Data collection for policy evaluation in reinforcement learning

Rujie Zhong

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2021

Abstract

In reinforcement learning (RL), policy evaluation is a task to estimate the expected returns when deploying a particular policy (called the evaluation policy) to the real environment. One common strategy to conduct policy evaluation is On-policy Sampling (OS), which collects data by sampling actions from the evaluation policy, and makes estimations by averaging the collected returns. However, due to sampling error, the distribution of OS data collection could be arbitrarily different from that under the evaluation policy, making the value estimation suffer from a large variance.

In this work, we propose two data collection strategies, Robust On-policy Sampling and Robust On-policy Acting, both of which can consider the historical collected data and reduce sampling error in future data collection. Experiments in different RL domains show that limited to the same amount, this low-sampling-error data can generally enable a more accurate policy value estimation.

Acknowledgements

I would like to kindly thank my supervisor Stefano Albrecht, co-supervisors Josiah Hanna and Lukas Schäfer for their insightful comments and guidance throughout this project.

Declaration

Table of Contents

Chapter 1

Introduction

Reinforcement learning (RL) is a learning task that aims to train a policy that can maximize a numerical reward when interacting with the environment. One crucial problem when applying reinforcement learning to real problems is policy evaluation. The goal is to estimate the expected returns produced by a particular policy (called the evaluation policy π_e). To accomplish this, two steps are indispensable: *data collection* and *value estimation*. In many RL applications like recommender systems [Afsar et al., 2021] and robot control [Polydoros and Nalpantidis, 2017], data collection could be very time-consuming and expensive. Therefore, it is important to design a data collection strategy that leads to lower error value estimation with a limited number of samples. Although exploration has been well studied in terms of RL agent training [Ostrovski et al., 2017, Tang et al., 2017], its goal is to efficiently collect data that can produce a higher return, rather than more accurate policy evaluation. Apart from this, there is a lack of research in this field and thus a huge research gap.

The most common data collection strategy is On-Policy Sampling (OS), which collects data by sampling actions from the evaluation policy. The main idea of this strategy is that it assumes the distribution of data collection is close to the distribution under the evaluation policy (called the evaluation distribution). To make value estimation on this data, one standard method is Monte Carlo estimation (MC), which estimates the policy value by averaging the total returns of all collected data [Sutton and Barto, 2018]. However, as OS samples data independently for each step, it could easily end up with the data collection where some actions are collected too often and some others too rarely, which we refer to as *sampling error*. With this biased data distribution, the value estimation could be of low accuracy.

In this work, we design and evaluate two robust on-policy strategies which can consider the historical sampling error, and be able to correct this error in future data collection. Experimental results show that these data collection strategies can generally produce a more accurate value estimation than other baseline strategies with the same amount of data.

1.1 Motivation

Based on whether aiming to collect data with distribution close to the evaluation distribution, data collection strategies can be divided into two categories: *on-policy strategy* and *off-policy strategy*. To our best knowledge, On-Policy Sampling (OS) is the only on-policy strategy in existing work. However, due to sampling error, the OS data distribution may be arbitrarily different from the evaluation distribution. To make more accurate estimations, Regression Importance Sampling (RIS) is designed to correct this error in the value estimation stage [Hanna et al., 2019]. It first estimates the empirical policy π_D, which is the policy that is most likely to generate the collected data, and uses the importance sampling technique to re-weight each sample of data. However, the importance ratio in RIS is computed according to the relative probability between π_e and π_D, it could lead to a large variance if there is a large difference between π_e and π_D (large sampling error). Therefore, it is necessary to reduce sampling error even with RIS as the value estimation method.

On the other hand, off-policy strategies can follow an arbitrary policy (called the behavior policy π_b) to collect data, which has a different distribution from the evaluation distribution. To mitigate this difference, the importance sampling technique is again commonly used to re-weights the returns according to the relative probability between π_e and π_b. Based on this technique, Behavior Policy Gradient (BPG) is designed to find the optimal behavior policy that can lead to the minimum of the mean square error (MSE) of the estimation [Hanna et al., 2017]. However, when using the importance sampling technique, it can usually introduce extra variance caused by the importance ratio. Although BPG can theoretically find the behavior policy that can minimize the MSE of important sampling estimation, there is still a lack of evidence whether this estimation can produce lower MSE than MC with OS data.

Therefore, this project will follow the idea of on-policy strategy, and explore how to collect on-policy data with lower sampling error.

1.2 Hypothesis

To form this work, we hold the following hypothesis:

- **Hypothesis 1**: Our robust on-policy strategies can collect data with lower sampling error. To test this hypothesis, we propose to evaluate sampling error by KL-divergence of π_D and π_e.

- **Hypothesis 2**: With the sampling error reduced, the value estimation will be more accurate.

1.3 Structure

In Chapter 2, we first review existing value estimation methods, divide them into trajectory-based and transition-based estimations, and discuss the reasons why current data collection strategies fail to collect data that can produce an accurate estimation. Among these problems, we choose to mitigate sampling error, and propose two data collection strategies (robust on-policy strategies) in Chapter 3. Theoretical necessity and effectiveness of these strategies are also discussed in this chapter. In Chapter 4, we design a series of experiments in all kinds of RL domains, to evaluate whether our proposed strategies can collect data with lower sampling error, and whether this data can enable more accurate estimations for different value estimation methods. In Chapter 5, we explore the properties of our proposed strategies, including the effects of hyper-parameters, richness of data collection and consistency. Finally, we present the conclusions of this project, and discuss the unsolved problem and possible future work in Chapter 6.

Chapter 2

Background

In this chapter, we will formally introduce the problem studied, and present relevant literature.

2.1 Problem Formulation

In reinforcement learning, Markov Decision Process (MDP) is a classical formalization of sequential decision making [Sutton and Barto, 2018], where the distribution of the current state and reward is determined by only last state and action. In this project, we assume the environment is a finite-horizon MDP with the state space $\mathcal{S}$, the action space $\mathcal{A}$, the transition model $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{R} \mapsto [0,1]$ and the initial-state distribution $P_0 : \mathcal{S} \mapsto [0,1]$. A policy $\pi : \mathcal{S} \mapsto [0,1]^{|\mathcal{A}|}$ is a mapping function from states to the distribution over actions. In particular, $\pi(a|s)$ represents the probability of choosing action a in state s. In a finite MDP problem, let $H = (S_0, A_0, R_1, S_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T)$ be a possible trajectory, where T is the terminal step. The probability of the episode H sampled under policy π can be computed as $\Pr(H|\pi) = P_0(S_0)\prod_{t=0}^{T-1}\pi(A_t|S_t)P(S_{t+1}, R_{t+1}|S_t, A_t)$. The discounted returns can be denoted as $G(H) = \sum_{t=0}^{T-1}\gamma^t R_{t+1}$ where γ is the discount factor. Policy evaluation is to estimate the expected discounted returns when deploying the *evaluation policy* π_e, which can be denoted as $v^{\pi_e} = \mathbb{E}[G(H)|H \sim \Pr(H|\pi_e)]$.

To achieve this, we usually need two steps: *data collection* and *value estimation*. Data collection strategy is to design a *behavior policy* π_b (can be dynamic or fixed), and use this policy to interact with the environment and collect data $\mathcal{D} = \{H^{(i)}\}_{i=1}^{n}$. Based on the collected data, the second step is to choose a proper value estimation method (VE) to estimate the policy value $\widehat{v^{\pi_e}} = \mathrm{VE}(\mathcal{D}, \pi_e, \pi_b)$. The goal is to minimize the difference between the estimation and the true value. In this project, we use mean squared error to measure this difference, denoted as $\mathbb{E}_{\mathcal{D}}[(\widehat{v^{\pi_e}} - v^{\pi_e})^2]$.

2.2 Policy Value Estimation

Most of the previous policy evaluation research focuses on the value estimation stage, assuming that the data is already given, and designs a method that can make an accurate estimation. This section will introduce and compare these estimation methods.

2.2.1 Monte Carlo Estimation

Monte Carlo Estimation (MC) is a method that assumes the distribution of collected data is close to the distribution under evaluation policy (evaluation distribution), i.e., $\Pr(H|\mathcal{D}) \approx \Pr(H|\pi_e)$ and therefore make estimation by averaging the resulting discounted returns [Sutton and Barto, 2018]:

$$\begin{aligned} v^{\pi_e} &= \mathbb{E}\left[G(H)|H \sim \Pr(H|\pi_e)\right] \\ &\approx \mathbb{E}\left[G(H)|H \sim \Pr(H|\mathcal{D})\right] = v^{\pi_e}_{\text{MC}} \end{aligned} \tag{2.1}$$

where $\Pr(H|\mathcal{D})$ denotes the probability of trajectory H under $\mathcal{D}$.

