Towards Provable Log Density Policy Gradient

Pulkit Katdare
University of Illinois at Urbana-Champaign

Anant A. Joshi
University of Illinois at Urbana-Champaign

Katherine Driggs-Campbell
University of Illinois at Urbana-Champaign

Abstract

Policy gradient methods are a vital ingredient behind the success of modern reinforcement learning. Modern policy gradient methods, although successful, introduce a residual error in gradient estimation. In this work, we argue that this residual term is significant and correcting for it could potentially improve sample-complexity of reinforcement learning methods. To that end, we propose log density gradient to estimate the policy gradient, which corrects for this residual error term. Log density gradient method computes policy gradient by utilising the state-action discounted distributional formulation. We first present the equations needed to exactly find the log density gradient for a tabular Markov Decision Processes (MDPs). For more complex environments, we propose a temporal difference (TD) method that approximates log density gradient by utilizing backward on-policy samples. Since backward sampling from a Markov chain is highly restrictive we also propose a min-max optimization that can approximate log density gradient using just on-policy samples. We also prove uniqueness, and convergence under linear function approximation, for this min-max optimization. Finally, we show that the sample complexity of our min-max optimization to be of the order of $m^{-1/2}$, where m is the number of on-policy samples. We also demonstrate a proof-of-concept for our log density gradient method on gridworld environment, and observe that our method is able to improve upon the classical policy gradient method by a clear margin, thus indicating a promising novel direction to develop reinforcement learning algorithms that require fewer samples.

1 Introduction

Policy gradient (PG) methods are a vital ingredient behind the success of modern reinforcement learning (Silver et al., 2017; John Schulman et al., 2023; Haarnoja et al.; Kakade, 2001). The success of PG methods stems from their simplicity and compatibility with neural network-based function approximations (Sutton et al., 1999; Baxter & Bartlett, 2001). Although modern policy gradient methods like PPO and TRPO, have achieved excellent results in various on-policy tasks (Schulman et al., 2017; 2015), they require extensive hyper-parameter tuning. Additionally, it has been shown by Ilyas et al. (2020) that the estimation error between policy gradient estimated by the methods like PPO and the true policy gradient increases significantly during the training process. Classical policy gradient methods typically approximate gradient of the policy using Q-function estimated with discount factor strictly less than 1, which leads to a error in gradient estimation (Morimura et al., 2010). In this paper, we empirically demonstrate that this error in indeed significant, in Figure 1. We further propose a novel algorithm to estimate policy gradient that corrects for this residual error, which could potentially lead to sample efficient reinforcement learning, thus enabling their deployment over a wide variety of complex scenarios. We call our method, log density gradient. We show that log density gradient method can be used to estimate the policy gradient for all values of discounting factor – including the *average reward scenario*. Log density gradient method is based on the average state-action

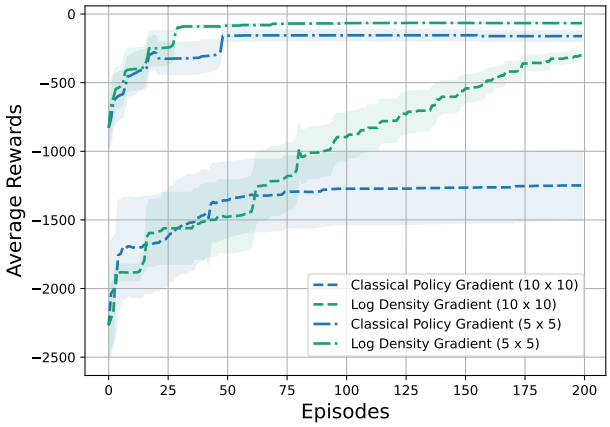

Figure 1: For the average reward scenario, performance of classical policy gradient (blue) algorithm as compared to log density gradient (green) algorithm over a $n \times n$ gridworld environment, for $n = 5, 10$. We observe that log density gradient algorithm consistently converges to better policy performance. Theoretical calculated solutions are used for implementation.

stationary distribution formulation of reinforcement learning, which allows for the estimation of policy gradient as a multiplication of the gradient of log density and the reward function (Nachum et al., 2019; Uehara et al.). This separation results in an improved correlation with the true policy gradient and requires fewer hyperparameters. We show that our method is consistent with the classical policy gradient theorem (Sutton, 1988) and also prove convergence properties and sample complexity.

Our main contributions are as follows. **1.** A novel method to *provably* calculate policy gradient by using the average state-action discounted formulation for all values of the discounting factor. We will show that policy gradient estimated in this manner for average reward scenario will correct for the residual error in policy gradient estimation, which is widely ignored in empirical implementations of policy gradients (as shown in Figure 1). **2.** A model-free Temporal Difference (TD) method for approximating policy gradient. We provide proof of contraction as well as convergence. However, there is a major drawback that it requires samples from the backward Markov chain (described in detail in the paper) which motivates the next contribution. **3.** A min-max optimization which yields the gradient of log density for all values of the discounting factor including the average reward scenario and a model free TD method to implement it, with proof of convergence. We also show that this min-max optimization has a closed form solution under linear function class assumptions, thus enabling their practical use with linear MDP problems (Zhang et al., 2022). We additionally show sample complexity of the order $O(m^{-1/2})$ for the projected version of the proposed TD method, where m is the number of on-policy samples. Our method is competitive with the sample complexity of classical vanilla policy gradient methods (Yuan et al., 2022).

Section 2 starts with problem formulation and motivation behind this paper. Section 3, discusses prior work in policy gradient methods, temporal difference methods, and min-max problems in off-policy evaluation and compares our work with existing works to situate our paper in the literature. Our main contributions are discussed in detail starting from Section 4 which starts with rigorously defining log density gradient. Additionally we also propose a TD approach to estimate log density gradient under strict reversibility assumptions, and we describe the issue caused by this assumption. In section 5, to overcome this issue to propose a min-max variant that allows us to estimate log density gradient algorithm using empirical samples. We finally demonstrate a proof-of-concept of our algorithm in Section 6 which shows that log density gradient can be potentially sample efficient as compared to classical policy gradient methods.

2 Background and Motivation

Notation: we let $(\cdot)^T$ denote matrix transpose, and let e represent the vector of ones, the size of which would be clear from context.

