Stable Policy Optimization via Off-Policy Divergence Regularization

Ahmed Touati[1, *] **Amy Zhang** [2, 3, *] **Joelle Pineau** [2, 3] **Pascal Vincent**[1, 3]
[*] Equal contribution [1] Mila, Université de Montréal [2]Mila, McGill University [3]Facebook AI Research

Abstract

Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO) are among the most successful policy gradient approaches in deep reinforcement learning (RL). While these methods achieve state-of-the-art performance across a wide range of challenging tasks, there is room for improvement in the stabilization of the policy learning and how the off-policy data are used. In this paper we revisit the theoretical foundations of these algorithms and propose a new algorithm which stabilizes the policy improvement through a proximity term that constrains the discounted state-action visitation distribution induced by consecutive policies to be close to one another. This proximity term, expressed in terms of the divergence between the visitation distributions, is learned in an off-policy and adversarial manner. We empirically show that our proposed method can have a beneficial effect on stability and improve final performance in benchmark high-dimensional control tasks.

1 INTRODUCTION

In Reinforcement Learning (RL), an agent interacts with an unknown environment and seeks to learn a policy which maps states to distribution over actions to maximise a long-term numerical reward. Combined with deep neural networks as function approximators, policy gradient methods have enjoyed many empirical successes on RL problems such as video games (Mnih et al., 2016) and robotics (Levine et al., 2016). Their recent success can be attributed to their ability to scale gracefully to high dimensional state-action spaces and complex dynamics.

The main idea behind policy gradient methods is to parametrize the policy and perform stochastic gradient ascent on the discounted cumulative reward directly (Sutton et al., 2000). To estimate the gradient, we sample trajectories from the distribution induced by the policy. Due to the stochasticity of both policy and environment, variance of the gradient estimation can be very large, and lead to significant policy degradation.

Instead of directly optimizing the cumulative rewards, which can be challenging due to large variance, some approaches (Kakade and Langford, 2002; Azar et al., 2012; Pirotta et al., 2013; Schulman et al., 2015) propose to optimize a surrogate objective that can provide local improvements to the current policy at each iteration. The idea is that the advantage function of a policy π can produce a good estimate of the performance of another policy π' when the two policies give rise to similar state visitation distributions. Therefore, these approaches explicitly control the state visitation distribution shift between successive policies.

However, controlling the state visitation distribution shift requires measuring it, which is non-trivial. Direct methods are prohibitively expensive. Therefore, in order to make the optimization tractable, the aforementioned methods rely on constraining action probabilities by mixing policies (Kakade and Langford, 2002; Pirotta et al., 2013), introducing trust regions (Schulman et al., 2015; Achiam et al., 2017) or clipping the surrogate objective (Schulman et al., 2017; Wang et al., 2019b).

Our key motivation in this work is that constraining the probabilities of the immediate future actions might not be enough to ensure that the surrogate objective is still a valid estimate of the performance of the next policy and consequently might lead to instability and premature convergence. Instead, we argue that we should reason about the long-term effect of the policies on the distribution of the future states.

In particular, we directly consider the divergence between state-action visitation distributions induced by succes-

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

sive policies and use it as a regularization term added to the surrogate objective. This regularization term is itself optimized in an adversarial and off-policy manner by leveraging recent advances in off-policy policy evaluation (Nachum et al., 2019a) and off-policy imitation learning (Kostrikov et al., 2019). We incorporate these ideas in the PPO algorithm in order to ensure safer policy learning and better reuse of off-policy data. We call our proposed method PPO-DICE.

The present paper is organized as follows: after reviewing conservative approaches for policy learning, we provide theoretical insights motivating our method. We explain how off-policy adversarial formulation can be derived to optimize the regularization term. We then present the algorithmic details of our proposed method. Finally, we show empirical evidences of the benefits of PPO-DICE as well as ablation studies.

2 PRELIMINARIES

2.1 MARKOV DECISION PROCESSES AND VISITATION DISTRIBUTIONS

In reinforcement learning, an agent interacts with its environment, which we model as a discounted Markov Decision Process (MDP) $(S, A, \gamma, P, r, \rho)$ with state space S, action space A, discount factor $\gamma \in [0, 1)$, transition model P where $P(s' \mid s, a)$ is the probability of transitioning into state s' upon taking action a in state s, reward function $r : (S \times A) \to \mathbb{R}$ and initial distribution ρ over S. We denote by $\pi(a \mid s)$ the probability of choosing action a in state s under the policy π. The value function for policy π, denoted $V^\pi : S \to \mathbb{R}$, represents the expected sum of discounted rewards along the trajectories induced by the policy in the MDP starting at state s: $V^\pi(s) \triangleq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \pi]$. Similarly, the action-value (Q-value) function $Q^\pi : S \times A \to \mathbb{R}$ and the advantage function $A^\pi : S \times A \to \mathbb{R}$ are defined as: $Q^\pi(s, a) \triangleq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid (s_0, a_0) = (s, a), \pi]$ and $A^\pi(s, a) \triangleq Q^\pi(s, a) - V^\pi(s)$. The goal of the agent is to find a policy π that maximizes the expected value from under the initial state distribution ρ:

$$\max_\pi J(\pi) \triangleq (1 - \gamma)\mathbb{E}_{s \sim \rho}[V^\pi(s)].$$

We define the discounted state visitation distribution d^π_ρ induced by a policy π:

$$d^\pi_\rho(s) \triangleq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}^\pi(s_t = s \mid s_0 \sim \rho),$$

where $\mathrm{Pr}^\pi(s_t = s \mid s_0 \sim \rho)$ is the probability that $s_t = s$, after we execute π for t steps, starting from initial state s_0 distributed according to ρ. Similarly, we define the discounted state-action visitation distribution $\mu^\pi_\rho(s, a)$ of policy π

$$\mu^\pi_\rho(s, a) \triangleq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}^\pi(s_t = s, a_t = a \mid s_0 \sim \rho).$$