2.2.2 Ordinary Importance Sampling

In some cases, the data may be collected by a stationary behavior policy π_b, which is different from the evaluation policy π_e. In these cases, the distribution of the data will be different from the true distribution when deploying the evaluation policy. One standard technique to correct this difference is Ordinary Importance Sampling (OIS), which re-weights returns according to the relative probability of their trajectories sampled under the evaluation and behavior policies [Sutton and Barto, 2018].

$$\begin{aligned} v^{\pi_e} &= \mathbb{E}\left[G(H)|H \sim \Pr(H|\pi_e)\right] \\ &= \mathbb{E}\left[\frac{\Pr(H|\pi_e)}{\Pr(H|\pi_b)} G(H) \middle| H \sim \Pr(H|\pi_b)\right] \\ &\approx \mathbb{E}\left[\frac{\prod_{t=0}^{T-1} \pi_e(A_t|S_t)}{\prod_{t=0}^{T-1} \pi_b(A_t|S_t)} G(H) \middle| H \sim \Pr(H|\mathcal{D})\right] = v^{\pi_e}_{\text{OIS}} \end{aligned} \tag{2.2}$$

2.2.3 Regression Importance Sampling

Due to *sampling error*, the trajectory distribution from collected data $\mathcal{D}$ is not necessary the same as that under the behavior policy, i.e., $\Pr(H|\mathcal{D}) \neq \Pr(H|\pi_b)$. Both MC and OIS could suffer from this error, reflected as the approximation process in Equation 2.1 and 2.2. To reduce this error, Hanna et al. [2019] propose to replace π_b with the *empirical policy* π_D, which is defined as the policy that $\mathcal{D}$ is most likely generated by. The empirical policy is computed as $\pi_D = \operatorname{argmin}_\pi \mathrm{KL}\left(\Pr(H|\mathcal{D}) \,||\, \Pr(H|\pi)\right)$. Therefore, $\Pr(H|\pi_D)$ is a closer approximation of the data distribution $\Pr(H|\mathcal{D})$. Then, regression importance sampling estimation (RIS) is computed as

$$\begin{aligned} v^{\pi_e} &= \mathbb{E}\left[G(H)|H \sim \Pr(H|\pi_e)\right] \\ &\approx \mathbb{E}\left[\frac{\Pr(H|\pi_e)}{\Pr(H|\mathcal{D})}G(H)\middle|H \sim \Pr(H|\mathcal{D})\right] \\ &\approx \mathbb{E}\left[\frac{\prod_{t=0}^{T-1}\pi_e(A_t|S_t)}{\prod_{t=0}^{T-1}\pi_D(A_t|S_t)}G(H)\middle|H \sim \Pr(H|\mathcal{D})\right] = v_{\text{RIS}}^{\pi_e} \end{aligned} \tag{2.3}$$

However, as the importance ratio $\frac{\prod_{t=0}^{T-1}\pi_e(A_t|S_t)}{\prod_{t=0}^{T-1}\pi_D(A_t|S_t)}$ is unbounded, the variance of this estimation can also be unbounded. To reduce this variance, RIS can be modified with the technique of Weighted Importance Sampling [Sutton and Barto, 2018], and become a new estimator Weighted Regression Importance Sampling (WRIS) as

$$v_{\text{WRIS}}^{\pi_e} = \sum_{H\in\mathcal{D}} \frac{\frac{\prod_{t=0}^{T-1}\pi_e(A_t|S_t)}{\prod_{t=0}^{T-1}\pi_D(A_t|S_t)}}{\sum_{H\in\mathcal{D}}\frac{\prod_{t=0}^{T-1}\pi_e(A_t|S_t)}{\prod_{t=0}^{T-1}\pi_D(A_t|S_t)}} G(H) \tag{2.4}$$

In the work of Hanna et al. [2019], this estimator has been empirically verified as the one that can produce estimation with lower MSE.

2.2.4 Model-Based

Unlike the MC and IS-based estimations making estimation under the distribution of collected data $\Pr(H|\mathcal{D})$, Model-based (MB) is designed to estimate the evaluation distribution $\Pr(H|\pi_e)$, and compute the expectation under this distribution [Zhang et al., 2021, Paduraru, 2007]. To accomplish this, MB first build a probabilistic model for the transition model and the initial state distribution, and train it using the maximum log-likelihood as

$$\begin{aligned}\widehat{P} &= \text{argmax}_{P'} \sum_{H\in\mathcal{D}} \sum_{t=0}^{T} -\log P'(S_{t+1}, R_{t+1}|S_t, A_t) \\ \widehat{P}_0 &= \text{argmax}_{P'_0} \sum_{H\in\mathcal{D}} -\log P'_0(S_0)\end{aligned} \tag{2.5}$$

With these estimated models, MB can approximate the evaluation distribution as $\widehat{\Pr}(H|\pi_e) = \widehat{P}_0(S_0)\prod_{t=0}^{T-1}\pi_e(A_t|S_t)\widehat{P}(S_{t+1}, R_{t+1}|S_t, A_t)$, and make the value estimation as:

$$\begin{aligned} v^{\pi_e} &= \mathbb{E}\left[G(H)|H \sim \Pr(H|\pi_e)\right] \\ &\approx \mathbb{E}\left[G(H)\middle|H \sim \widehat{\Pr}(H|\pi_e)\right] = v^{\pi_e}_{\text{MB}}\end{aligned} \tag{2.6}$$

The different between $\Pr(H|\mathcal{D})$ and $\widehat{\Pr}(H|\pi_e)$ is that $\Pr(H|\mathcal{D}) = 0$ if $H \notin \mathcal{D}$, while $\widehat{\Pr}(H|\pi_e) > 0$ if all transitions in H is collected in $\mathcal{D}$. Therefore, MB can make full use of each transition of the collected data $\mathcal{D}$. Moreover, the accuracy of the estimation only relies on the accuracy of the estimated transition model $\widehat{P}$ and initial state distribution $\widehat{P}_0$.

2.2.5 Fitted Q-evaluation

Fitted Q-evaluation (FQE) is another that can make full use of each transition of the collected data $\mathcal{D}$, which is designed to train an action-state value function from bootstrapping data [Le et al., 2019, Paine et al., 2020].

Given policy π, the state-value function $V^{\pi}(s) = \mathbb{E}_{H\sim\Pr(H|\pi)}[\sum_{k=t}^{T-1}\gamma^t R_{t+1}|S_t = s]$ quantifies the expected discounted returns in any state s, and the action-state value function $Q^{\pi}(s,a) = \mathbb{E}_{H\sim\Pr(H|\pi)}[\sum_{k=t}^{T-1}\gamma^t R_{t+1}|S_t = s, A_t = a]$ quantifies expected discounted returns in any state s with action a. Based on this, the policy evaluation can also be denoted as $v^{\pi_e} = \mathbb{E}\left[\sum_a \pi(a|s)Q^{\pi}(S_0, a)|S_0 \sim P_0(S_0)\right]$. According to Bellman Equation, the action-state value function can be computed as $Q^{\pi}(S_t, A_t) = R_{t+1} + \gamma\sum_a \pi_e(a|S_{t+1})Q^{\pi}(S_{t+1}, a)$ [Sutton and Barto, 2018]. Therefore, FQE first estimates the state-action function as

$$\widehat{Q} = \text{argmin}_Q \mathbb{E}\left[\sum_{t=0}^{T-1}\left(Q(S_t, A_t) - R_{t+1} - \gamma\sum_a \pi_e(a|S_{t+1})Q(S_{t+1}, a)\right)^2 \middle| H \sim \Pr(H|\mathcal{D})\right] \tag{2.7}$$

Then the FQE estimation can be computed as

$$\begin{aligned} v^{\pi_e} &= \mathbb{E}\left[\sum_a \pi_e(a|S_0) Q^{\pi_e}(S_0,a) \middle| S_0 \sim P_0(S_0)\right] \\ &\approx \mathbb{E}\left[\sum_a \pi_e(a|S_0) \widehat{Q}(S_0,a) \middle| S_0 \sim \widehat{P}_0(S_0)\right] = v^{\pi_e}_{\text{FQE}} \end{aligned} \tag{2.8}$$

where $\widehat{P}_0(S_0)$ is the probability of trajectory starting with S_0 in $\mathcal{D}$.

2.2.6 Comparison

Based on whether using bootstrapping data, we can separate value estimation methods into two categories: trajectory-based estimations (MC, OIS, RIS) and transition-based estimations (FQE, MB). For trajectory-based estimations, each trajectory can only contribute one return to the averaging result, while transition-based methods can "create" infinite data based on the collected transitions. Therefore, theoretically, to estimate the true policy value, trajectory-based estimations will require collecting all possible trajectories, while transition-based methods only require collecting all possible transitions, which is much easier for MDP problems. However, with small data, these transition-based methods may not be able to "create" bootstrapping data accurately, and make estimations with extremely large error. For example, suppose we have not collect any transition of state-action pair (s,a). When using MB, the estimated transition model may predict an extreme reward, and to transfer to some other un-tested states. This may happen back and forth in MB and result in an extreme estimation. For domains with continuous state or action space, this could happen more easily as it can never collect all transitions.

Therefore, there is no one certain value estimation method that can always produce the most accurate estimation. In this work, we will evaluate the performance of our data collection strategies with different value estimation methods.

2.3 Data Collection Strategy

In this section, we will introduce the only two existing data collection strategies that are designed for policy evaluation.

2.3.1 On-policy Sampling

On-policy sampling (OS) is the most common data collection strategy, which simulates deploying the evaluation policy π_e in the environment by sampling actions from π_e to interact with the environment. When using this strategy, we expect to collect data with distribution close to the evaluation distribution, i.e., $\Pr(H|\mathcal{D}) \approx \Pr(H|\pi_e)$. The advantage of this strategy is that data with higher probability in the evaluation distribution is more likely to be collected, which is more important to the value estimation according to the policy evaluation equation $v^{\pi_e} = \sum_H \Pr(H|\pi_e)G(H)$. However, since OS samples an action for each step independently, making it unable to avoid collect the same data repeatedly, which could make little contribution to value estimation. In particular, the data with low probability under the evaluation distribution could always be hard to sample with this strategy, even if this is the only data that have not been collected.