We define Markov Decision Process (MDP) as a 6-tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, d_0)$. Here, $\mathcal{S}$ is a finite state space of the MDP, $\mathcal{A}$ is a finite action space, $\mathcal{P}$ is the transition probability matrix, r is the reward function and d_0 is the initial distribution. The reinforcement learning problems is optimise for a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes $J_\gamma(\pi)$, defined as

$$J_\gamma(\pi) := (1-\gamma)\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim d_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)], \text{ for } \gamma \in [0, 1)$$

$$J_1(\pi) := \lim_{\mathcal{T}\to\infty} \mathbb{E}[\frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}} \gamma^t r(s_t, a_t), s_0 \sim d_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)], \text{ for } \gamma = 1,$$

where $\gamma \in [0, 1]$ is the discounting factor which accounts for the impact of future rewards in present decision making. When $\gamma = 1$, $J_1(\pi)$ the scenario is called the average reward formulation. Most practical problems in reinforcement learning typically aim to solve for an optimal policy $\pi^* = \arg\max_\pi J_1(\pi)$(See Figure 1 Haarnoja et al.). Modern reinforcement learning methods aim to parameterise policy with a set of parameters $\theta \in \mathbb{R}^n$, where n is the dimensions of the parameter space. We refer to such paremterisation as π_θ. This kind of parameterisation enables us search for optimal set of parameters θ^* instead of a search over $\mathcal{S} \times \mathcal{A}$ which in practice could be very large. We define

$$\theta^* := \arg\max_{\theta \in \mathbb{R}^n} J_1(\pi_\theta).$$

The Q-function $Q_\gamma^{\pi_\theta}$ is commonly used function used to describe the performance of an RL agent. Q-function calculates the long term (discounted) rewards accumulated by an agent following a fixed policy π_θ while starting from a state $s \in \mathcal{S}$ and taking an action $a \in \mathcal{A}$

$$Q_\gamma^{\pi_\theta}(s, a) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a, a_t \sim \pi_\theta(\cdot|s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)] \tag{1a}$$

$$= r(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a), a' \sim \pi_\theta(\cdot|s')}[Q_\gamma^{\pi_\theta}(s', a')] \tag{1b}$$

Where, equation 1b is called the Bellman Equation. Bellman equation is popularly used to estimate the Q-function using just empirical data collected on the MDP. Q-function approximation methods typically use $\gamma < 1$ for stable estimation of the Q-function. We also similarly define value function $V_\gamma^{\pi_\theta}(s) = \mathbb{E}_{a \sim \pi_\theta(\cdot|s)}[Q_\gamma^{\pi_\theta}(s, a)]$.

Modern RL algorithms generally solve for θ^* by estimating the gradient of policy performance $J_\gamma(\pi_\theta)$ with respect to policy parameters θ. This is also commonly referred to as the policy gradient theorem $\nabla_\theta J_1(\pi_\theta)$ (Sutton et al., 1999) which says

$$\nabla_\theta J_\gamma(\pi_\theta) = \mathbb{E}_{(s,a) \sim d_\gamma^{\pi_\theta}}[Q_\gamma^{\pi_\theta}(s, a) \cdot \nabla_\theta \log \pi_\theta(a|s)], \quad \gamma \in [0, 1]. \tag{2}$$

Here, $d_\gamma^{\pi_\theta}$ is the average state-action discounted stationary distribution, which is defined as the cumulative sum of discounted state-action occupancy across the time horizon.

$$d_\gamma^{\pi_\theta}(s, a) := (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s, a_t = a | s_0 \sim d_0, a_t \sim \pi_\theta(s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t), \gamma < 1) \tag{3a}$$

$$d_1^{\pi_\theta}(s, a) := \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \mathbb{P}(s_t = s, a_t = a | s_0 \sim d_0, a_t \sim \pi_\theta(s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t), \gamma = 1). \tag{3b}$$

In this paper we make a standard assumption that the Markov chain induced by policy π_θ is ergodic. In particular, this implies that $d_\gamma^{\pi_\theta}(s, a) > 0$ for all state action pairs (s, a) (Puterman, 2014). In scenarios

where we are trying to optimize for $J_1(\pi)$, estimating the policy gradient becomes difficult. This is because the Bellman equation cannot be used to estimate Q-function for $\gamma = 1$. As a compromise policy gradient for average reward scenarios are instead approximated by calculating the Q-function for a discounting factor $\gamma < 1$, but close to 1, and using that estimate in the policy gradient equation 2

$$\hat{\nabla}_\theta J_1(\pi_\theta) = \mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[Q_1^{\pi_\theta}(s,a) \cdot \nabla_\theta \log \pi_\theta(a|s)], \tag{4a}$$

$$\approx \mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[Q_\gamma^{\pi_\theta}(s,a) \cdot \nabla_\theta \log \pi_\theta(a|s)], \quad \gamma < 1. \tag{4b}$$

In this paper we argue that the policy gradient calculated in this manner induces a significant residual error, which keeps on compounding as the reinforcement learning training proceeds even leading to a sub optimal solution. The following equation, derived in Proposition 2 characterizes that error,

$$\nabla_\theta J_1(\pi_\theta) = \mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[Q_\gamma^{\pi_\theta}(s,a) \cdot \nabla_\theta \log \pi_\theta(a|s)] + \underbrace{(1-\gamma)\mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[\nabla_\theta \log d_1^{\pi_\theta}(s) \cdot V_\gamma^{\pi_\theta}(s)]}_{\textbf{Residual Error}} \tag{5}$$

In this paper, we prove that this residual error is significant. What more, we also propose another method to exactly obtain the policy gradient, for all values of the discounting factor including $\gamma = 1$ which we call as the log density gradient. Our estimation of log density gradient utilises average state-action discounted distributional formulation of a reinforcement learning problem which re-states $J_\gamma(\pi)$ as expectation under $d_\gamma^{\pi_\theta}$ (Nachum et al., 2019; Uehara et al.) as

$$J_\gamma(\pi) = \mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[r(s,a)].$$

Under this formulation, policy gradient can similarly be obtained using log derivative trick as follows,

$$\nabla_\theta J_\gamma(\pi) = \mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[\nabla_\theta \log d_\gamma^{\pi_\theta}(s,a) \cdot r(s,a)]. \tag{6}$$

We refer to $\nabla_\theta \log d_\gamma^{\pi_\theta}$ as the log density gradient. A key advantage of log density gradient is that it would allow us to approximate policy gradient for average reward scenarios in a provable manner. In this work, we show that log density gradient can be approximated even under average reward scenarios ($\gamma = 1$). To that end, we first propose a model based approach to approximate the log density gradient for tabular scenarios. Under reversibility assumptions we then show that this log density gradient can also be approximated using TD method. Since reversibility assumption is highly restrictive, we finally propose a min-max version of log density gradient which allows to approximate $\nabla_\theta \log d^{\pi_\theta}$ using empirical samples. Our experimental further show that this manner of policy gradient estimation can potentially make reinforcement learning sample efficient, thus helping them scale.