It is known (Puterman, 1990) that $\mu^\pi_\rho(s, a) = d^\pi_\rho(s) \cdot \pi(a \mid s)$ and that μ^π is characterized via: $\forall (s', a') \in S \times A$ [1]

$$\mu^\pi_\rho(s', a') = (1 - \gamma)\rho(s')\pi(a' \mid s') + \gamma \int \pi(a' \mid s')P(s' \mid s, a)\mu^\pi_\rho(s, a)\, ds\, da, \quad (1)$$

2.2 CONSERVATIVE UPDATE APPROACHES

Most policy training approaches in RL can be understood as updating a current policy π to a new improved policy π' based on the advantage function A^π or an estimate $\hat{A}$ of it. We review here some popular approaches that implement conservative updates in order to stabilize policy training.

First, let us state a key lemma from the seminal work of Kakade and Langford (2002) that relates the performance difference between two policies to the advantage function.

Lemma 2.1 (The performance difference lemma (Kakade and Langford, 2002)). *For all policies π and π',*

$$J(\pi') = J(\pi) + \mathbb{E}_{s \sim d^{\pi'}_\rho}\mathbb{E}_{a \sim \pi'(.\mid s)}[A^\pi(s, a)]. \quad (2)$$

This lemma implies that maximizing Equation (2) will yield a new policy π' with guaranteed performance improvement over a given policy π. Unfortunately, a naive direct application of this procedure would be prohibitively expensive since it requires estimating $d^{\pi'}_\rho$ for all π' candidates. To address this issue, Conservative Policy Iteration (CPI) (Kakade and Langford, 2002) optimizes a surrogate objective defined based on current policy π^i at each iteration i,

$$L_{\pi_i}(\pi') = J(\pi_i) + \mathbb{E}_{s \sim d^{\pi_i}_\rho}\mathbb{E}_{a \sim \pi'(.\mid s)}[A^{\pi_i}(s, a)], \quad (3)$$

by ignoring changes in state visitation distribution due to changes in the policy. Then, CPI returns the stochastic mixture $\pi_{i+1} = \alpha_i \pi_i^+ + (1 - \alpha_i)\pi_i$ where $\pi_i^+ = \arg\max_{\pi'} L_{\pi_i}(\pi')$ is the greedy policy and $\alpha_i \in [0, 1]$ is tuned to guarantee a monotonically increasing sequence of policies.

Inspired by CPI, the Trust Region Policy Optimization algorithm (TRPO) (Schulman et al., 2015) extends the policy improvement step to any general stochastic policy

[1] By abuse of notation, we confound probability distributions with their Radon–Nikodym derivative with respect to the Lebesgue measure (for continuous spaces) or counting measure (for discrete spaces).

rather than just mixture policies. TRPO maximizes the same surrogate objective as CPI subject to a Kullback-Leibler (KL) divergence constraint that ensures the next policy π_{i+1} stays within δ-neighborhood of the current policy π_i:

$$\pi_{i+1} = \arg\max_{\pi'} L_{\pi_i}(\pi') \qquad (4)$$
$$\text{s.t } \mathbb{E}_{s \sim d^{\pi_i}}[D_{KL}(\pi'(\cdot|s)\|\pi_i(\cdot|s))] \le \delta;$$

where D_{KL} is the Kullback–Leibler divergence. In practise, TRPO considers a differentiable parameterized policy $\{\pi_\theta; \theta \in \Theta\}$ and solves the constrained problem (4) in parameter space. In particular, the step direction is estimated with conjugate gradients, which requires the computation of multiple Hessian-vector products. Therefore, this step can be computationally heavy.

To address this computational bottleneck, Proximal Policy Optimization (PPO) (Schulman et al., 2017) proposes replacing the KL divergence constrained objective (4) of TRPO by clipping the objective function directly as:

$$L^{clip}_{\pi_i}(\pi') = \mathbb{E}_{(s,a)\sim\pi_i}\Big[\min\Big(A^{\pi_i}(s,a)\,\rho_{\pi'/\pi_i}(s,a),$$
$$A^{\pi_i}(s,a)\,\text{clip}(\rho_{\pi'/\pi_i}(s,a), 1-\epsilon, 1+\epsilon)\Big)\Big]; \qquad (5)$$

where $\epsilon > 0$ and $\rho_{\pi'/\pi_i}(s,a) = \frac{\pi'(s,a)}{\pi_i(s,a)}$ is the importance sampling ratio.

3 THEORETICAL INSIGHTS

In this section, we present the theoretical motivation of our proposed method.

At a high level, algorithms CPI, TRPO, and PPO follow similar policy update schemes. They optimize some surrogate performance objective ($L_{\pi_i}(\pi')$ for CPI and TRPO and $L^{clip}_{\pi_i}(\pi')$ for PPO) while ensuring that the new policy π_{i+1} stays in the vicinity of the current policy π_i. The vicinity requirement is implemented in different ways:

1. CPI computes a sequence of stochastic policies that are mixtures between consecutive greedy policies.
2. TRPO imposes a constraint on the KL divergence between old policy and new one ($\mathbb{E}_{s\sim d^{\pi_i}}[D_{KL}(\pi'(\cdot|s)\|\pi_i(\cdot|s))] \le \delta$).
3. PPO directly clips the objective function based on the value of the importance sampling ratio ρ_{π'/π_i} between the old policy and new one.