2.3.2 Behavior Policy Gradient

Based on the OIS estimation, Hanna et al. [2017] propose a data collection strategy, Behavior Policy Gradient (BPG), which is designed to find the policy that can minimize the mean square error of OIS estimation. To accomplish this, BPG adjusts the behavior policy iteratively during data collection by performing one-step gradient descent on the loss function as

$$\begin{aligned}\text{Loss}(\pi_\theta) &= \text{MSE}(v^{\pi_e}_{\text{OIS}}, v^{\pi_e}) \\ &= \mathbb{E}\left[-\text{IS}(H,\pi_e,\pi_\theta)^2 \sum_{t=0}^{T-1} \log \pi_\theta(A_t|S_t) \mid H \sim \pi_\theta\right]\end{aligned} \tag{2.9}$$

where π_θ is the behavior policy and $\text{IS}(H,\pi_e,\pi_\theta)$ is the importance-sampled returns. As a result, this will increase the probabilities of trajectories with higher returns or importance ratios, and thus reduce the magnitude and variance of re-weighted returns.

However, practically BPG can only compute the gradient on collected data. Suppose that we have only collected data with very high probabilities, since $\text{IS}(H,\pi_e,\pi_\theta)^2$ is always positive, the probabilities of collected data will further increase, and the low-probability data could be harder to sample. In this case, BPG is not efficient to collect low-probability data, which could make it hard to further reduce the error. Another problem is that each update of BPG can only use the data collected by the current behavior policy, which makes it inefficient to find the optimal behavior policy.

Chapter 3

Methodology

In this chapter, we will discuss the problem of sampling error in On-policy Sampling, and propose two novel data collection strategies, robust on-policy strategy, that can reduce this error.

3.1 Preliminaries

For both trajectory-based and transition-based estimations, we believe on-policy data can have benefit because 1) for trajectory-based methods, importance ratio is introduced to re-weight off-policy data, which can bring more variance; 2) for transition-based methods, model generalization relies on the distribution of training data, especially in domains with continuous spaces. However, On-policy Sampling (OS) is not an efficient strategy to collect on-policy data.

To explain it, we decompose the data distribution as RIS: $\Pr(H|\mathcal{D}) \approx \Pr(H|\pi_D) \approx \prod_{t=0}^{T-1} \pi_D(A_t|S_t)P(S_{t+1}, R_{t+1}|S_t, A_t)$ where the empirical policy π_D can be seen as the approximated action density of collected data $\mathcal{D}$. Suppose at a certain point of data collection, we have the evaluation policy π_e and π_D as Figure 3.1. The difference between π_e and π_D is what we refer to as sampling error. As the goal of on-policy strategy is to approximate the evaluation distribution $\Pr(H|\pi_e)$, it is intuitive to convert it to making π_D close to π_e, i.e., reducing sampling error. However, continuing data collection using OS is not an efficient way to correct sampling error. To explain this, suppose π_D is the empirical policy for an n-step data collection in a domain with only one state. If we follow OS to collect one step of data, we will expect to update the empirical policy as $\pi_D \leftarrow (n\pi_D + \pi_e)/(n+1)$. Although this new π_D will get closer to π_e, this data collection still suffers from sampling error ($\pi_D \neq \pi_e$). Instead, if we

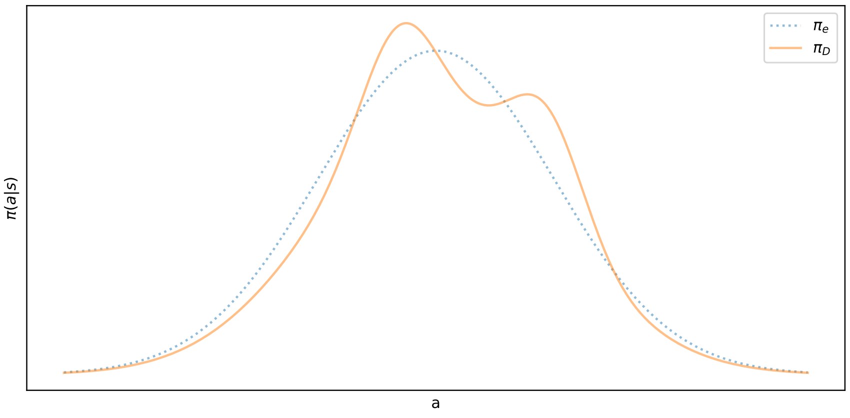


Figure 3.1: Distribution of evaluation policy π_e and empirical policy π_D.

follow the ideal behavior policy $\pi_b^* = (n+1)\pi_e - n\pi_D$ to this one step of data, we will expect to update the empirical policy as $\pi_D \leftarrow \left(n\pi_D + \pi_b^*\right)/(n+1) = \pi_e$, which will fully correct sampling error. Therefore, following this ideal behavior policy will be more an efficient strategy to reduce sampling error than OS.

To better introduce our proposed strategies in the next two sections, we assume the policy model is built for domains with discrete action space and continuous action space as Equation 3.1 and 3.2, respectively [Williams, 1992, Mnih et al., 2016].

$$\pi_\theta(a|s) = \text{softmax}_a\left(\mathbf{w}_a^\top \phi(s)\right) \tag{3.1}$$

$$\pi_\theta(a|s) = \mathcal{N}\left(a, \mathbf{w}_\mu^\top \phi(s), \mathbf{w}_\sigma^\top \phi(s)\right) \tag{3.2}$$

where ϕ 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 θ to denote all parameters of the policy model.

3.2 Robust On-policy Sampling

Following the idea above, one alternative way to get the ideal behavior policy π_b^* is that we can first obtain the empirical policy by computing $\pi_D = \text{argmin}_{\pi_\theta}\mathbb{E}\left[-\log \pi_\theta(a|s) \mid s, a \sim \mathcal{D}\right]$. However, the "argmin" computation can be very time-consuming when π_θ is a neural network as it requires many steps of gradient descent to obtain π_D.

To make it more efficient, we propose a novel data collection strategy called Robust On-policy Sampling (ROS), shown in Algorithm 1. The main idea of this strategy is

Algorithm 1 Robust On-policy Sampling.

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

to obtain and follow an approximated ideal behavior policy. At the beginning of the data collection, we initialize $\pi_\theta = \pi_e$. As we know if we perform one-step gradient ascent on $\text{Loss}(\theta) = \mathbb{E}\left[\log \pi_\theta(a|s) \mid s, a \sim \mathcal{D}\right]$, and get the new parameter θ', the new policy model $\pi_{\theta'}$ will get closer to π_D, which can be regarded as the evaluation policy π_e is updated in the direction of π_D. On the other hand, ideal behavior policy can be reformulated as $\pi_b^* = (n+1)\pi_e - n\pi_D = \pi_e + n(\pi_e - \pi_D)$ which can be seen as the evaluation policy π_e is updated in the opposite direction of π_D. Therefore, to obtain the approximated ideal behavior policy, we propose to perform one-step gradient descent on $\text{Loss}(\theta)$.

To explain more about the "direction" and why this approximated ideal behavior policy can also be efficient to reduce sampling error, we first show the inequality for the ideal behavior policy based on its expression as $(\pi_b^* - \pi_e)(\pi_D - \pi_e) < 0$. From this inequality, we can also observe that with π_e as the origin, π_b^* is in the opposite direction of π_D, meaning that it will reduce the probabilities of actions that are collected too often (with $\pi_D > \pi_e$) and increase those too rarely (with $\pi_D < \pi_e$). Therefore, with this behavior policy, sampling error can be corrected more efficiently. Although ROS does not follow the exact ideal behavior policy, we believe it is still efficient to reduce sampling error because its behavior policy is in the same direction as the ideal one.

Moreover, as the gradient is computed as $\nabla_\theta \text{Loss}(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) \mid s, a \sim \mathcal{D}]$, where $\nabla_\theta \log \pi_\theta(a|s)$ is unchanged for the historical data, we can use incremental implementation to store and compute the average gradient. In this way, we can adjust the behavior policy by considering all historical data, and the whole data collection time is only linear to the number of action steps.

3.3 Robust On-policy Acting

However, ROS is not capable of obtaining a behavior policy in the same direction as the ideal one in domains with continuous action space. That is because the ROS behavior policy $\pi_{\theta'}$ in Algorithm 1 has the same structure as π_θ in Equation 3.2, meaning that this behavior policy can only output normal distribution. Therefore, we may obtain the ROS behavior policy $\pi_{\theta'}$ as shown in Figure 3.2. It can be observed from this figure that $\pi_{\theta'}$ is not an ideal behavior policy as it cannot meet $(\pi_{\theta'} - \pi_e)(\pi_D - \pi_e) < 0$.

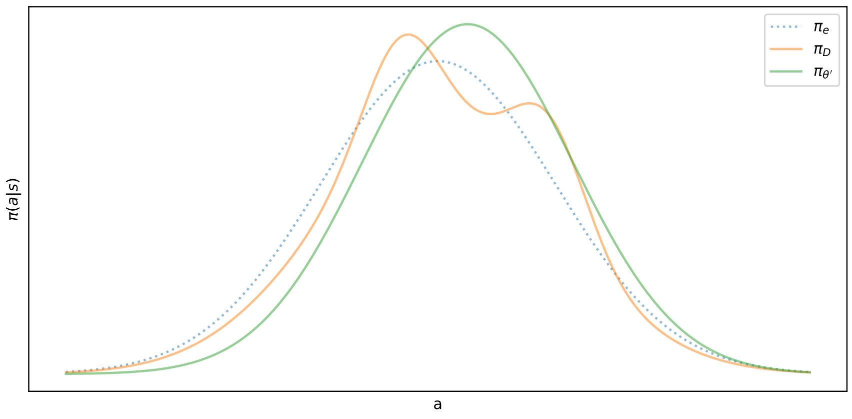


Figure 3.2: Distribution of evaluation policy π_e, empirical policy π_D and ROS behavior policy $\pi_{\theta'}$.