3 Survey of Related Work and Comparison

In this section we will discuss existing studies in policy gradient methods including the framework of log density gradient first introduced by Morimura et al. (2010). We also briefly discuss density ratio learning methods which have been very popular in off-policy evaluation. A short discussion on Temporal Difference (TD) learning methods may be found in Appendix 8.1.

3.1 Policy Gradient Methods

Literature survey: Policy gradient methods are a widely studied topic in reinforcement learning. One of the earliest works in this area proposed a closed-form solution for evaluating policy gradients called the policy gradient theorem (Sutton et al., 1999). Initially implementations for policy gradient methods used episodic estimates to update policy parameters (Williams, 1992) and GPOMDP (Baxter & Bartlett, 2001). Unfortunately this way of implementing policy gradient suffered from high variance, thus inhibiting scalability to large problem spaces (Schulman et al., 2016). To address this problem, actor-critic methods approximate the Q-function or advantage function using an additional neural network, which are then used to update the policy (Mnih et al., 2016; Schulman et al., 2016). Furthermore policy gradient methods are also designed to

be compatible with deterministic policies (Lillicrap et al., 2016; Silver et al., 2014). Recently Trust region methods, such as Trust Region Policy Optimization (TRPO) Schulman et al. (2015) and Proximal Policy Optimization (PPO) Schulman et al. (2017) have been introduced which update policies while ensuring monotonic performance improvement. To the best of our knowledge, Log density gradient has only been discussed in Morimura et al. (2010) in which a TD method to estimate log density gradient for average reward scenarios by using reversible backward Markov chain is proposed.

Comparison: In our paper, we re-introduce the idea of log density gradient introduced by Morimura et al. (2010) for estimating gradient in the average reward scenario. Morimura et al. (2010) was also the first work to find out the residual error in policy gradient approximation (Proposition 2). Additionally, this work proposes estimating log density gradient specifically for average reward scenarios ($\gamma = 1$) using a TD update, with additional extensions for linear function approximation.

Our work not only fixes many technical gaps evident in the theory of log density gradient as proposed by Morimura et al. (2010) but also builds on them to make log density gradient practical. We first define log density gradient (equation 10) over a range of discounting factor $\gamma \in [0, 1]$, which also includes the average reward scenario. Using Lemma 2 and 3 we then prove mathematical conditions under which the log density gradient is unique and can be exactly calculated. We further use this relation to propose a TD form of updates (equation 13) for log density gradient estimation for all values of discounting factor $\gamma \in [0, 1]$ as against the average reward scenario proposed by Morimura et al. (2010). In Lemma 4 we further prove that these TD updates converge to a unique solution for all values of discounting factor γ except 1. Thus, effectively demonstrating that TD-updates proposed by Morimura et al. (2010) *does not* converge to the true log density gradient, further limiting their use for large scale problems. Additionally, to make log density gradient estimation viable for practical problems, we propose a min-max optimization approach (equation 16) that allows us to estimate log density gradient using empirical samples. We also demonstrate that under linear function approximation settings, this min-max optimization not only has a closed form but also converges to a unique solution (Theorem 1). Under weighted updates, as proposed in algorithm 1 we also show a bound on sample complexity of log density gradient estimation of the order of $\mathcal{O}(\frac{1}{\sqrt{n}})$.

3.2 Density Ratio Learning

Literature survey: Off-policy evaluation estimates the performance of a target policy π using an offline dataset generated by a behavior policy μ (Voloshin et al., 2019; Katdare et al.). This is done by estimating the average state-action density ratio $\frac{d^{\pi}_{\gamma}}{d^{\mu}}$, which allows approximation of the target policy's performance. In this work, we are primarily interested in the DICE class of off-policy evaluation algorithms (Zhang et al., 2020b; Nachum et al., 2019; Zhang et al., 2020a). These algorithms typically approximate the divergence (for some $f-$divergence of their choice) between the two distributions d^{π} and d^{μ} in their convex dual form, eliminating the need to obtain samples from d^{π}, which results in a min-max form optimization problem.

Comparison: Inspired by the DICE class of algorithms we too propose a min-max form of estimating the log density gradient. We show that such a method of estimating log density gradient converges to the true policy under linear function approximations assumptions. We also show that the sample complexity of such an estimator is of the order $(n^{-1/2})$, with n being the number of on-policy samples.

4 Log Density Gradient

In this section, we introduce log density gradient and further show that classical policy gradient typically ignores a residual error which may be significant. We then propose a TD version of our log density gradient method that estimates the policy gradient without a need to account for this error.

4.1 Model Based Log Density Gradient

The log density gradient method attempts to estimate the gradient of log of average state-action discounted stationery distribution $d^{\pi_\theta}_{\gamma}$. We start by observing that $d^{\pi_\theta}_{\gamma}$ satisfies an identity called the Bellman flow equation Liu et al. (2018).

Lemma 1. *The average state-action density distribution satisfies the following identity for all $\gamma \in [0,1]$,*

$$d_\gamma^{\pi_\theta}(s') = (1-\gamma)d_0(s') + \gamma \sum_{s,a} d_\gamma^{\pi_\theta}(s,a)\mathcal{P}(s'|s,a) \tag{7}$$

We prove this result in the appendix 8.3. Note that this equation is similar to the Bellman equation but in a backward manner, which means, we need samples from the backward conditional distribution (defined rigorously in Section 4.2). It can also be understood as a form of flow conservation, wherein the flow out of state s' (LHS) would be equal to the flow into s' (RHS).