Such conservative updates are critical for the stability of the policy optimization. In fact, the surrogate objective $L_{\pi_i}(\pi')$ (or its clipped version) is valid only in the neighbourhood of the current policy π_i, i.e, when π' and π_i visit all the states with similar probabilities. The following lemma more precisely formalizes this[2].

Lemma 3.1. For all policies π and π',

$$J(\pi') \ge L_\pi(\pi') - \epsilon^\pi D_{TV}(d^{\pi'}\|d^\pi) \qquad (6)$$
$$\ge L^{clip}_\pi(\pi') - \epsilon^\pi D_{TV}(d^{\pi'}\|d^\pi);$$

where $\epsilon^\pi = \max_{s\in S}|\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^\pi(s,a)]|$ and D_{TV} is the total variation distance.

The proof is provided in appendix for completeness. Lemma 3.1 states that $L_\pi(\pi')$ (or $L^{clip}_\pi(\pi')$) is a sensible lower bound to $J(\pi')$ as long as π and π' are close in terms of total variation distance between their corresponding state visitation distributions $d^{\pi'}$ and d^π. However, the aforementioned approaches enforce closeness of π' and π in terms of their action probabilities rather than their state visitation distributions. This can be justified by the following inequality (Achiam et al., 2017):

$$D_{TV}(d^{\pi'}\|d^\pi) \le \frac{2\gamma}{1-\gamma}\mathbb{E}_{s\sim d^\pi}[D_{TV}(\pi'(\cdot|s)\|\pi(\cdot|s))]. \qquad (7)$$

Plugging the last inequality (7) into (6) leads to the following lower bound:

$$J(\pi') \ge L_\pi(\pi') - \epsilon^\pi\frac{2\gamma}{1-\gamma}\mathbb{E}_{s\sim d^\pi}[D_{TV}(\pi'(\cdot|s)\|\pi(\cdot|s))]. \qquad (8)$$

The obtained lower bound (8) is, however, clearly looser than the one in inequality (7). Lower bound (8) suffers from an additional multiplicative factor $\frac{1}{1-\gamma}$, which is the effective planning horizon. It is essentially due to the fact that we are characterizing a long-horizon quantity, such as the state visitation distribution $d^\pi(s)$, by a one-step quantity, such as the action probabilities $\pi(\cdot|s)$. Therefore, algorithms that rely solely on action probabilities to define closeness between policies should be expected to suffer from instability and premature convergence in long-horizon problems.

Furthermore, in the exact case if we take at iteration i, $\pi_{i+1} \leftarrow \arg\max_{\pi'} L_{\pi_i}(\pi') - \epsilon^{\pi_i} D_{TV}(d^{\pi'}\|d^{\pi_i})$, then

$$\begin{aligned} J(\pi_{i+1}) &\ge L_{\pi_i}(\pi_{i+1}) - \epsilon^{\pi_i} D_{TV}(d^{\pi_{i+1}}\|d^{\pi_i}) \\ &\ge L_{\pi_i}(\pi_i) \qquad \text{(by optimality of } \pi_{i+1}) \\ &= J(\pi_i) \end{aligned}$$

Therefore, this provides a monotonic policy improvement, while TRPO suffers from a performance degradation that depends on the level of the trust region δ (see Proposition 1 in Achiam et al. (2017)).

[2] The result is not novel, it can be found as intermediate step in proof of theorem 1 in Achiam et al. (2017), for example.

It follows from our discussion that $D_{TV}(d_{\pi'}\|d_\pi)$ is a more natural proximity term to ensure safer and more stable policy updates. Previous approaches excluded using this term because we don't have access to $d_{\pi'}$, which would require executing π' in the environment. In the next section, we show how we can leverage recent advances in off-policy policy evaluation to address this issue.

4 OFF-POLICY FORMULATION OF DIVERGENCES

In this section, we explain how divergences between state-visitation distributions can be approximated. This is done by leveraging ideas from recent works on off-policy learning (Nachum et al., 2019a; Kostrikov et al., 2019).

Consider two different policies π and π'. Suppose that we have access to state-action samples generated by executing the policy π in the environment, i.e, $(s,a) \sim \mu_\pi$. As motivated by the last section, we aim to estimate $D_{TV}(d_{\pi'}\|d_\pi)$ without requiring on-policy data from π'. Note that in order to avoid using importance sampling ratios, it is more convenient to estimate $D_{TV}(\mu_{\pi'}\|\mu_\pi)$, i.e, the total divergence between state-action visitation distributions rather than the divergence between state visitation distributions. This is still a reasonable choice as $D_{TV}(d_{\pi'}\|d_\pi)$ is upper bounded by $D_{TV}(\mu_{\pi'}\|\mu_\pi)$ as shown below:

$$\begin{aligned} D_{TV}(d_{\pi'}\|d_\pi) &= \int_s \left|(d_{\pi'} - d_\pi)(s)\right| ds \\ &= \int_s \left|\int_a (\mu_{\pi'} - \mu_\pi)(s,a)\, da\right| ds \\ &\le \int_s \int_a \left|(\mu_{\pi'} - \mu_\pi)(s,a)\right| da\, ds \\ &= D_{TV}(\mu_{\pi'}\|\mu_\pi). \end{aligned}$$