To mitigate this problem, we propose another algorithm, called Robust On-policy Acting (ROA), shown in Algorithm 2. Instead of finding and sampling from the ideal behavior policy to correct sampling error, this algorithm aims to identify and choose the action that can minimize sampling error, which is where the word "Acting" comes from. As we know if $\pi_D = \pi_e$ (no sampling error), we will have $\nabla_\theta \text{Loss} = 0$. Therefore, in order to reduce sampling error, we propose to choose the action that can lead to the minimum of the L1-norm of the gradient, as $a^* = \text{argmin}_{a \in \mathcal{A}} ||\nabla_\theta \text{Loss}||$. We refer to this action as the ideal action and choosing this action as the correction operation. As the gradient generated by historical data is also unchanged, this algorithm follows the same way as ROS to update and store the average gradient.

However, there is one problem in this algorithm for domains with continuous state space where ϕ is a neural network. Theoretically, if the algorithm tries to reduce sampling error in a certain state, only those historical data in similar states should be considered. However, when ϕ is a neural network, the gradient generated by all historical data will be stored and merged together. Therefore, suppose that we have not col-

Algorithm 2 Robust On-policy Acting.

```
Input: evaluation policy $\pi_e$, correction probability $\varepsilon$, potential action number $m$
Initialize $\nabla \leftarrow 0$
Initialize $\pi_\theta = \pi_e$
for step $i$ do
    Observe $s$ from the environment
    Sample $u$ from $\mathcal{U}(0,1)$
    if $u < \varepsilon$ then
        if $\mathcal{A}$ is a finite set then
            $\widetilde{\mathcal{A}} \leftarrow \mathcal{A}$
        else
            $\widetilde{\mathcal{A}} \leftarrow \{F^{-1}(\frac{i}{m+1}|\pi_e(s))\}_{i=1}^{m}$ where $F^{-1}$ is the inverse CDF.
        end if
        Choose $a = \text{argmin}_{a' \in \widetilde{\mathcal{A}}} || \frac{i}{i+1}\nabla + \frac{1}{i+1}\nabla_\theta \log \pi_\theta(a'|s) ||$
    else
        Choose $a \sim \pi_\theta(s)$
    end if
    $\nabla \leftarrow \frac{i}{i+1}\nabla + \frac{1}{i+1}\nabla_\theta \log \pi_\theta(a|s)$
end for
```

lected any or only collected a small amount of data in a certain state, this algorithm will choose the action that can reduce the gradient mostly generated by other unrelated states, which in fact cannot reduce sampling error. What is worse is that the norm of gradient has already reduced, meaning that some historical sampling error will be erased from the gradient and thus cannot be corrected by this algorithm in future data collection. To reduce the effects from unrelated states, we propose to reserve a probability ε to sample actions from the evaluation policy, by which we can collect enough data for the algorithm to identify the ideal action more precisely.

In addition, for domains with continuous action space, $\mathcal{A}$ is an infinite set, which makes it hard to conduct the “argmin” operation. To make it more efficient, this algorithm only considers an m-action finite subset of the whole action space. This finite subset is obtained by computing the inverse cumulative distribution function (CDF) on the uniform probability. which could consider a wide range of actions.

3.4 Comparison of ROS and ROA

To correct sampling error, the main idea of ROS is to increase the probabilities of actions that is sampled too rarely in historical data collection (called less-often actions). while that of ROS is to identify the action that can minimize sampling error. However, one problem of ROS is that choosing less-often actions cannot guarantee to reduce sampling error.

To explain this, suppose at step t, we have the empirical policy as $\pi_D^t(a|s) = \frac{N_t(s,a)}{N_t(s,\cdot)}$ where $N_t(s,a)$ means the count of state-action pair (s,a) occurring in t-step historical data. If we collect data of (s,a') for the next step $t+1$, the empirical policy will become $\pi_D^{t+1}(a|s) = \frac{N_t(s,a)+\delta_{aa'}}{N_t(s,\cdot)+1}$ where δ is the Kronecker delta. ROS will increase the behavior probability of those less-often action a' such that $\pi_D^t(a'|s) < \pi_e(a'|s)$. However, especially when $N_t(s,\cdot)$ is small, choosing this kind of action may dramatically change the empirical policy, and cannot guarantee reducing sampling error. For example, suppose we have the evaluation probability $\pi_e(a_1|s) = 0.9$ and $\pi_e(a_2|s) = 0.1$. If we have only collected one-step data of action a_1, the empirical policy is $\pi_D^t(a_1|s) = 1$ and $\pi_D^t(a_2|s) = 0$. At this point, action a_2 is a less-often action for state s. However, if we choose action a_2, the empirical policy will become $\pi_D^{t+1}(a_1|s) = 0.5$ and $\pi_D^{t+1}(a_2|s) = 0.5$, which will significantly change the empirical policy and enlarge sampling error. We refer to this phenomenon as *over-correction*. On the other hand, ROA can avoid over-correction, as it first "simulates" choosing actions and then actually choose the action that can minimize the updated gradient, which it thought to be equivalent to minimizing sampling error.

Moreover, as both ROS and ROA use the gradient to store information of historical data, they share the same defect: for policy models that involves neural networks, when choosing an action for a certain state, both of them may consider historical data in other unrelated states, and reduce sampling error in a wrong way, which we refer to as *over-consideration*.

To better understand this, we start from policy models that contain no neural network. We can observe from Equation 3.1 and 3.2 that ϕ is a non-parametric one-hot encoding operator for discrete state space, which makes each parameter of the policy model only responsible for one state. In this case, the gradient generated by data of a certain state will have no effect on the parameter responsible for other states. Therefore, the action decisions in ROS and ROA will not be affected by data from different states. However, for continuous state space, ϕ is a parametric neural network, and the parameters are shared among different states. In this case, it is inevitable that the gradient generated by a certain state could have effects on the action decisions in other states. If these effects happen across totally unrelated states, over-consideration occurs.

3.5 Connection to RIS

Regression Importance Sampling uses maximum likelihood to compute the “actual” behavior policy, aiming to correct sampling error for data that have already been collected, while our robust on-policy strategies are designed to reduce sampling error during data collection. There are two main reasons why it is still necessary to collect low-sampling-error data even with RIS value estimation: 1) RIS uses importance sampling technique to re-weight the returns. If the empirical policy π_D has a large difference from the evaluation policy π_e, the importance ratio could be extremely large or small, which can cause a large variance of the estimation. 2) RIS will be a biased estimation if there exists data that has a positive probability under π_e but have not been collected in historical data. As the empirical probability for this un-tested data is 0, it will always be seen as the less-often data. To reduce sampling error, this data will inevitably be taken into account, and usually have a higher probability of being collected (more discussion see Section 5.3).

3.6 Measurement of Sampling Error

In order to compare sampling error across different data collection strategies, we propose to quantify sampling error using KL divergence of the empirical policy π_D and the evaluation policy π_e as

$$\begin{aligned} \mathrm{KL}\left(\pi_D \parallel \pi_e\right) &= \mathbb{E}\left[\log \pi_D(a|s) - \log \pi_e(a|s) \mid a \sim \pi_D(a|s)\right] \\ &\approx \mathbb{E}\left[\log \pi_D(a|s) - \log \pi_e(a|s) \mid s, a \sim \mathcal{D}\right] \end{aligned} \tag{3.3}$$

In this equation, as π_D getting closer to π_e, we will have $\mathrm{KL}\left(\pi_D \parallel \pi_e\right) \rightarrow 0$, meaning that the data collection suffers from less sampling error.

Chapter 4

Experiments

In this chapter, we will evaluate the performance of Robust On-policy Sampling (ROS) and Robust On-policy Acting (ROA) in reinforcement learning domains with different state and action space (detailed description see Appendix A):

- **Discrete states and discrete actions:** k-armed bandit problem [Sutton and Barto, 2018], 4x4 Gridworld [Hanna et al., 2017].
- **Continuous states and discrete actions:** CartPole [Brockman et al., 2016].
- **Continuous states and continuous actions:** MountainCarContinuous [Brockman et al., 2016].

4.1 Preparation

To conduct policy evaluation, we need to prepare a set of evaluation policies. For domains with discrete action space and domains with continuous action space, we build the policy model as Equation 3.1 and 3.2, respectively. For all domains, we use REINFORCE [Williams, 1992] to train the policy model, and randomly choose a policy snapshot during training as the evaluation policy, which has higher returns than random policies, 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 μ_g and σ_g to denote the mean and standard deviation of discounted returns, and $\sigma_\mu = \frac{\sigma_g}{\sqrt{n}}$ is the estimated standard deviation for μ_g, representing the error bound of the true policy value, as shown in Table 4.1. To compare across different domains, we normalize the policy value to 1 and during the following experiments, all the rewards will be normalized as $\widetilde{R} = \frac{R}{\mu_g}$.

Table 4.1: Information of prepared policies in different domains.

Domain	$\overline{T}$	$\mu_g \pm \sigma_\mu$	$\widetilde{\mu_g} \pm \widetilde{\sigma_\mu}$
MultiBandit	1.00	0.78 ± 4.45e-04	1 ± 5.72e-04
GridWorld	7.18	4.70 ± 3.73e-03	1 ± 7.94e-04
CartPole	147.44	73.41 ± 1.73e-02	1 ± 2.36e-04
MountainCarContinuous	181.74	7.76 ± 9.35e-03	1 ± 1.20e-03

* $\overline{T}$ denotes the average steps of each trajectory. μ_g denotes the mean of discounted returns, σ_μ denotes the standard deviation of μ_g, and $\widetilde{\mu_g}$, $\widetilde{\sigma_\mu}$ denote their normalized values.