Lemma 2. *For all $\gamma \in [0,1]$ the solution to the following optimisation is unique and equal to $d_\gamma^{\pi_\theta}$.*

$$\arg\min_{w:\mathcal{S}\to\mathbb{R}} \sum_{s'} \left(w(s') - (1-\gamma)d_0(s') + \gamma \sum_{s,a} w(s)\pi_\theta(a|s)\mathcal{P}(s'|s,a) \right)^2 + \frac{\lambda}{2}(\sum_s w(s) - 1)^2 \tag{8}$$

Detailed proof can be found in the appendix 8.4. Intuitively speaking, the term $\frac{\lambda}{2}(\sum_s w(s)-1)^2$ is redundant for $\gamma < 1$. It becomes useful for average reward scenarios wherein $\gamma = 1$ ensures uniqueness. Although it is hard to estimate a closed form solution for $\nabla_\theta \log d_\gamma^{\pi_\theta}$, it is still possible to estimate it numerically. Using equation 7 we can similarly calculate an equivalent identity for $\nabla_\theta \log d_\gamma^{\pi_\theta}$

$$d_\gamma^{\pi_\theta}(s')\nabla_\theta \log d_\gamma^{\pi_\theta}(s') = \gamma \sum_{s,a} d_\gamma^{\pi_\theta}(s,a)\mathcal{P}(s'|s,a)\nabla_\theta \log d_\gamma^{\pi_\theta}(s,a). \tag{9}$$

Multiplying both sides by $\pi_\theta(a'|s')$ and recalling that $d_\gamma^{\pi_\theta}(s,a) = d_\gamma^{\pi_\theta}(s)\pi_\theta(a|s)$, we obtain

$$d_\gamma^{\pi_\theta}(s',a')(\nabla_\theta \log d_\gamma^{\pi_\theta}(s',a') - \nabla_\theta \log \pi(a'|s')) = \gamma \sum_{s,a} d_\gamma^{\pi_\theta}(s,a)\nabla_\theta \log d_\gamma^{\pi_\theta}(s,a)\mathcal{P}(s'|s,a)\pi_\theta(a'|s') \tag{10}$$

Similar to Lemma 2 we can estimate $\nabla_\theta \log d_\gamma^{\pi_\theta}$ by solving for the equation 10 above. To that end, we propose the following optimization (where $\lambda > 0$ is a fixed regularizer)

$$\min_{w:\mathcal{S}\times\mathcal{A}\to\mathbb{R}^n} \left\{ \mathbb{E}_{(s',a')\sim d_\gamma^{\pi_\theta}} \|\nu(s',a')\|^2 + \frac{\lambda}{2}\left\| E_{(s,a)\sim d_\gamma^{\pi_\theta}}[w(s',a')] \right\|^2 \right\} \tag{11}$$
$$\nu(s',a') := d_\gamma^{\pi_\theta}(s',a')(w(s',a') - \nabla_\theta \log \pi_\theta(a'|s')) - \gamma \sum_{s,a} d_\gamma^{\pi_\theta}(s,a)\mathcal{P}(s'|s,a)\pi_\theta(a'|s')w(s,a)$$

Lemma 3. *The solution to equation 11 is unique and equal to $\nabla_\theta \log d_\gamma^{\pi_\theta}$ for all $\gamma \in [0,1]$.*

We describe the proof for this Lemma in the appendix 8.5. Similar to Lemma 11, the constraint $\frac{\lambda}{2}\|E_{(s,a)\sim d_\gamma^{\pi_\theta}}[w(s',a')]\|^2$ is only useful for average reward scenarios $\gamma = 1$. The proof follows from the fact that the solution to the equation 11 can be written in a linear form $A \cdot w = b$ and showing that A is invertible. It is worth reiterating that, once we have an estimate $\nabla_\theta \log d_\gamma^{\pi_\theta}$, we can use this estimate to approximate the policy gradient using equation 6. We will now recall two important properties of log density gradient.

Proposition 1. *The policy gradient method as mentioned in equation 6 is exactly equal to the classical policy gradient (Sutton et al., 1999) as mentioned in equation 2.*

Detailed proof for this proposition can be found in Appendix 8.2. In essence this means that log density gradient calculates the same policy gradient but using a different formulation. We show next that this formulation allows us to correct for the residual error in policy approximation which is typically ignored in many actor-critic implementations of policy gradient methods.

Proposition 2. *The following identity, stated in equation 5, is true*

$$\nabla_\theta J_1(\pi_\theta) = \mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[Q_\gamma^{\pi_\theta}(s,a)\cdot\nabla_\theta \log \pi_\theta(a|s)] + \underbrace{(1-\gamma)\mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[\nabla_\theta \log d_1^{\pi_\theta}(s)\cdot V_\gamma^{\pi_\theta}(s)]}_{\textit{Residual Error}}$$

Proof. From the definition of log density gradient equation 6 we have $\nabla_\theta J_1(\pi_\theta) = \mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[\nabla_\theta \log d_1^{\pi_\theta}(s,a) \cdot r(s,a)]$. Let $\gamma < 1$, and we use the Bellman equation 1b to obtain

$$\nabla_\theta J_1(\pi_\theta) = \mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[\nabla_\theta \log d_1^{\pi_\theta}(s,a) \cdot (Q_\gamma^{\pi_\theta}(s,a) - \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[V_\gamma^{\pi_\theta}(s')])] \tag{12a}$$

$$= \mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[(\nabla_\theta \log d_1^{\pi_\theta}(s) + \nabla_\theta \log \pi_\theta(a|s)) \cdot (Q_\gamma^{\pi_\theta}(s,a) - \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[V_\gamma^{\pi_\theta}(s')])] \tag{12b}$$

$$= \mathbb{E}_{(s,a)\sim d_1^{\pi_\theta}}[Q_\gamma^{\pi_\theta}(s,a) \cdot \nabla_\theta \log \pi_\theta(a|s)] + (1-\gamma)\mathbb{E}_{s\sim d_1^{\pi_\theta}}[\nabla_\theta \log d_1^{\pi_\theta}(s,a) \cdot V^{\pi_\theta}(s)] \tag{12c}$$

Here, we go from equation 12a to 12b by utilizing $\nabla_\theta \log d_\gamma^{\pi_\theta}(s,a) = \nabla_\theta \log d_\gamma^{\pi_\theta}(s) + \nabla_\theta \log \pi_\theta(a|s)$. We finally go from 12b to 12c by using a key identity of log density gradient equation 10.

using identity for log density gradient as shown in the equation 9. Note that the first term in equation 12b is the practical instantiation of classical policy gradient theorem, while the second term $(1-\gamma)\mathbb{E}_{s\sim d_1^{\pi_\theta}}[\nabla_\theta \log d_1^{\pi_\theta}(s) \cdot V_\gamma^{\pi_\theta}(s)]$ being the residual error. This completes the proof. □

Actor-Critic implementation of policy gradient methods first approximate the Q-function using a discounting factor γ strictly less than 1 and then approximate the policy gradient $\nabla_\theta J_1(\pi_\theta)$ using equation 4b. This leads to a residual error in policy gradient approximation as shown in Proposition 2. We believe that correcting for this gradient estimation can potentially make policy gradient algorithms sample efficient and scalable to complex problems.