The total variation distance belongs to a broad class of divergences known as ϕ-divergences (Sriperumbudur et al., 2009). A ϕ-divergence is defined as,

$$D_\phi(\mu_{\pi'}\|\mu_\pi) = \mathbb{E}_{(s,a)\sim\mu_{\pi'}}\left[\phi\left(\frac{\mu_\pi(s,a)}{\mu_{\pi'}(s,a)}\right)\right], \quad (9)$$

where $\phi : [0,\infty) \to \mathbb{R}$ is a convex, lower-semicontinuous function and $\phi(1) = 0$. Well-known divergences can be obtained by appropriately choosing ϕ. These include the KL divergence ($\phi(t) = t\log(t)$), total variation distance ($\phi(t) = |t-1|$), χ^2-divergence ($\phi(t) = (t-1)^2$), etc. Working with the form of ϕ-divergence given in Equation (9) requires access to analytic expressions of both μ_π and $\mu_{\pi'}$ as well as the ability to sample from $\mu_{\pi'}$. We have none of these in our problem of interest. To bypass these difficulties, we turn to the alternative variational representation of ϕ-divergences (Nguyen et al., 2009; Huang et al., 2017) as

$$D_\phi(\mu_{\pi'}\|\mu_\pi) = \sup_{f:S\times A\to\mathbb{R}} \Big[\mathbb{E}_{(s,a)\sim\mu_{\pi'}}[f(s,a)] - \mathbb{E}_{(s,a)\sim\mu_\pi}[\phi^\star(f(s,a))]\Big], \quad (10)$$

where $\phi^\star(t) = \sup_{u\in\mathbb{R}}\{tu - \phi(u)\}$ is the convex conjugate of ϕ. The variational form in Equation (10) still requires sampling from $\mu_{\pi'}$, which we cannot do. To address this issue, we use a clever change of variable trick introduced by Nachum et al. (2019a). Define $g : S\times A \to \mathbb{R}$ as the fixed point of the following Bellman equation,

$$g(s,a) = f(s,a) + \gamma P^{\pi'} g(s,a), \quad (11)$$

where $P^{\pi'}$ is the transition operator induced by π', defined as $P^{\pi'} g(s,a) = \int \pi'(a'|s')P(s'|s,a)g(s',a')$. g may be interpreted as the action-value function of the policy π' in a modified MDP which shares the same transition model P as the original MDP, but has f as the reward function instead of r. Applying the change of variable (11) to (10) and after some algebraic manipulation as done in Nachum et al. (2019a), we obtain

$$D_\phi(\mu_{\pi'}\|\mu_\pi) = \sup_{g:S\times A\to\mathbb{R}} \Big[(1-\gamma)\mathbb{E}_{s\sim\mu_0, a\sim\pi'}[g(s,a)] - \mathbb{E}_{(s,a)\sim\mu_\pi}\Big[\phi^\star\big((g - \gamma P^{\pi'} g)(s,a)\big)\Big]\Big]. \quad (12)$$

Thanks to the change of variable, the first expectation over $\mu_{\pi'}$ in (10) is converted to an expectation over the initial distribution and the policy i.e $s \sim \mu_0(\cdot), a \sim \pi'(\cdot|s)$. Therefore, this new form of the ϕ-divergence in (12) is completely off-policy and can be estimated using only samples from the policy π.

Other possible divergence representations: Using the variational representation of ϕ-divergences was a key step in the derivation of Equation (12). But in fact any representation that admits a linear term with respect to $\mu_{\pi'}$ (i.e $\mathbb{E}_{(s,a)\sim\mu_{\pi'}}[f(s,a)]$) would work as well. For example, one can use the the Donkser-Varadhan representation (Donsker and Varadhan, 1983) to alternatively express the KL divergence as:

$$D_{KL}(\mu_{\pi'}\|\mu_\pi) = \sup_{f:S\times A\to\mathbb{R}} \Big[\mathbb{E}_{(s,a)\sim\mu_{\pi'}}[f(s,a)] - \log \mathbb{E}_{(s,a)\sim\mu_\pi}[\exp(f(s,a))]\Big]. \quad (13)$$

The log-expected-exp in this equation makes the Donkser-Varadhan representation (13) more numerically stable than the variational one (12) when working with KL divergences. Because of its genericity for ϕ-divergences, we base the remainder of our exposition on (12). But it is straightforward to adapt the approach and algorithm to using (13) for better numerical stability when working with KL divergences specifically. Thus, in practice we will use the latter in our experiments with KL-based regularization, but not in the ones with ^{2}-based regularization.

5 A PRACTICAL ALGORITHM USING ADVERSARIAL DIVERGENCE

We now turn these insights into a practical algorithm. The lower bounds in lemma 3.1, suggest using a regularized PPO objective[3]: $L^{clip}(\ ^0) \quad D_{TV}(d\ ^0 \| d\ )$, where is a regularization coefficient. If in place of the total variation we use the off-policy formulation of -divergence $D(\ ^0 \|\ )$ as detailed in Equation (12), our main optimization objective can be expressed as the following min-max problem:

$$\max_{^0} \min_{g:S\ A \to R} L^{clip}_{i}(\ ^0) \quad (1\quad)E_{s\ _i ;a\ ^0}[g(s;a)] \quad E_{(s;a)\ _i}\ ^?(g \quad P\ ^0 g)(s;a) \quad ; \tag{14}$$

When the inner minimization over g is fully optimized, it is straightforward to show – using the score function estimator – that the gradient of this objective with respect to is (proof is provided in appendix):

$$r_{\ ^0} L^{clip}_{i}(\ ^0) \quad (1\quad)E_{s\ ,a\ ^0}[g(s;a) r_{\ ^0} \log\ ^0(a\ j\ s)] + E_{(s;a)\ _i}\frac{@\ ^?}{@t}(g \quad P\ ^0 g)(s;a) \tag{15}$$

$$E_{s^0\ P(\ j s;a);a^0\ ^0(\ j s^0)}[g(s^0;a^0) r_{\ ^0} \log\ ^0(a^0 j\ s^0)] \quad :$$

Furthermore, we can use the reparametrization trick if the policy is parametrized by a Gaussian, which is usually the case in continuous control tasks. We call the resulting new algorithm PPO-DICE, (detailed in Algorithm 1), as it uses the clipped loss of PPO and leverages the Distribution Correction Estimation idea from Nachum et al. (2019a).