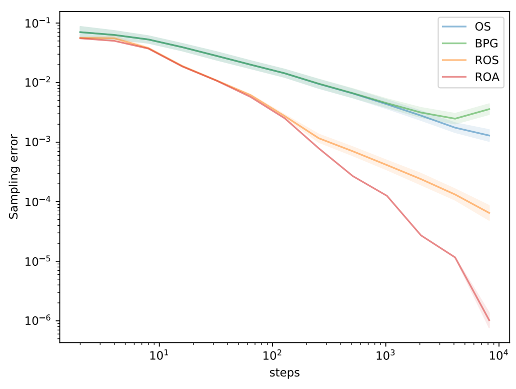
4.2 Experimental Set-up

For each domain, we follow different data collection strategies to collect around 2^{13} trajectories of data, which is equivalent to $2^{13} \times \overline{T}$ steps of data. For every step in $\{2^i \times \overline{T}\}_{i=1}^{13}$, we use the current collected data to perform evaluation, including value estimation and sampling error evaluation. Each experiment is conducted with 200 different seeds, and the metrics are averaged over these seeds. In this chapter, we choose step size $\alpha = 1000$ for ROS and correction probability $\varepsilon = 1.0$ for ROA in MultiBandit and GridWorld, $\alpha = 10$ for ROS and $\varepsilon = 0.05$ for ROA in CartPole, $\alpha = 0.1$ for ROS and $\varepsilon = 0.05, m = 15$ for ROA in MountainCarContinuous. More discussion about hyper-parameters will be presented in Chapter 5. In addition, we take On-policy Sampling (OS) and Behavior Policy Gradient (BPG) as the baselines in this project. Note that we have experimented with adapting Upper Confidence Bound (UCB) for policy evaluation [Sutton and Barto, 2018]. However, as there is no consideration of the evaluation policy π_e in UCB, it may end up with data collection with extremely low probability under π_e. In our experiments, we found that this data can make little contributions to policy evaluation, and thus omitted it from our reported results.

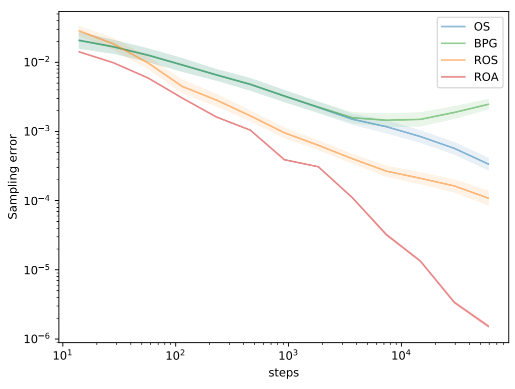
4.3 Main Results

In this section, we will show and compare the performances of ROS, ROA and other baseline data collection strategies.

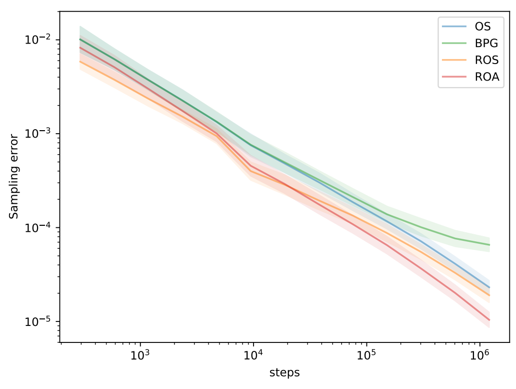
Figure 4.1 compares sampling error (KL-divergence) of different data collection strategies. **Hypothesis 1** can be verified that our proposed strategies (ROS and ROA)



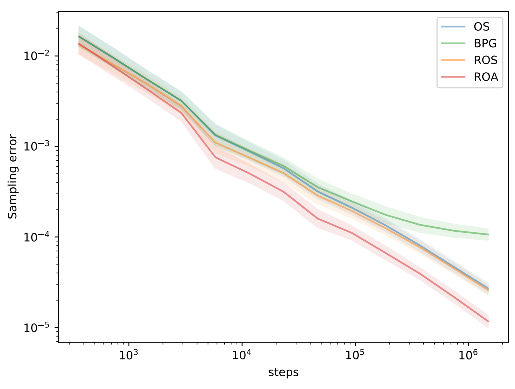
(a) MultiBandit



(b) GridWorld



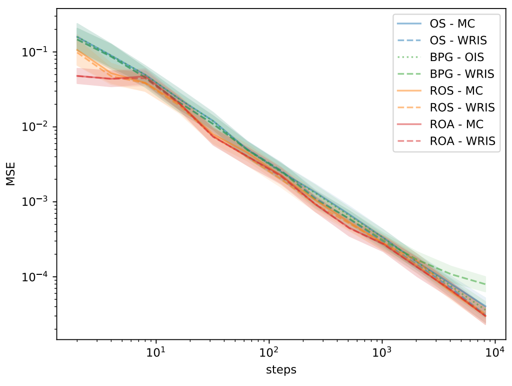
(c) CartPole



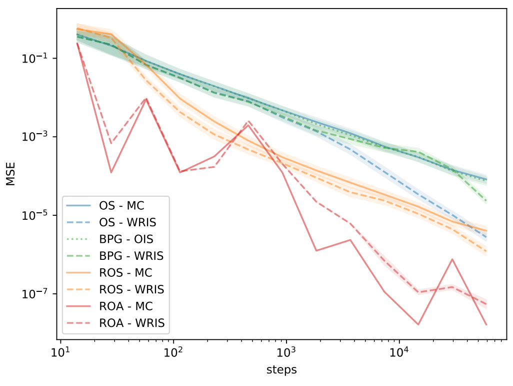
(d) MountainCarContinuous

Figure 4.1: Sampling Error of Different Data Collected Strategies

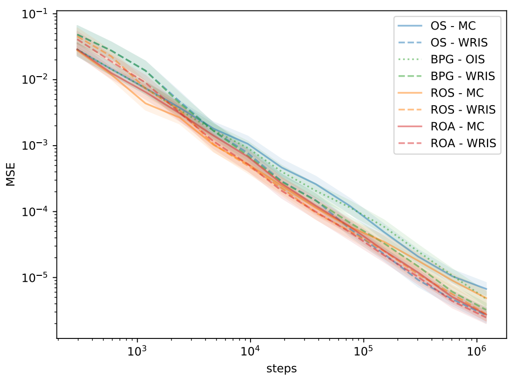
can generally reduce sampling error faster than OS and BPG. It is worth noting that ROS can lead to larger sampling error with small data collection in GridWorld (Figure 4.1b). That is because as discussed in Section 3.4, ROS tends to choose less-often data, while this data cannot guarantee to reduce sampling error, especially with small data. Another interesting finding is that in CartPole (Figure 4.1c), ROS can enable lower sampling error than ROA with small data, but gradually have the sampling error close to data from OS. That is because this result is from ROS with large step size ($\alpha = 10$), meaning that it can have a large probability to choose less-often actions with small data. On the other hand, the correction probability ε for ROA is only 0.05, meaning that it only spends few steps to make correction operation. However, as it collects more data, the average gradient decreases significantly, which makes the ROS behavior policy very close to the evaluation policy (the behavior policy of OS) (more discussion see Section 5.4). In addition, we can also notice that BPG has similar sampling error compared to OS, which is because BPG has a similar behavior policy



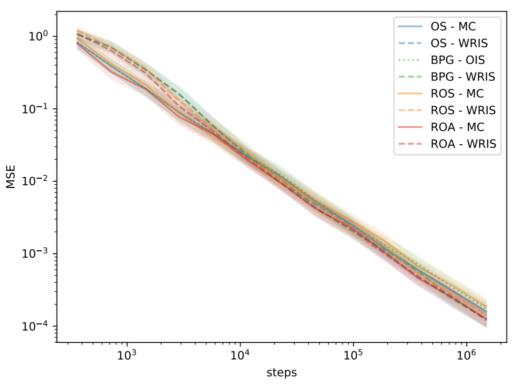
(a) MultiBandit



(b) GridWorld



(c) CartPole



(d) MountainCarContinuous

Figure 4.2: Mean Squared Error of Different Policy Evaluations

as OS during most of the data collection, as every update of BPG requires a batch of trajectories and a small step size to perform reliable gradient descent. As BPG collects more data, its behavior policy will be more different from the evaluation policy (i.e., BPG becomes more "off-policy"), and its sampling error will stop decreasing and even start increasing.

To evaluate the performance of these data collection strategies, we first perform trajectory-based value estimation on them: MC, WRIS on data from on-policy strategies (OS, ROS, ROA) and OIS, WRIS on data from the off-policy strategy (BPG). Figure 4.2 shows the curves of mean squared error between these estimations and the true policy value. From these figures, it can be observed that "BPG - OIS" has a very close MSE as "OS - MC", which is because BPG requires large data and makes small updates. On the other hand, when using the same value estimation methods, ROS and ROA can generally lead to lower MSE than OS in domains with discrete action space (Figure 4.2a, 4.2b and 4.2c), while only ROA can lower the MSE of OS in the domain

Table 4.2: The final MSE of trajectory-based estimations with different data collection strategies.