Although the solution to equation equation 11 is unique, solving it requires access the transition matrix $\mathcal{P}$ which is impractical for complex environments. Additionally, the log density gradient estimation also becomes computationally infeasible with exponential scaling of state-action. We thus propose a temporal difference approach to estimate log density gradient from empirical data next .

4.2 Temporal Difference Log Density Gradient

To propose an update equation for temporal difference (TD) method, we first begin with re-arranging few terms in equation 10 for log density gradient.

$$\nabla_\theta \log d_\gamma^{\pi_\theta}(s',a') = \nabla_\theta \log \pi_\theta(a'|s') + \gamma \sum_{s,a} \frac{d_\gamma^{\pi_\theta}(s,a)\mathcal{P}(s'|s,a)\pi_\theta(a'|s')}{d_\gamma^{\pi_\theta}(s',a')} \nabla_\theta \log d_\gamma^{\pi_\theta}(s,a)$$

We define the backward distribution of (s,a) given (s',a') as

$$\mathcal{P}_b(s,a|s',a') := \frac{d_\gamma^{\pi_\theta}(s,a)\mathcal{P}(s'|s,a)\pi_\theta(a'|s')}{d_\gamma^{\pi_\theta}(s',a')} = \frac{d_\gamma^{\pi_\theta}(s,a)\mathcal{P}^{\pi_\theta}(s',a'|s,a)}{d_\gamma^{\pi_\theta}(s',a')}$$

which is a consequence of Bayes' rule. The summation therefore becomes an expectation under $\mathcal{P}_b$. The log density gradient is therefore said to follow a backward recursion and it requires samples from backward conditional probability $\mathcal{P}_b$ to estimate log density gradient[1]. We first generalize algorithm of Morimura et al. (2010), who do it only for $\gamma = 1$, to estimate log density gradient w for all discounting factor $\gamma \in [0,1]$ in form of a temporal difference (TD(0)) method where our update equation is

$$w(s',a') \leftarrow w(s',a') + \alpha[\gamma w(s,a) + g(s',a') - w(s',a')] \tag{13}$$

with $(s',a') \sim d_\gamma^{\pi_\theta}, (s,a) \sim \mathcal{P}_b(\cdot|s',a')$ and $g(s',a') := \nabla_\theta \log \pi_\theta(a'|s')$. Define operator Y_γ to capture the behaviour of update rule equation 13 after taking expectation,

$$(Y_\gamma \cdot w)(s',a') := \gamma \mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[w(s,a)|(s',a')] + g(s',a').$$

We can write this in matrix form as follows,

$$Y_\gamma \cdot W = \gamma D_{\pi_\theta}^{-1} \mathcal{P}_{\pi_\theta}^\top D_{\pi_\theta} W + G \tag{14}$$

[1]Although for $\gamma = 1$ we can use samples from $\mathcal{P}$ as well (Morimura et al., 2010)

where, $W \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|\times n}$ is the matrix with every row corresponding to $w(s,a)$ for each state-action pair (s,a). Similarly, $G \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|\times n}$ has its rows as $\nabla_\theta \log \pi_\theta$ for each state-action pair. Let $\mathcal{P}_{\pi_\theta}, D_{\pi_\theta} \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|\times|\mathcal{S}|\cdot|\mathcal{A}|}$ where $(\mathcal{P}_{\pi_\theta})_{((s,a),(s',a'))} = \mathbb{P}^{\pi_\theta}(s',a'|s,a)$ and D_{π_θ} is a diagonal matrix whose every element correspond to $d_\gamma^{\pi_\theta}$ for each state-action pair. We use this matrix form for the operator Y_γ in the proof of the following lemma.

Lemma 4. *Let $w_0 \in \Delta(\mathcal{S},\mathcal{A})$ be an arbitrary initial guess. Let $w_k = Y_\gamma \cdot w_{k-1}$ for all natural numbers $k \geq 1$. For $\gamma \in [0,1)$, the operator Y_γ is a contraction, and $\{w_k\}_{k\geq 0}$ converges to a unique fixed point $\nabla_\theta \log d_\gamma^{\pi_\theta}$.*

Detailed proof of Lemma 4 can be found in Appendix 8.6. Extension of Lemma 4 to linear function approximation, and proof of convergence for the same, can be found in the Appendix 8.7. Although TD methods are known to converge, they still suffer from two problems. One, the access to samples from backward conditional probability. Two, scalability to large problem spaces. We attempt to solve both of these problems in the next section where we propose a min-max optimization procedure for estimating the log density gradient.

5 Min-Max Log Density Gradient

In this section, we propose another approach for evaluating the log density gradient which uses the min-max form for optimization which does not need samples from the backward distribution. Min-max optimizations also allow us to use a large variety of function classes like neural networks to estimate log density gradient.

Let us return to the loss function that we initially propose in equation 11. Classical machine learning algorithms typically require loss function in the form of an expectation under a distribution. These kind of algorithms are often called as Empiricial Risk Minimization, which allows us to approximate the loss using samples from that distribution. Consider a modified form of the optimization proposed in equation 11 (the modification is that $\delta(s',a')$ is divided by $d_\gamma^{\pi_\theta}(s',a')$ where the ergodicity assumption ensures this operation is well defined),

$$\begin{aligned}
&\underset{w\in\mathcal{S}\times\mathcal{A}\to\mathbb{R}^n}{\arg\min}\ \mathbb{E}_{(s',a')\sim d_\gamma^{\pi_\theta}}\left[\left\|\frac{\nu(s',a')}{d_\gamma^{\pi_\theta}(s',a')}\right\|^2\right] + \frac{\lambda}{2}\|\mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[w(s,a)]\|^2 \qquad (15)\\
&\nu(s',a') := d_\gamma^{\pi_\theta}(s',a')(w(s',a') - \nabla_\theta \log \pi_\theta(a'|s')) - \gamma \sum_{s,a} d_\gamma^{\pi_\theta}(s,a)\mathcal{P}(s'|s,a)\pi(a'|s')w(s,a)
\end{aligned}$$

The denominator term $d_\gamma^{\pi_\theta}$ is added to simplify the final optimization, which we shall see soon. We also add the term $\frac{\lambda}{2}\|\mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[w(s,a)]\|^2$ to satisfy one of the properties of the gradient of log density $\mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[\nabla_\theta \log d_\gamma^{\pi_\theta}(s,a)] = 0$.