In the min-max objective (14), g plays the role of a discriminator, as in Generative Adversarial Networks (GAN) (Goodfellow et al., 2014). The policy ^{0} plays the role of a generator, and it should balance between increasing the likelihood of actions with large advantage versus inducing a state-action distribution that is close to the one of $_i$.

As shown in Algorithm 1, both policy and discriminator are parametrized by neural networks and g respectively. We estimate the objective (14) with samples from $_i = \ _i$ as follows. At a given iteration i, we generate a batch of M rollouts $f s_1^{(j)}; a_1^{(j)}; r_1^{(j)}; s_1^{(j)}; :::; s_T^{(j)}; a_T^{(j)}; r_T^{(j)}; s_{T+1}^{(j)} g_{j=1}^M$ by executing the policy $_i$ in the environment for T steps. Similarly to the PPO procedure, we learn a value function $V_!$ by updating its parameters ! with gradient descent steps, optimizing the following squared error loss:

$$\hat{L}_V(!) = \frac{1}{MT}\sum_{j=1}^{M}\sum_{t=1}^{T} V_!(s_t^{(j)}) \quad y_t^{(j)}\ ^2 ; \tag{16}$$

where $y_t^{(j)} = r_t^{(j)} + \ r_{t+1}^{(j)} + ::: + \ ^{T+1\ t} V_!(s_{T+1})$. Then, to estimate the advantage, we use the truncated generalized advantage estimate

$$\hat{A}(s_t^{(j)};a_t^{(j)}) = \sum_{t=1}^{T}(\quad)^{t\ 1}(r_t^{(j)} + \ V_!(s_{t+1}^{(j)}) \quad V_!(s_t^{(j)})): \tag{17}$$

This advantage estimate is used to compute an estimate of L^{clip}_{i} given by:

$$\hat{L}^{clip}(\ ) = \tag{18}$$

$$\frac{1}{MT}\sum_{j=1}^{M}\sum_{t=1}^{T} \min \Big\{ \hat{A}(s_t^{(j)};a_t^{(j)}) \ \frac{\ }{\ _i}(s_t^{(j)};a_t^{(j)}); \quad \hat{A}(s_t^{(j)};a_t^{(j)}) \ \text{clip}(\ \frac{\ }{\ _i}(s_t^{(j)};a_t^{(j)}); 1 \quad ; 1+\ ) \Big\}$$

The parameters of the discriminator are learned by gradient descent on the following empirical version of the regularization term in the min-max objective (14)

$$\hat{L}_D(\ ;\ ) = \frac{1}{MT}\sum_{j=1}^{M}\sum_{t=1}^{T}(1\quad)g\ (s_1^{(j)};a_t^{0(j)}) \tag{19}$$

$$^?\ g\ (s_t^{(j)};a_t^{(j)}) \quad g\ (s_{t+1}^{(j)};a_{t+1}^{0(j)}) \quad ;$$

where $a_t^{0(j)} \quad (\ j\ s_1^{(j)})$ and $a_{t+1}^{0(j)} \quad (\ j\ s_{t+1}^{(j)})$.

If the reparametrization trick is applicable (which is almost always the case for continuous control tasks), the parameters of the policy are updated via gradient ascent on the objective $\hat{L}^{clip}(\ ) + \ \hat{L}_D(\ ;\ )$ as we can backpropagate gradient though the action sampling while computing $\hat{L}_D(\ ;\ )$ in Equation (19). Otherwise, are

[3] Both regularized L^{clip}_{i} and $L_{\ _i}$ are lower bounds on policy performance in Lemma 3.1. We use L^{clip}_{i} rather than $L_{\ _i}$ because we expect it to work better as the clipping already provides some constraint on action probabilities. Also this will allow a more direct empirical assessment of what the regularization brings compared to vanilla PPO.