Strategy	Estimation	MultiBandit	GridWorld	CartPole	MountainCarCon
OS	MC	4.04e-05	8.23e-05	6.72e-06	1.59e-04
OS	WRIS	3.08e-05	2.78e-06	2.68e-06	1.25e-04
BPG	OIS	3.61e-05	7.46e-05	4.74e-06	1.74e-04
BPG	WRIS	7.96e-05	2.35e-05	3.23e-06	1.24e-04
ROS	MC	3.02e-05	4.04e-06	4.87e-06	1.87e-04
ROS	WRIS	3.18e-05	1.21e-06	2.79e-06	1.44e-04
ROA	MC	3.00e-05	**1.64e-08**	2.75e-06	1.26e-04
ROA	WRIS	**2.99e-05**	5.45e-08	**2.48e-06**	**1.20e-04**

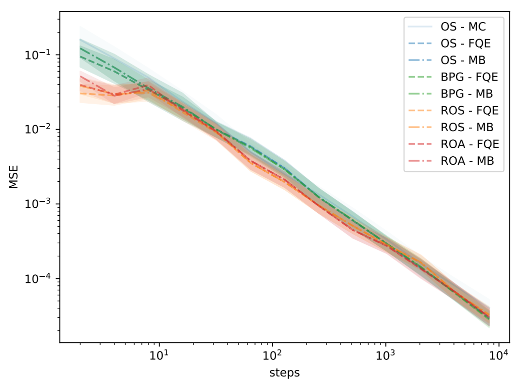
with continuous action space (Figure 4.2d). Therefore, for **Hypothesis 2**, we suggest that data with lower sampling error can generally enable lower MSE, but no guarantee can be made. In addition, WRIS can generally lower MSE from these on-policy strategies with large data, but not necessarily for small data in non-tabular domains. That is because WRIS could re-weight data according to the difference between π_D and π_e and correct sampling error. For small data collection, π_D could be very different from π_e, and thus could bring a large variance of WRIS. The MSE of estimations on the final data collection is reported in Table 4.2, and it is shown that ROA is the strategy that can produce the lowest MSE of trajectory-based estimations in these domains.

To further explore how ROS and ROA perform in different RL domains, we will discuss each experiment more thoroughly. In Multi-armed Bandit (Figure 4.2a), obvious improvement of ROS and ROA can be seen with small data collection, which is brought by correcting sampling error. In particular, ROA can enable lower MSE than ROS, because ROA with $\varepsilon = 1$ can identify and choose the ideal action for each step, while ROS can only increase the probability of less-often data and still requires sampling. As they collect more data, the performance gap between ROS, ROA and OS is getting closer, which is because sampling error can be easily reduced even with OS in this one-state domain. Moreover, it is worth noting that with small data collection, 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.

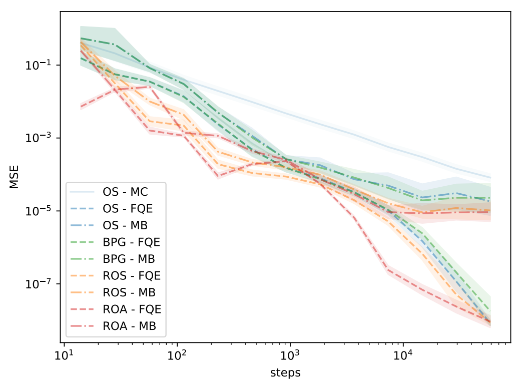
In GridWorld (Figure 4.2b), at the beginning of the data collection, estimations on data from ROS has slightly higher MSE than that from OS, which is perfectly consistent with the sampling error curve in Figure 4.1b. That is because ROS tends to choose the un-tested data, which may have low probability and thus can only make a very small contribution to accurate policy evaluation with small data. However, as it collects more data, the benefit of reducing sampling error is shown with ROS showing large improvement over OS. On the other hand, for ROA, the error range of MC estimation is interestingly 0, because GridWorld is a pure deterministic domain, and this result is from ROA with $\varepsilon = 1$, a deterministic strategy. Although the MSE of estimation on data from ROA fluctuates as data collection, it can still be observed that this MSE is decreasing and is generally lower than the MSE from all other strategies. Moreover, WRIS can significantly reduce the MSE of OS data but only makes little difference for the MSE of ROS and ROA data, showing ROS and ROA data have less sampling error to be corrected. Although WRIS can reduce MSE of OS data, there is still a gap to the performances of ROS and ROA with WRIS.

In CartPole (Figure 4.2c) and MountainCarContinuous (Figure 4.2d), we can first notice that compared with MC, WRIS can even enlarge MSE with small data collection, showing that WRIS may require larger data to work in non-tabular domains. In CartPole, it is noteworthy that the MSE of “ROS - MC” is getting closer to “OS - MC”, which is also very similar to the sampling error curves in Figure 4.1c, and has the same reason that the ROS behavior policy is getting closer to the evaluation policy π_e (more discussion see Section 5.4). On the other hand, ROA will not be affected by that because it has a fixed correction probability, and thus can maintain the performance improvement as collecting more data. In MountainCarContinuous, although ROS can collect data with lower sampling error in Figure 4.1d, it cannot produce lower MSE of policy evaluation, showing that the MSE curve is not always consistent with the sampling error curve. Moreover, ROA is the only one that can enable lower MSE than OS among these strategies.

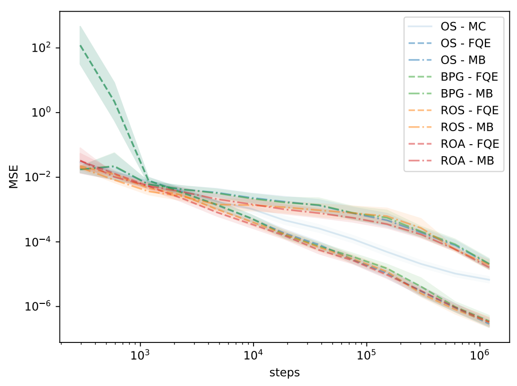
To sum up, although ROS can generally enable lower MSE than OS, it has the following limitations: 1) with small data, it may increase sampling error; 2) with large data, it may end up with the behavior policy very close to the evaluation policy, making it inefficient to further reduce sampling error; 3) it could have higher MSE in domains with continuous action space. On the other hand, when using MC as value estimation, ROA can always lead to lower MSE than OS, and there are no sign that this MSE gap between will get smaller if continuing data collection. Moreover, WRIS can hardly produce a more accurate estimation for ROA data, as this data has low sampling error.



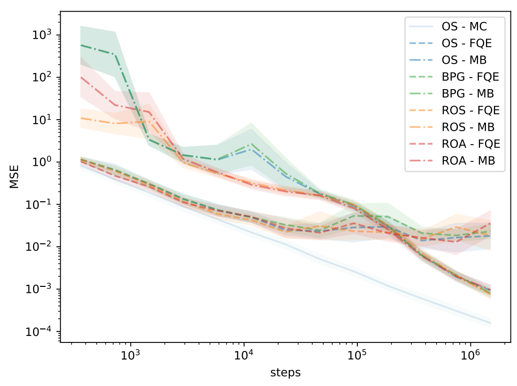
(a) MultiBandit



(b) GridWorld



(c) CartPole



(d) MountainCarContinuous

Figure 4.3: Mean Squared Error of Different Policy Evaluations

4.4 Performance with FQE and MB

In this section, we will evaluate the performance of these data collection strategies with the transition-based value estimations (FQE, MB). Value estimation is performed on the same data collection as Section 4.3, and the MSE curves are shown in Figure 4.3.

As discussed before, the behavior policy of BPG only experiences slight updates from the evaluation policy, and therefore, we can observe a very close performance between BPG and OS with FQE and MB, within small data collection. As they collect more data, BPG can generally enable larger MSE of these transition-based estimations than OS, showing that this slightly off-policy data cannot help improve the accuracy of estimation. On the other hand, our proposed robust on-policy strategies (ROS and ROA) can also generally lower MSE than OS with these estimation methods.

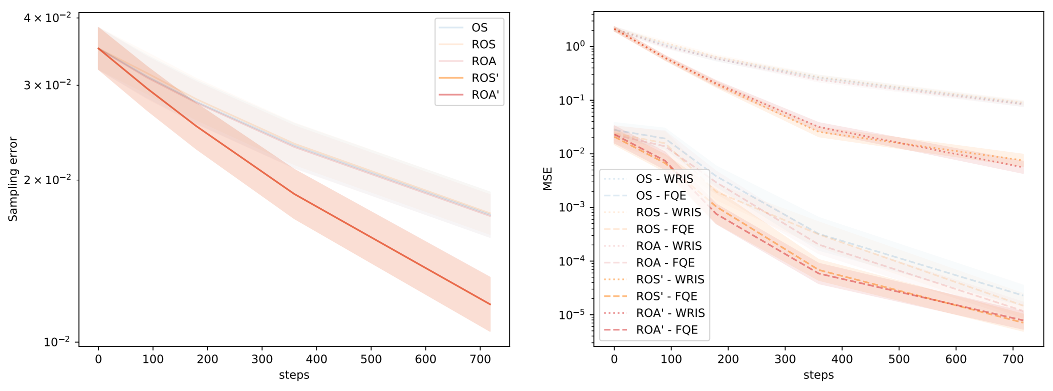
To further compare their performances in different RL domains, we will discuss each experiment more thoroughly. In pure tabular domains (Figure 4.3a and 4.3b), these transition-based estimations can generally enable lower MSE than the classical trajectory-based estimation (MC and OIS). That is because the accuracy of transition-based estimations heavily relies on the accuracy of the bootstrapping data, which can be improved by collecting more different transition data (state-action pairs). Since ROS and ROA will choose less-often data to reduce sampling error, they can usually collect data with more unique transitions (more discussion see Section 5.3), and thus lead to lower MSE of these estimations. As these strategies collect all transitions, they will end up with very similar estimations, shown as the close MSE gap between them at the end of data collection. Surprisingly, there is still a small gap of MB estimations among ROS, ROA and OS in GridWorld, even though all of these strategies have collected all transitions (shown in Figure 5.3b). That may be because the estimated transition model is under-fitting, and tries to fit the data with higher frequency in the data collection. In this case, low sampling error could still be important even for large data.