It is worth noting that equation 15 is just a re-weighting of equation 11 with the $\frac{1}{d_\gamma^{\pi_\theta}(s,a)}$. This implies that the optimal solution for the both the equation is the same because the minimum value for both the optimization can only be reached when $w(s,a) = \nabla_\theta \log d_\gamma^{\pi_\theta}(s,a)$. By exploiting the Fenchel-duality, we can re-write this optimization in the minimax form (Rockafellar, 2015; Zhang et al., 2020b).

$$\begin{aligned}
&\arg \min_{w:\mathcal{S}\times\mathcal{A}\to\mathbb{R}^d} \max_{f:\mathcal{S}\times\mathcal{A}\to\mathbb{R}^d,\tau\in\mathbb{R}^d} L_\gamma(w,f,\tau) := \Big\{\mathbb{E}_{(s',a')\sim d_\gamma^{\pi_\theta}}[f(s',a')\cdot w(s',a')]\\
&- \mathbb{E}_{(s',a')\sim d_\gamma^{\pi_\theta}}[f(s',a')\cdot \nabla_\theta \log \pi_\theta(a'|s')] - \gamma \mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a),a'\sim\pi_\theta(\cdot|s')}[f(s',a')]\cdot w(s,a)]\\
&- \frac{1}{2}\mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[\|f(s',a')\|^2] + \lambda(\tau\cdot \mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[w(s,a)] - \frac{1}{2}\|\tau\|^2)\Big\} \qquad (16)
\end{aligned}$$

In many cases searching over all set of functions is not possible, hence we search over tractable function classes $\mathcal{W},\mathcal{F}$ and the aim is to approximate

$$\nabla_\theta \log d_\gamma^{\pi_\theta} \approx \arg\min_{w\in\mathcal{W}} \max_{f\in\mathcal{F},\tau\in\mathbb{R}^n} L_\gamma(w,f,\tau).$$

Such a practical consideration allows us to use different types of function approximators like linear function approximation, neural networks, and reproducible kernel Hilbert spaces (RKHS). We will now provide an

update rule to solve equation 16 under linear function approximation. For that we choose a feature map $\Phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and parameters $\alpha, \beta \in \mathbb{R}^{d\times n}$ that need to be learnt, so that we can approximate the optima of equation 16, $w^*(s,a)$ and $f^*(s,a)$ with $\alpha^T\Phi(s,a)$, and $\beta^T\Phi(s,a)$ respectively, for each state action pair (s,a). The update rule is

$$\delta_t = \Phi_t\Phi_t^T - \gamma\Phi_t(\Phi_t')^T \tag{17a}$$

$$\alpha_{t+1}^T = \alpha_t^T - \varepsilon_t(\beta^T\delta_t + \lambda(\tau\Phi_t^T)) \tag{17b}$$

$$\beta_{t+1}^T = \beta_t^T + \varepsilon_t(\alpha_t^T\delta_t - g_t\Phi_t^T - \beta_t^T\Phi_t\Phi_t^T) \tag{17c}$$

$$\tau_{t+1} = \tau_t + \varepsilon_t(\lambda(\alpha_t^T\Phi_t - \tau_t)) \tag{17d}$$

where, $\Phi_t := \Phi(s_t, a_t)$ is the feature encountered at time t, $g_t := \nabla_\theta \log \pi_\theta(a_t|s_t)$, and $\Phi_t' := \Phi_t(s_t', a_t')$ for $(s_t, a_t) \sim d_\gamma^{\pi_\theta}, s_t' \sim \mathcal{P}(\cdot|s_t, a_t), a_t' \sim \pi_\theta(a_t'|s_t')$. We first re-write the updates in equation 17 in form of $d_t = [\alpha_t, \beta_t, \tau_t^T]$ so that the updates can be written in matrix form $d_{t+1} = d_t + \varepsilon_t(G_{t+1}d_t + h_{t+1})$, where, G_{t+1}, h_{t+1} are as follows,

$$G_{t+1} := \begin{bmatrix} 0 & -A_t & -\lambda\Phi_t \\ A_t & -C_t & 0 \\ \lambda\Phi_t^T & 0 & -\lambda \end{bmatrix}, \quad h_{t+1} := \begin{bmatrix} 0 \\ -B_t \\ 0 \end{bmatrix}$$

and $A_t := (\Phi_t\Phi_t^T - \gamma\Phi_t(\Phi_t')^T), B_t := \Phi g_t^T, C_t := \Phi_t\Phi_t^T$. We can calculate the expectation for each of these matrices as follows,

$$G := \mathbb{E}_p[G_{t+1}] = \begin{bmatrix} 0 & -A & -\lambda\Psi D_{\pi_\theta} e \\ A & C & 0 \\ \lambda e^T D_{\pi_\theta}\Psi^T & 0 & -\lambda \end{bmatrix}, \quad h := \mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}}[h_{t+1}] = \begin{bmatrix} 0 \\ -B \\ 0 \end{bmatrix}$$

Here, each column of $\Psi \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|\times n}$ is the feature vector $\Phi(s,a)$, for each $(s,a) \in \mathcal{S}\times\mathcal{A}$ and $e \in \mathbb{R}^n$ is a vector of 1's at every element. We can similarly write $A = \Psi D_{\pi_\theta}(I - \gamma P_{\pi_\theta})\Psi^T, B = \Psi D_{\pi_\theta} G^T, C = \Psi D_{\pi_\theta}\Psi^T$ and $E_p[\cdot] := \mathbb{E}_{(s,a)\sim d_\gamma^{\pi_\theta}, s'\sim\mathcal{P}(\cdot|s,a), a'\sim\pi_\theta(\cdot|s')}[\cdot]$. We can now prove the convergence of linear function approximation under the following key assumptions.

Assumption 1. *1. The matrix Ψ has linearly independent columns.*

2. The matrix A is non-singular or the regularizer $\lambda > 0$.

3. The feature matrix Φ has uniformly bounded second moments.

Theorem 1. *Under the assumptions 1, the update equation 17 converges in probability to a unique solution. That is, $\lim_{t\to\infty} d_t = G^{-1}h$ in probability.*

The detailed proof is provided in Appendix 8.8. The proof is similar to (Zhang et al., 2020b, Theorem 2) and invokes theorem 2.2 Borkar & Meyn (2000).