Algorithm 1 PPO-DICE

```
Initialisation : random initialize parameters $_1$ (policy), $_1$ (discriminator) and $!_1$ (value function).
for i=1, . . . do
    Generate a batch of M rollouts $f\, s_1^{(j)}; a_1^{(j)}; r_1^{(j)}; s_1^{(j)}; :::; s_T^{(j)}; a_T^{(j)}; r_T^{(j)}; s_{T+1}^{(j)}\, g_{j=1}^{M}$ by executing policy $_i$ in the environment for T steps.
    Estimate Advantage function: $\hat{A}(s_t^{(j)}; a_t^{(j)}) = \text{P}_{t=1}^{T} ( \quad )^{t \quad 1}(r_t^{(j)} + V_{!_i}(s_{t+1}^{(j)}) \quad V_{!_i}(s_t^{(j)}))$
    Compute target value $y_t^{(j)} = r_t^{(j)} + r_{t+1}^{(j)} + ::: + ^{T+1 \quad t} V_{!_i}(s_{T+1})$
    $! = !_i; \quad = \quad _i; \quad = \quad _i$
    for epoch n=1, . . . N do
        for iteration k=1, . . . K do
            // Compute discriminator loss:
            $\hat{L}_D( ; ) = \frac{1}{MT} \text{P}_{j=1}^{M} \text{P}_{t=1}^{T} \ ^{?} \ g (s_t^{(j)}; a_t^{(j)}) \quad g (s_{t+1}^{(j)}; a_{t+1}^{0(j)}) \quad (1 \quad ) g (s_1^{(j)}; a_t^{0(j)})$ where $a_t^{0(j)} \quad ( \ j \ s_1^{(j)}); a_{t+1}^{0(j)} \quad ( \ j \ s_{t+1}^{(j)})$.
            // Update discriminator parameters: (using learning rate c )
                $c \quad r \ \hat{L}_D( ; );$
        end for
        // Compute value loss:
        $\hat{L}_V(!) = \frac{1}{MT} \text{P}_{j=1}^{M} \text{P}_{t=1}^{T} \ V_! (s_t^{(j)}) \quad y_t^{(j)} \ ^2$
        // Compute PPO clipped loss:
        $\hat{L}^{clip}( ) = \frac{1}{MT} \text{P}_{j=1}^{M} \text{P}_{t=1}^{T} \min \ \hat{A}(s_t^{(j)}; a_t^{(j)}) \ _{=} \ _i(s_t^{(j)}; a_t^{(j)}); \hat{A}(s_t^{(j)}; a_t^{(j)})\text{clip}( \ _{=} \ _i (s_t^{(j)}; a_t^{(j)}); 1 \ ; 1 + )$
        // Update parameters: (using learning rate)
        $! \quad ! \quad r_! \hat{L}_V(!);$
        $\quad + \ r \ (\hat{L}^{clip}( ) + \ \hat{L}_D( ; ))$ (if reparametrization trick applicable, else gradient step on Eq. 20)
    end for
    $!_{i+1} = !; \quad _{i+1} = \ ; \quad _{i+1} = $
end for
```

updated via gradient ascent on the following objective:

$$\hat{L}^{clip}( ) \qquad \frac{}{MT} \sum_{j=1}^{M} \sum_{t=1}^{T} (1 \quad) g (s_1^{(j)}; a_t^{0(j)}) \log \ (a_t^{0(j)} \ j \ s_1^{(j)}) + \frac{@^{?}}{@t} \ g (s_t^{(j)}; a_t^{(j)}) \quad g (s_{t+1}^{(j)}; a_{t+1}^{0(j)}) \quad g (s_{t+1}^{(j)}; a_{t+1}^{0(j)}) \log \ (a_{t+1}^{0(j)}) \ j \ s_{t+1}^{(j)}) \tag{20}$$

Note that the gradient of this equation with respect to corresponds to an empirical estimate of the score function estimator we provided in Equation 15.

We train the value function, policy, and discriminator for N epochs using M rollouts of the policy $_i$. We can either alternate between updating the policy and the discriminator, or update g for a few steps M before updating the policy. We found that the latter worked better in practice, likely due to the fact that the target distribution $_i$ changes with every iteration. We also found that increasing the learning rate of the discriminator by a multiplicative factor c of the learning rate for the policy and value function improved performance.

Choice of divergence: The algorithmic approach we just described is valid with any choice of f-divergence for measuring the discrepancy between state-visitation distributions. It remains to choose an appropriate one. While Lemma 3.1 advocates the use of total variation distance ((t) = jt 1j), it is notoriously hard to train high dimensional distributions using this divergence (see Kodali et al. (2017) for example). Moreover, the convex conjugate of $(t) = jt \quad 1j$ is $^{?}(t) = t$ if $jtj \quad \frac{1}{2}$ and $^{?}(t) = 1$ otherwise. This would imply the need to introduce an extra constraint $kg \quad \text{P} \ gk_1 \quad \frac{1}{2}$ in the formulation (12), which may be hard to optimize.

Therefore, we will instead use the KL divergence ((t) = t log(t); $^{?}(t) = \exp(t \quad 1)$). This is still a well justified choice as we know that $D_{TV}(\quad ^0k \quad) \quad \sqrt{\frac{1}{2} D_{KL}(\quad ^0k \quad)}$ thanks to Pinsker's inequality. We will also try ^{2}-divergence ((t) = $(t \quad 1)^2$) that yields a squared regularization term.

6 RELATED WORK

Constraining policy updates, in order to minimize the information loss due to policy improvement, has been an active area of investigation. Kakade and Langford (2002) originally introduce CPI by maximizing a lower bound on the policy improvement and relaxing the greedification step through a mixture of successive policies. Pirotta et al. (2013) build on Kakade and Langford (2002) refine the lower bounds and introduce a new mixture scheme. Moreover, CPI inspired some popular Deep RL algorithms such as TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2015), Deep CPI (Vieillard et al., 2019) and MPO (Abdolmaleki et al., 2018). The latter uses similar updates to TRPO/PPO in the parametric version of its E-step. So, our method can be incorporated to it.