In CartPole (Figure 4.3c), the most striking observation is that "OS - FQE" has extremely large MSE compared to ROS and ROA with FQE. That is because with very small data, the Q-value model could be easily over-fitting. Suppose OS happens to collect a trajectory with low probability under π_e but with large returns. The bootstrapping data will have relatively large rewards, and since the Q-value model is updated recursively, it could cause large MSE of estimation. However, ROS and ROA could lower the probability of this accident, as this low-probability trajectory can cause large sampling error for small data collection. Another finding worth noting is that MB fails to make more accurate estimation than classical trajectory-based estimations. That is because this domain involves continuous state space, and will require large data of different states for the estimated transition model to generalize well.

In MountainCarContinuous (Figure 4.3d), both FQE and MB have larger MSE than classical trajectory-based estimations. That is because this domain involves continuous state and action space, and will also require large data of different states and actions to train the estimation model. In addition, an obvious performance gap between ROS, ROA and OS with MB can be observed, again showing the importance of low-sampling-error data for estimation model training.

4.5 Extension

In this section, we design experiments to simulate a certain scenario where there already exists some off-policy data, and evaluate how much our proposed strategies can



(a) Sampling error (b) MSE

Figure 4.4: Performance of continuing data collection in GridWorld

improve the estimation by collecting more data.

By taking GridWorld as an examples, we first prepare around 100 off-policy trajectories ($100 \times \overline{T}$ steps) by following the behavior policy 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, or otherwise it will sample from the evaluation policy π_e. To make this prepared data collection more random, we sample δ from the uniform distribution $\mathcal{U}(0,1)$ for each trajectory.

After that, we then use OS, ROS, ROA respectively to continue collecting also around 100 trajectories, and combine this new data collection with the off-policy data prepared above. In particular, we denote ROS and ROA as the version that only considers their newly collected data as historical data, while ROS$'$ and ROA$'$ as the version that could also consider the off-policy data as historical data. Therefore, the aim of OS, ROS and ROA is to make the distribution of newly collected data close to the evaluation distribution, while that of ROS$'$ and ROA$'$ is to make the whole distribution close to the evaluation distribution. We can also observe this difference from Figure 4.4a that ROS$'$ and ROA$'$ (overlap) could reduce sampling error of whole data collection much faster than others.

We then perform one trajectory-based estimation (WRIS) and one transition-based estimation (FQE) on these merged data collections, shown in Figure 4.4b. It is shown that without considering the prepared off-policy data, ROS and ROA will lead to very simialr performance as OS with WRIS, and only slight improvement to OS with FQE. However, with the prepared off-policy data into consideration, both ROS and ROA can enable much lower MSE than OS with WRIS or FQE. This result could show the rationality of how ROS and ROA make use of the historical data.

Chapter 5

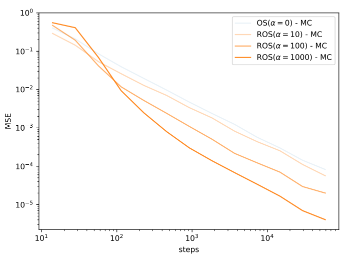
Analysis

In this chapter, we will explore the properties of ROS and ROA both empirically and theoretically.

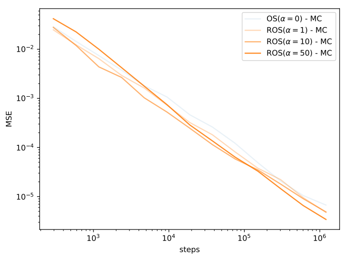
5.1 Effect of Hyper-parameter in ROS

From Algorithm 1, ROS only involves one hyper-parameter, step size α, which controls how much the ROS behavior policy is updated in the "opposite" direction of the empirical policy. Therefore, we can say ROS with larger α will have higher probability to sample less-often data. In Figures 5.1, we show the MSE curves of ROS with different step size α. In particular, OS can be seen as ROS with $\alpha = 0$.

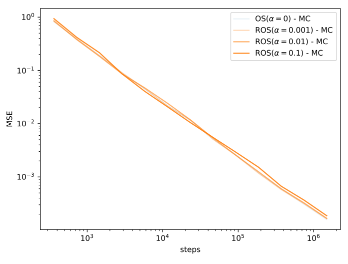
A similar trend can be observed in domains with discrete actions (Figure 5.1a and 5.1b): ROS with small α can enable lower MSE with small data, while ROS with large α leads to lower MSE with large data. As we discussed in Section 3.4, choosing less-often data, especially with small data, may enlarge sampling error (over-correction). As ROS with larger α could have higher probability to choose this less-



(a) GridWorld



(b) CartPole



(c) MountainCarContinuous

Figure 5.1: ROS with different α

often data, it could cause over-correction more easily with small data. However, the problem of over-correction could disappear, as it collects more data, because the empirical policy will not be affected dramatically by one single step of data. Therefore, with large data, ROS with larger α could have higher probability to reduce sampling error, and lead to a more accurate estimation. Based on this result, we believe a changing α during data collection can make a better ROS version. Unfortunately, due to the time limit, we have not further explored this in this project.

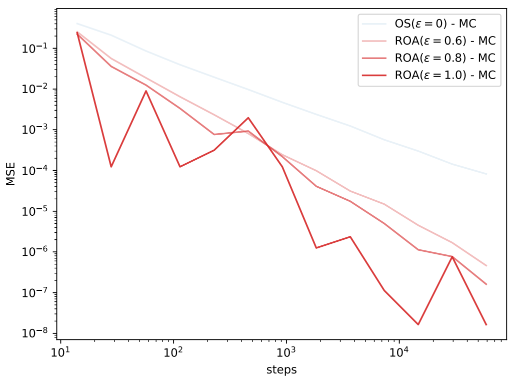
On the other hand, in domains with continuous action space, as the ROS behavior policy could only approximate the same distribution as the evaluation policy π_e (normal distribution), it cannot guarantee to be an ideal behavior policy. From Figure 5.1c, we can observe that when using small α (0.001 or 0.01), ROS has very similar performance as OS, as the ROS behavior policy is only changed very slightly. As α is increased to 0.1, ROS will generally lead to larger MSE. Furthermore, results of ROS with $\alpha > 0.1$ cannot be shown because when ROS behavior policy is changed too much from π_e, it could easily sample actions with probability density function close to 0 under π_e. In this case, the gradient $\nabla_\theta \log \pi_e(a|s)$ will be exploding, and the ROS behavior policy will diverge.

5.2 Effect of Hyper-parameters in ROA

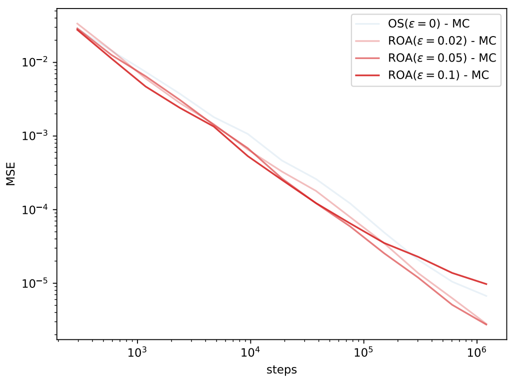
From Algorithm 2, ROA involves two hyper-parameter, correction probability ε and potential action number m. In Figures 5.2, we show the MSE curves of ROA with different ε and m. In particular, OS can be seen as ROS with $\varepsilon = 0$.

From these figures, we can notice that ROA with larger ε can generally enable lower MSE in the domain with discrete state space (Figure 5.2a), while this pattern cannot hold for domains with continuous state space (Figure 5.2b and 5.2c). As discussed in Section 3.4, in domains with discrete state space, the ROA action decisions are only affected by historical data from the same state. Therefore, its correction operation can always reduce sampling error. On the other hand, in domains with continuous state space, the ROA correction operation may over-consider data from other unrelated states, and may reduce sampling error in the wrong way. In this case, ROA is expected to follow a smaller ε, which will spend more steps to collect enough data for each state, and help to make a more precise correction operation. Therefore, the setting of ε for ROA is a trade-off between precision and frequency of correction operations.

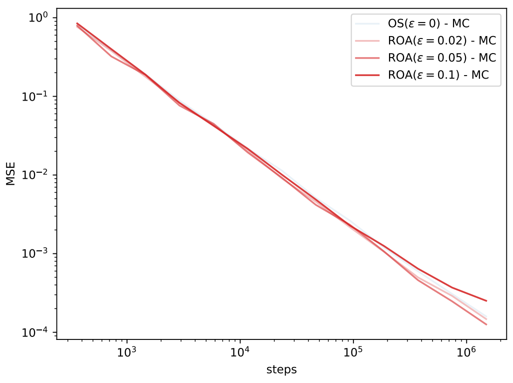
On the other hand, potential action number m is a hyper-parameter that only matters