We provide a sample complexity analysis for a projected version of the update rule equation 17. To that end, we propose Algorithm 1 called the Projected Log Density Gradient. We choose closed, bounded and convex sets $X \subset \mathbb{R}^{d\times n}, Y \subset \mathbb{R}^{d\times n}, Z \subset \mathbb{R}^{1\times n}$ and define a projection operator Π_X, Π_Y, Π_Z that project our variables $\alpha_t, \beta_t, \tau_t$ onto X, Y, Z respectively. Moreover, we choose a learning rate $\{\varepsilon_t\}_{t=1}^m$ where we run the algorithm for m steps. The details of the choice of learning rate are found in Appendix 8.9.

Theorem 2. *Under assumptions 1 for $(\bar{\alpha}, \bar{\beta}, \bar{\tau})$ obtained from Algorithm 1 after m steps, the optimality gap $\epsilon_g(\bar{\alpha}\bar{\beta}, \bar{\tau})$ (defined below) is bounded with probability $1-\delta$ as follows,*

$$\epsilon_g(\bar{\alpha}, \bar{\beta}, \bar{\tau}) := \max_{(\beta,\tau)\in Y\times Z} L(\bar{\alpha}, \beta, \tau) - \min_{\alpha\in X} L(\alpha, \bar{\beta}, \bar{\tau}) \leq C_0\sqrt{\frac{5}{m}}(8 + 2\log\frac{2}{\delta}) \text{ w.p. } 1-\delta$$

where, C_0 is a constant which is a function of the sets X, Y, Z, and the second moment of Φ.

We present the proof of this result in appendix 8.9. This result essentially shows us that the upper-bound for log density gradient estimation requires $O(\frac{1}{\sqrt{m}})$ (where m is the number of steps the algorithm runs for) samples to learn an accurate estimation.

Algorithm 1 Projected Log Density Gradient

1: **for** $t = 1, 2, ..., m$ do:
2: $\delta_t = \Phi_t \Phi_t^T - \gamma \Phi_t (\Phi_t')^T$
3: $\alpha_{t+1}^T = \Pi_X(\alpha_t^T - \varepsilon_t(\beta^T \delta_t + \lambda(\tau \Phi_t^T)))$
4: $\beta_{t+1}^T = \Pi_Y(\beta_t^T + \varepsilon_t(\alpha_t^T \delta_t - g_t \Phi_t^T - \beta_t^T \Phi_t \Phi_t^T))$
5: $\tau_{t+1} = \Pi_Z(\tau_t + \varepsilon_t(\lambda(\alpha_t^T \Phi_t - \tau_t)))$
6: **Return** $\bar{\alpha}$, $\bar{\beta}$, $\bar{\tau}$
Where, $\bar{\alpha} = \frac{\sum_{i=1}^n \varepsilon_i \alpha_i}{\sum_{i=0}^n \varepsilon_i}$, $\bar{\beta} = \frac{\sum_{i=1}^n \varepsilon_i \beta_i}{\sum_{i=0}^n \varepsilon_i}$, $\bar{\tau} = \frac{\sum_{i=1}^n \varepsilon_i \tau_i}{\sum_{i=0}^n \varepsilon_i}$

6 Experiments

In this section, we present a proof of concept for our log density gradient estimation on two sets of environments 5×5 and 3×3 gridworld environment (Towers et al., 2023). For the gridworld experiments, we approximate log density gradient by using linear function approximation. Here, the features are $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{|\mathcal{S}| \cdot |\mathcal{A}|}$ such that it maps every state to the corresponding standard basis vector. Our results for 5×5 are in Figure 2 and for 3×3 in Figure 3.

We compare our algorithm against 3 different baselines. The first is theoretical log density gradient as described in Lemma 3. The second baseline implements REINFORCE algorithm, which is the practical rendition of the policy gradient theorem (Williams, 1992). The third is theoretical policy gradient method which exactly computes the classical policy gradient theorem, as in equation 4b (Sutton et al., 1999).

We observe in that both log density gradient approaches are more sample efficient than both policy gradient approaches. This is because policy gradient methods approximate the gradient for average reward scenarios ($\gamma = 1$) by estimating a Q-function for a discounting factor less than 1. Moreover, we observe that our method tends to outperform REINFORCE with much reduced variance. Our approach is always very close in performance to the theoretical log density gradient which serves to validate correctness of our algorithm. In 5×5 gridworld we also observe our algorithm to outperforms theoretical log density gradient. This is because, theoretical log density gradient suffers from some numerical computation issues arising from average reward scenarios.

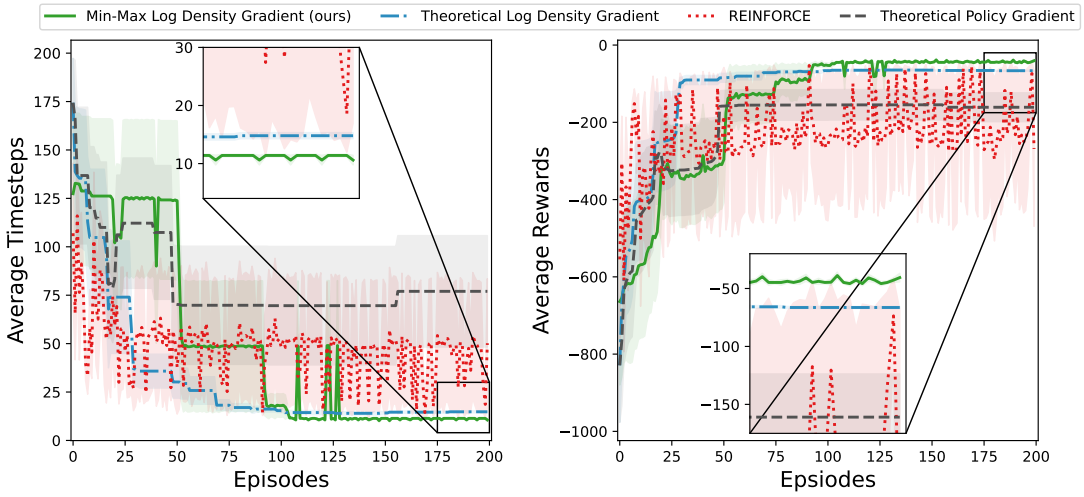

Figure 2: For 5×5 gridworld, comparison of Log Density Gradient algorithms (in light green) as compared to REINFORCE (light red), theoretical policy gradient (gray) and theoretical log density gradient (blue). We observe that our empirical algorithm comfortably outperforms the other baselines.