Our work is related to regularized MDP literature (Neu et al., 2017; Geist et al., 2019). Shannon Entropic regularization is used in value iteration scheme (Haarnoja et al., 2017; Dai et al., 2018) and in policy iteration schemes (Haarnoja et al., 2018). Note that all the mentioned works employ regularization on the action probabilities. Recently, Wang et al. (2019a) introduce divergence-augmented policy optimization where they penalize the policy update by a Bregman divergence on the state visitation distributions, motivated the mirror descent method. While their framework seems general, it doesn't include the divergences we employ in our algorithm. In fact, their method enables the use of the conditional KL divergence between state-action visitations distribution defined by $\int (s;a) \log \frac{(a|s)}{{}^0(a|a)}$ and not the KL divergence $\int (s;a) \log \frac{(s;a)}{{}^0(s;a)}$. Note the action probabilities ratio inside the log in the conditional KL divergence allows them to use the policy gradient theorem, a key ingredient in their framework, which cannot be done for the KL divergence.

Our work builds on recent off-policy approaches: DualDICE (Nachum et al., 2019a) for policy evaluation and ValueDICE (Kostrikov et al., 2019) for imitation learning. Both use the off-policy formulation of KL divergence. The former uses the formulation to estimate the ratio of the state visitation distributions under the target and behavior policies. Whereas, the latter learns a policy by minimizing the divergence.

The closest related work is the recently proposed AlgaeDICE (Nachum et al., 2019b) for off-policy policy optimization. They use the divergence between state-action visitation distribution induced by and a behavior distribution, motivated by similar techniques in Nachum et al. (2019a). However, they incorporate the regularization to the dual form of policy performance $J( ) = E_{(s;a)} [r(s;a)]$ whereas we consider a surrogate objective (lower bound on the policy performance). Moreover, our method is online off-policy in that we collect data with each policy found in the optimization procedure, but also use previous data to improve stability. Whereas, their algorithm is designed to learn a policy from a fixed dataset collected by behaviour policies. Further comparison with AlgaeDICE is provided in appendix.

Figure 1: Comparison of ^{2} and KL divergences for PPO-DICE for two randomly selected environments in OpenAI Gym MuJoCo and Atari, respectively. We see that KL performs better than ^{2} in both settings. Performance plotted across 10 seeds with 1 standard error shaded.

7 EXPERIMENTS AND RESULTS

We use the PPO implementation by Kostrikov (2018) as a baseline and modify it to implement our proposed PPO-DICE algorithm. We run experiments on a randomly selected subset of environments in the Atari suite (Bellemare et al., 2013) for high-dimensional observations and discrete action spaces, as well as on the OpenAI Gym (Brockman et al., 2016) MuJoCo environments, which have continuous state-action spaces. All shared hyperparameters are set at the same values for both methods, and we use the hyperparameter values recommended by Kostrikov (2018) for each set of environments, Atari and MuJoCo[4].

7.1 IMPORTANT ASPECTS OF PPO-DICE

7.1.1 Choice of Divergence

We conducted an initial set of experiments to compare two different choices of divergences, KL and ^{2}, for the regularization term of PPO-DICE. Figure 1 shows training curves for one continuous action and one discrete action environment. There, as in the other environments in which we run this comparison, KL consistently performed better than ^{2}. We thus opted to use KL divergence in all subsequent experiments.

[4] Code: https://github.com/facebookresearch/ppo-dice

7.1.2 Effect of Varying

Figure 2: Varying in Hopper_v2 , 10 seeds, 1 standard error shaded. PPO-DICE is somewhat sensitive to value, but the theoretically-motivated adaptive version works well.

Next we wanted to evaluate the sensitivity of our method to the parameter that controls the strength of the regularization. We examine in Figure 2 the performance of PPO-DICE when varying . There is a fairly narrow band for Hopper-v2 that performs well, between 0 01 and 1. Theory indicates that the proper value for is the maximum of the absolute value of the advantages (see Lemma 3.1). This prompted us to implement an adaptive approach, where we compute the 90th percentile of advantages within the batch (for stability), which we found performed well across environments. To avoid introducing an additional hyperparameter by tuning , we use the adaptive method for subsequent experiments.

Figure 3: Comparison of PPO-DICE with clipped loss L^{clip} and without L. We see that clipping the action loss is crucial for good performance.

7.1.3 Importance of Clipping the Action Loss

We earlier mentioned (see Footnote 3) two possible forms of our regularized objective: one with clipped action loss L^{clip} and one without L. Clipping the action loss was an extra regularizing measure proposed in PPO (Schulman et al., 2017). For our algorithm also, we hypothesized that it would be important for providing additional constraints on the policy update to stay within the trust region. Figure 3 confirms this empirically: we see the effect on our method of clipping the action loss versus keeping it unclipped. Initially, not having the additional regularization allows it to learn faster, but it soon crashes, showing the need for clipping to reduce variance in the policy update.

7.2 RESULTS ON ATARI

Given our above observations we settled on using a KL-regularized L^{clip}, with the adaptive method for that we explained Section 7.1.2. We run PPO-DICE on randomly selected environments from Atari. We tuned two additional hyperparameters, the learning rate for the discriminator and the number of discriminator optimization steps per policy optimization step. We found that $K = 5$ discriminator optimization steps per policy optimization step performed well. Fewer steps showed worse performance because the discriminator was not updating quickly enough, while more optimization steps introduced instability from the discriminator overfitting to the current batch. We also found that increasing the discriminator learning rate to be $c = 10$ the policy learning rate helped most environments. We used the same hyperparameters across all environments. Results are shown in Table 1. We see that PPO-DICE significantly outperforms PPO on a majority of Atari environments. See Appendix C.2 for training curves and hyperparameters.