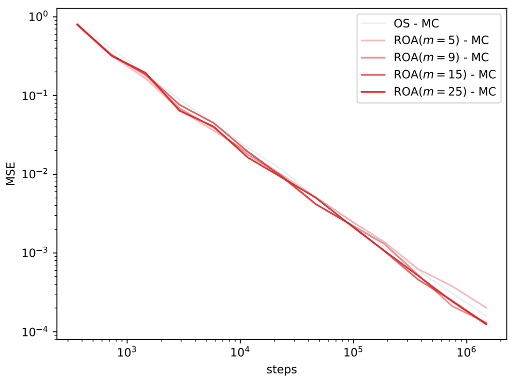
(a) GridWorld



(b) CartPole



(c) MountainCarContinuous



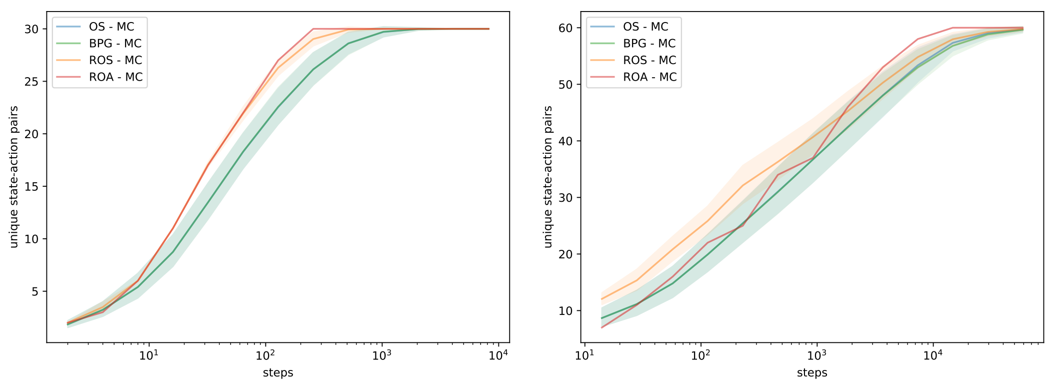
(d) MountainCarContinuous

Figure 5.2: ROA with different ε and m

in domains with continuous action space. In Figure 5.2d, we show the MSE curves of ROS with $\varepsilon = 0.05$ and $m = 5, 9, 15, 25$. It can be observed that with large data, ROS with $m = 5$ cannot enable lower MSE. That may be because ROS can only choose action from these 5 potential actions, which may not contain any action that can reduce sampling error. As m increases, ROA can enable lower MSE, but we can notice that ROA with $m = 15$ has a similar performance as ROA with $m = 25$. However, ROA with $m = 25$ could have a higher computation cost, as it needs to compute the gradient of 25 actions for each correction operation. Therefore, the setting of potential action number m is a trade-off between precision and computation cost.

5.3 Richness of State-action Pairs

In Figure 5.3, we show the count of unique state-action pairs along data collection. It can be observed that ROS tends to collect more unique state-action pairs than the



(a) MultiBandit (b) GridWorld

Figure 5.3: Unique State-action pairs of Different Data Collected Strategies

baseline strategies (OS and OIS) before they reach the maximum number. The reason is that ROS is performing gradient descent every step to reduce the behavior probabilities of collected data, and thus increase the probabilities of un-tested data. On the other hand, ROA tends to collect fewer state-action pairs with a small amount of data, which is because this un-tested data cannot guarantee to reduce sampling error, especially with small data, as discussed in Section 3.4. However, it can be noticed that ROA is the first strategy that can reach the maximum number of unique pairs. That is because although ROS can increase the probability of un-tested data, this data may still not be sampled, while ROA with $\varepsilon = 1$ can choose this un-tested data directly. Therefore, ROA is the most efficient strategy among them to collect all state-action pairs.

5.4 Consistency

This section will discuss whether ROS and ROA can lead to the consistent estimation, i.e., whether the estimation will converge to the true policy value, as we collect more and more data. To explore this, we show the average absolute gradient of ROS along data collection in Figure 5.4. From Algorithm 1, we can see that the ROS behavior policy $\pi_{\theta'}$ will get closer to π_e as the gradient $|\nabla|$ decreases. Moreover, once the empirical policy π_D is equal to π_e, we will have $\nabla_\theta \mathbb{E}\left[\log \pi_\theta(a|s) \mid a \sim \pi_D(a|s)\right] = 0$, and the behavior policy will be the same as the evaluation policy. Therefore, we suggest ROS is a consistent data collection strategy.

On the other hand, ROA can only guarantee to be a consistent data collection strategy for tabular domains, while its consistency relies on the settings of hyper-parameters

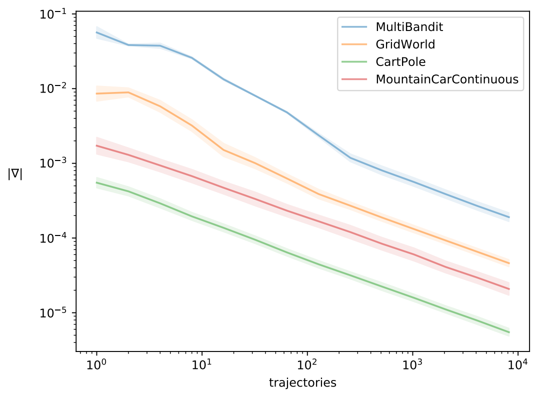


Figure 5.4: Average Absolute Gradient of ROS in different domains.

ε and m in non-tabular domains. That is because in domains with continuous state space, the gradient generated by data in a certain state s may be offset by data from other unrelated states. In this case, some sampling error in s may be missing and some actions may never be collected if $\varepsilon = 1$. In domains with continuous action space, as the potential subset of action space is limited, there may be some actions that cannot be contained and collected. If either of these above happens, ROA cannot guaranetee to collect all possible data as it collects more data, and thus will be a inconsistent data collection strategy.

Chapter 6

Conclusion

In this work, we have discussed the problem of sampling error that On-policy Sampling (OS) suffers from, which can lower its accuracy of policy estimation. To solve this problem, we propose two alternative data collection strategies, Robust On-policy Sampling (ROS) and Robust On-policy Acting (ROA), which aim to consider the historical data and reduce sampling error in future data collection. Experimental results show that our proposed strategies can generally collect data with lower sampling error. We also find that limited to the same amount, this low-sampling-error data can generally enable lower mean squared error (MSE) for trajectory-based estimations, and can reduce the probability of extreme estimation of transition-based estimations. In particular, ROA can generally lead to lower MSE than ROS, as ROA can choose the ideal action directly, while ROS still requires sampling. Besides, ROS fails to generalize to domains with continuous action space due to its inflexible output distribution, while ROA can solve this problem. Moreover, we have explored two important properties of our proposed strategies: 1) data richness: both ROS and ROA can collect more unique state-action pairs quicker than OS; 2) consistency: ROS can guarantee to be a consistent data collection strategy, while ROA cannot.

6.1 Unsolved Problems

Although we have explored the effects of hyper-parameters, we have not provided reliable guidance for hyper-parameter settings for different domains and different policy models, which could damage the performance of our proposed strategies when applying them to real applications. In particular, although empirical results show that step size α for ROS should be increased as data collection, we fail to design a dynamic scheme for it due to the time limit. Moreover, when the policy model involves a neural network, both ROS and ROA cannot avoid considering historical data from unrelated states and thus may reduce sampling error in the wrong way. However, this effect has not been well studied and solved in this project.

6.2 Future Work

In this work, we have shown that the gradient generated by collected data could contain information of historical data. With this information, we may be able to design data collection strategies with other aims, e.g., uncertain-aware exploration strategy, which aims to collect data that we have less knowledge about, and has been proved to be data-efficient in many RL training applications [O'Donoghue et al., 2017, Osband et al., 2016]. Therefore, in the future, we believe it worth studying how we can use this gradient to identify data with more uncertainty.

Bibliography

M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286*, 2021.

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`.

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`.

Josiah P. Hanna, Philip S. Thomas, Peter Stone, and Scott Niekum. Data-efficient policy evaluation through behavior policy search. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1394–1403, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/hanna17a.html`.

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

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Brendan O'Donoghue, Ian Osband, Rémi Munos, and Volodymyr Mnih. The uncertainty bellman equation and exploration. *CoRR*, abs/1709.05380, 2017. URL `http://arxiv.org/abs/1709.05380`.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. *CoRR*, abs/1602.04621, 2016. URL `http://arxiv.org/abs/1602.04621`.

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.

Cosmin Paduraru. Planning with approximate and learned models of markov decision processes. *These de maıtre, University of Alberta*, 2007.

Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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.

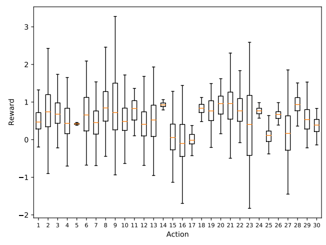
Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Michael R Zhang, Thomas Paine, Ofir Nachum, Cosmin Paduraru, George Tucker, ziyu wang, and Mohammad Norouzi. Autoregressive dynamics models for offline policy evaluation and optimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=kmqjgSNXby`.

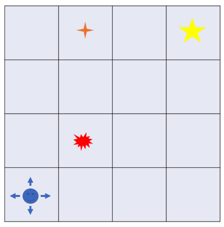
Appendix A

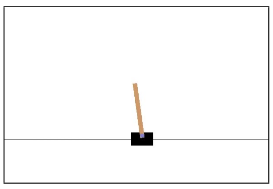
Experiment Domains

In this project, we evaluate data collection strategies mainly in 4 domains: Multi-armed Bandit, Grid-world, CartPole and MountainCarContinuous.



(a) MultiBandit


(b) GridWorld


(c) CartPole

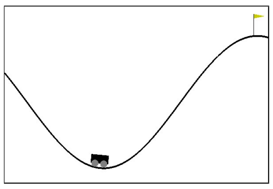

(d) MountainCarContinuous

Figure A.1: Domains

Multi-armed Bandit problem 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 a Gaussian distribution with different means and variance, as shown in Figure A.1a.

Gridworld is a domain with 4×4 states, shown in Figure A.1b. The agent starts from $(0,0)$ and has action spaces $\{$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.

CartPole 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 A.1c. The agent will get $+1$ reward every timestep, and terminates once the pole is more than 15 degrees from vertical or the cart move out of the picture. The maximum number of steps is 200.

MountainCarContinuous is another continuous control domain where the agent aims to drive the car to the top of the right mountain, as shown in Figure A.1d. The agent could spend energy to drive the car back and forth, and will get the energy cost as a reward ($r \in [-0.1, 0]$) every timestep. Once the car reaches the goal, it will get $+100$ as a reward and terminates. The maximum number of steps is 200.