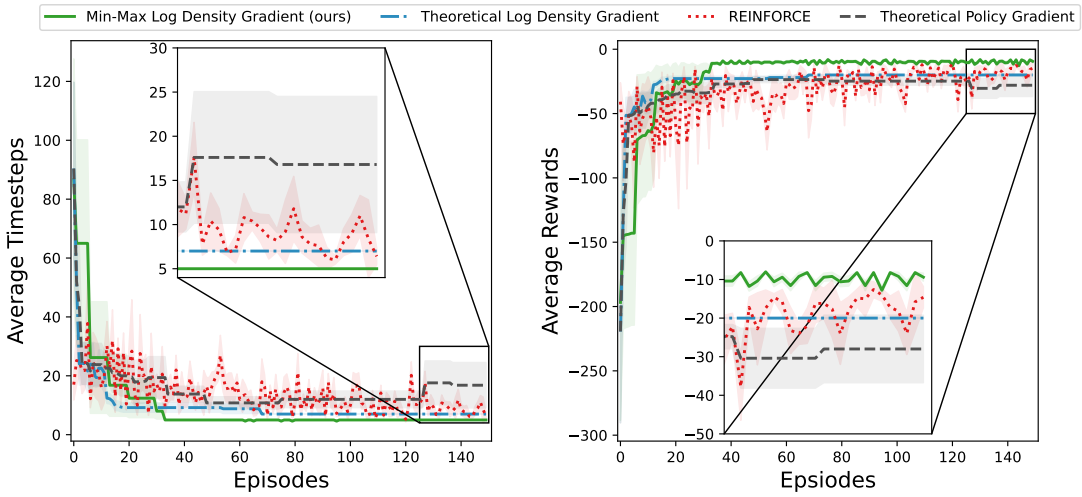

Figure 3: For 3×3 gridworld, comparison of Log Density Gradient algorithms (in light green) as compared to REINFORCE (light red), theoretical policy gradient (gray) and theoretical log density gradient (blue). We observe that our empirical algorithm comfortably outperforms the other baselines.

7 Conclusion and Future Work

We present log density gradient algorithm that estimates policy gradient using state-action discounted formulation of a reinforcement learning problem. We observe that policy gradient estimated in this manner, corrects for a residual error common in many reinforcement learning tasks. We show that with a known model, we can exactly calculate the gradient of the log density by solving two sets of linear equations. We further propose a TD(0) algorithm to implement the same, but it needs samples from the backward Markov chain, which becomes too restrictive. Therefore, we propose a min-max optimization that estimates log density gradient using just on-policy samples. We not only prove theoretical properties like convergence and uniqueness but also experimentally demonstrate that our method is sample efficient as compared to classical policy gradient methods like REINFORCE. This approach looks promising, and further studies of log density gradient will focus on scaling their performance to complex tasks.

References

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *J. Artif. Int. Res.*, 2001.

Borkar, V. S. and Meyn, S. P. The o.d. e. method for convergence of stochastic approximation and reinforcement learning. 2000.

Boyan, J. A. Least-squares temporal difference learning. In *ICML*, pp. 49–56, 1999.

Bradtke, S. J. and Barto, A. G. Linear least-squares algorithms for temporal difference learning. *Mach. Learn.*, 1996.

Gelada, C. and Bellemare, M. G. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*.

Hallak, A. and Mannor, S. Consistent on-line off-policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, 2017.

Ilyas, A., Engstrom, L., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. A closer look at deep policy gradients. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

John Schulman, Barret Zoph, C. K. et al. Chatgpt: Optimizing language models for dialogue, 2023.

Kakade, S. M. A natural policy gradient. In Dietterich, T. G., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14*, 2001.

Katdare, P., Liu, S., and Campbell, K. D. Off environment evaluation using convex risk minimization. In *2022 International Conference on Robotics and Automation, ICRA 2022*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR*, 2016.

Liu, B., Liu, J., Ghavamzadeh, M., Mahadevan, S., and Petrik, M. Finite-sample analysis of proximal gradient TD algorithms. In Meila, M. and Heskes, T. (eds.), *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015*.

Liu, Q., Li, L., Tang, Z., and Zhou, D. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems 31*, 2018.

Maei, H. R. Gradient temporal-difference learning algorithms. In *Ph.D thesis*, 2011.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, 2016.

Morimura, T., Uchibe, E., Yoshimoto, J., Peters, J., and Doya, K. Derivatives of logarithmic stationary distributions for policy gradient reinforcement learning. *Neural Comput.*, 2010.

Nachum, O., Chow, Y., Dai, B., and Li, L. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems 32*, 2019.

Nemirovski, A., Juditsky, A. B., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.*, 2009.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons. 2014.

Robbins, H. and Monro, S. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22 (3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

Rockafellar, R. T. *Convex Analysis*. Princeton University Press, 2015.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. A. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML*, 2014.

Silver, D., Schrittwieser, J., Simonyan, K., and thers. Mastering the game of go without human knowledge. *Nat.*, 2017.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 1988.

Sutton, R. S., McAllester, D. A., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1999.

Tesauro, G. Practical issues in temporal difference learning. *Mach. Learn.*, 1992.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., and Younis, O. G. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Uehara, M., Huang, J., and Jiang, N. Minimax weight and q-function learning for off-policy evaluation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*.

Voloshin, C., Le, H. M., Jiang, N., and Yue, Y. Empirical study of off-policy policy evaluation for reinforcement learning. 2019.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 1992.

Yuan, R., Gower, R. M., and Lazaric, A. A general sample complexity analysis of vanilla policy gradient. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2022*, Proceedings of Machine Learning Research, 2022.

Zhang, R., Dai, B., Li, L., and Schuurmans, D. Gendice: Generalized offline estimation of stationary values. *CoRR*, abs/2002.09072, 2020a. URL https://arxiv.org/abs/2002.09072.

Zhang, S., Liu, B., and Whiteson, S. Gradientdice: Rethinking generalized offline estimation of stationary values. *CoRR*, abs/2001.11113, 2020b. URL https://arxiv.org/abs/2001.11113.

Zhang, T., Ren, T., Yang, M., Gonzalez, J., Schuurmans, D., and Dai, B. Making linear mdps practical via contrastive representation learning. In *International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research, 2022.