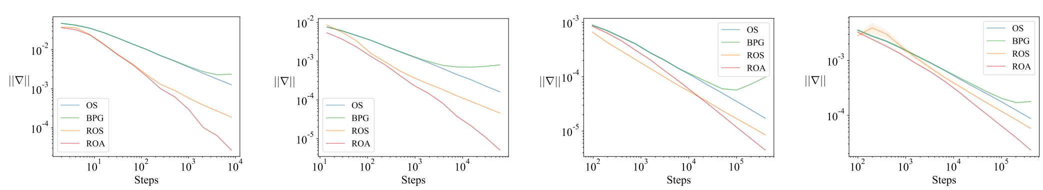



(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

Figure 10: Sampling error ($D_{\mathrm{KL}}\left(\pi_D \parallel \pi_e\right)$) of data collection with initial data. Steps, axes, trials and intervals are the same as Figure 9.

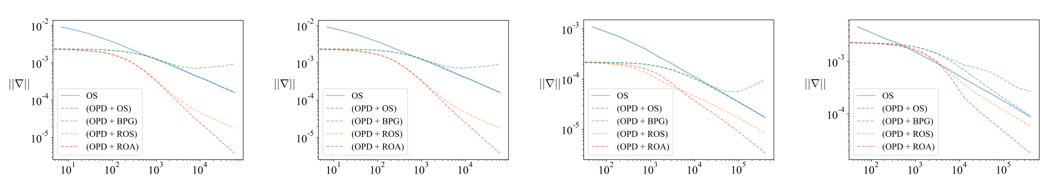
# H Log-likelihood Gradient of Data Collection

We also compute the gradient norm of log-likelihood with respect to the data collection without and with initial data, shown in Figure 11 and 12, respectively. These curves are generally consistent with their corresponding sampling error curves in Appendix G, showing the rationality of ROA identifying the minimum-gradient action as the lowest-sampling-error action.




(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

Figure 11: Log-likelihood gradient of the data collection without initial data. Steps, axes, trials and intervals are the same as Figure 9.



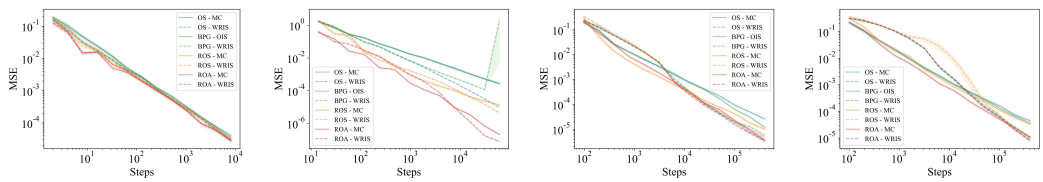

(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

Figure 12: Log-likelihood gradient of the data collection with initial data. Steps, axes, trials and intervals are the same as Figure 9.

# I Mean Squared Error of Policy Evaluation with WRIS and FQE

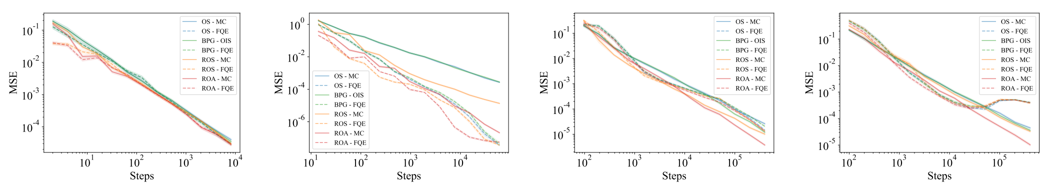
In this section, we explore the performance of ROS and ROA when using two off-policy evaluation methods: weighted regression importance sampling (WRIS) [Hanna et al., 2021] and fitted q-evaluation (FQE) [Le et al., 2019]. Based on the data collection from experiments without initial data, we present the MSE curves of WRIS and FQE in Figure 13 and 14, respectively.

It can be observed that in tabular domains, OS cannot enable the same level MSE as ROS and ROA, even with the help of WRIS, which is designed to correct sampling error during the value estimation stage. This shows the importance of reducing sampling error during the data collection stage. The same pattern can also be observed when using FQE. In non-tabular domains, the performances of off-policy evaluation methods on different data collection are very similar. However, WRIS usually requires larger sizes of data to make accurate estimation, and can only achieve similar MSE as ROA when collecting a large amount of data. When using FQE in non-tabular domains, data from ROA can generally enable lower MSE than OS, although this improvement is very small.




(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

Figure 13: MSE of WRIS without initial data. Steps, axes, trials and intervals are the same as Figure 3.

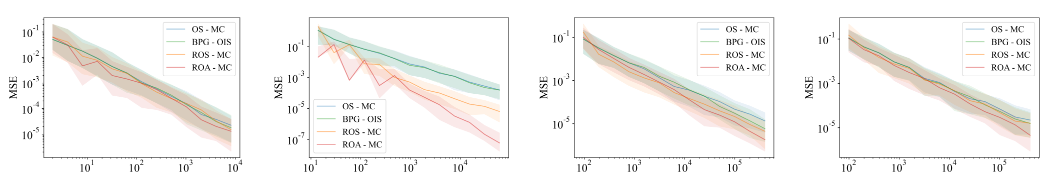



(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

Figure 14: MSE of FQE without initial data. Steps, axes, trials and intervals are the same as Figure 3.

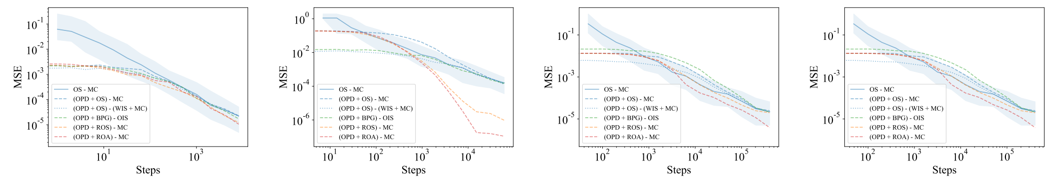
# J Median and Inter-quartile Range of Policy Evaluation

We also compute the median and inter-quartile range of the squared error (SE) of policy evaluation without and with initial data, shown in Figure 15 and 16, respectively. From these figures, we can observe that all these strategies have a similar inter-quartile range of SE, and ROA can enable the lowest median of SE among them.




(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

Figure 15: Median and inter-quartile range of squared error (SE) of policy evaluation without initial data. The lines in these figures denote the median of squared error over 200 trials, and the shading indicates the inter-quartile range. Axes in these figures are log-scaled.




(a) MultiBandit (b) GridWorld (c) CartPole (d) CartPoleContinuous

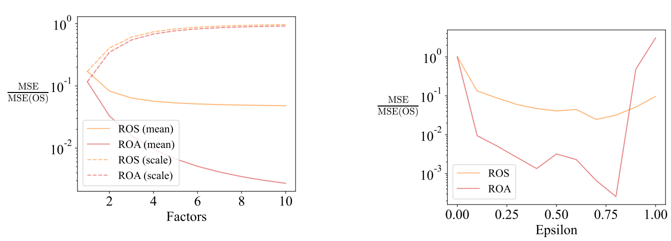
Figure 16: Median and inter-quartile range of squared error (SE) of policy evaluation with initial data. Axes, trials and intervals are the same as Figure 15.

## K Environment and Policy Sensitivity in GridWorld

In this section, we evaluate how the performance of ROS and ROA in GridWorld can be affected by the environment settings and the evaluation policies with different randomness. In particular, we choose $\alpha = 1000$ for ROS and $\rho = 1.0$ for ROA for the following experiments.

We first create GridWorld with different randomness in two directions: (1) multiplying the original reward by mean factors; (2) adding random noise to the original rewards, which is sampled uniformly from $[-f_{\text{scale}}, f_{\text{scale}}]$ where $f_{\text{scale}}$ is the scale factor. We then follow OS, ROS and ROA to collect 1000 trajectories, and perform Monte Carlo estimation, respectively. The relative MSEs are shown in Figure 17a.

We also create different evaluation policies using parameterized epsilon-greedy policies with epsilon $\epsilon$ from 0 to 1. Complete details about the creation of these policies can be found in Appendix D. We then perform the same policy evaluation as above and compute the relative MSEs, shown in Figure 17b.

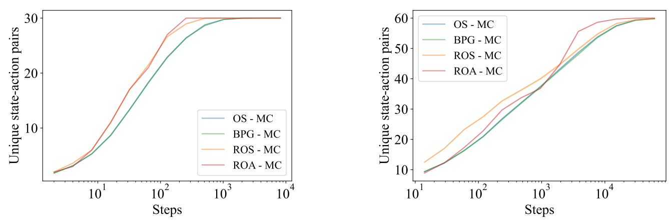



(a) Different environment settings. (b) Different evaluation policies.

Figure 17: Improvement of ROS and ROA to OS with different settings in GridWorld. Axes and trials are the same as Figure 7.

## L Richness of Unique State-action Pairs

In this section, we explore the count of unique state-action pairs in different data collection to evaluate their abilities to collect *new* information. We take MultiBandit and GridWorld as the experiment domains, as the agent can never experience the same state-action pairs in non-tabular domains. In Figure 18, we show this count of unique state-action pairs along data collection. It can be observed that both ROS and ROA tend to collect more unique state-action pairs than the baseline strategies (OS and BPG) before they reach the maximum number. This results can also explain the better performance of ROS and ROA when using off-policy evaluation methods in Appendix I, which usually require more unique state-action pairs.




(a) MultiBandit (b) GridWorld

Figure 18: Unique State-action pairs of Different Data Collected Strategies.