7.3 RESULTS ON OpenAI Gym MuJoCo

For the OpenAI Gym MuJoCo suite, we also used $K = 5$ discriminator optimization steps per policy optimization step, and $c = 10$ learning rate for the discriminator in all environments. We selected 5 of the more difficult environments to showcase in the main paper (Figure 4), but additional results on the full suite and all hyperparameters used can be found in Appendix C.1. We again see improvement in performance in the majority of environments with PPO-DICE compared to PPO and TRPO.

8 CONCLUSION

In this work, we have argued that using the action probabilities to constrain the policy update is a suboptimal approximation to controlling the state visitation distribution shift. We then demonstrate that using the recently proposed DIstribution Correction Estimation idea (Nachum et al., 2019a), we can directly compute the divergence between the state-action visitation distributions of successive policies and use that to regularize the policy opti-

Game	PPO	PPO-DICE
AirRaid	4305:0 638:15	5217:5 769:19
Asterix	4300:0 169:31	6200:0 754:10
Asteroids	1511:0 125:03	1653:0 112:20
Atlantis	2120400:0 471609:93	3447433:33 100105:82
BankHeist	1247:0 21:36	1273:33 7:89
BattleZone	29000:0 2620:43	19000:0 2463:06
Carnival	3243:33 369:51	3080:0 189:81
ChopperCommand	566:67 14:91	900:0 77:46
DoubleDunk	6:0 1:62	4:0 1:26
Enduro	1129:9 73:18	1308:33 120:09
Freeway	32:33 0:15	32:0 0:00
Frostbite	639:0 334:28	296:67 5:96
Gopher	1388:0 387:65	1414:0 417:84
Kangaroo	4060:0 539:30	6650:0 1558:16
Phoenix	12614:0 621:71	11676:67 588:24
Robotank	7:8 1:33	12:1 2:91
Seaquest	1198:0 128:82	1300:0 123:97
TimePilot	5070:0 580:53	7000:0 562:32
Zaxxon	7110:0 841:60	6130:0 1112:48

Table 1: Mean final reward and 1 standard error intervals across 10 seeds for Atari games evaluated at 10M steps.

Figure 4: Results from OpenAI Gym MuJoCo suite in more complex domains, with 10 seeds and 1 standard error shaded. Results on the full suite of environments can be found in Appendix C.1.

mization objective instead. Through carefully designed experiments, we have shown that our method beats PPO in most environments in Atari (Bellemare et al., 2013) and OpenAI Gym MuJoCo (Brockman et al., 2016) benchmarks.

9 Acknowledgements

We would like to thank Ofir Nachum and Ilya Kostrikov for their helpful feedback and advice during discussions at the early stage of the project.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. A. (2018). Maximum a posteriori policy optimisation. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 22–31. JMLR. org.

Azar, M. G., Gómez, V., and Kappen, H. J. (2012). Dynamic policy programming. Journal of Machine Learning Research, 13(Nov):3207–3245.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. J. Artif. Int. Res., 47(1):253–279.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., and Song, L. (2018). Sbeed: Convergent reinforcement learning with nonlinear function approximation. In *International Conference on Machine Learning*, pages 1125–1134.

Donsker, M. D. and Varadhan, S. S. (1983). Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212.

Geist, M., Scherrer, B., and Pietquin, O. (2019). A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870.

Huang, G., Berard, H., Touati, A., Gidel, G., Vincent, P., and Lacoste-Julien, S. (2017). Parametric adversarial divergences are good task losses for generative modeling. *arXiv preprint arXiv:1708.02511*.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*.

Kostrikov, I. (2018). Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail.

Kostrikov, I., Nachum, O., and Tompson, J. (2019). Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Nachum, O., Chow, Y., Dai, B., and Li, L. (2019a). Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, pages 2315–2325.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. (2019b). Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*.

Neu, G., Jonsson, A., and Gómez, V. (2017). A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*.

Nguyen, X., Wainwright, M. J., Jordan, M. I., et al. (2009). On surrogate loss functions and f-divergences. *The Annals of Statistics*, 37(2):876–904.

Pirotta, M., Restelli, M., Pecorino, A., and Calandriello, D. (2013). Safe policy iteration. In *International Conference on Machine Learning*, pages 307–315.

Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., and Lanckriet, G. R. (2009). On integral probability metrics, -divergences and binary classification. *arXiv preprint arXiv:0901.2698*.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Vieillard, N., Pietquin, O., and Geist, M. (2019). Deep conservative policy iteration. *arXiv preprint arXiv:1906.09784*.

Wang, Q., Li, Y., Xiong, J., and Zhang, T. (2019a). Divergence-augmented policy optimization. In *Advances in Neural Information Processing Systems*, pages 6097–6108.

Wang, Y., He, H., and Tan, X. (2019b). Truly proximal policy optimization. *arXiv preprint arXiv:1903.07940*.

A Omitted Proofs

A.1 Proof of Lemma 3.1

According to performance difference lemma 2.1, we have

$$
\begin{aligned}
J(\pi') &= J(\pi) + \mathbb{E}_{s\sim d^{\pi'}}\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] \\
&= J(\pi) + \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] + \int_{s\in S}\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)]\,(d^{\pi'}(s) - d^{\pi}(s))\,ds \\
&\geq J(\pi) + \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] - \int_{s\in S}|\mathbb{E}_{a\sim\pi(\cdot|s)}[A^{\pi}(s,a)]|\,|d^{\pi'}(s) - d^{\pi}(s)|\,ds \\
&\geq J(\pi) + \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] - \epsilon\int_{s\in S}|d^{\pi'}(s) - d^{\pi}(s)|\,ds \\
&\geq J(\pi) + \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] - \epsilon D_{TV}(d^{\pi'}\|d^{\pi}) \\
&= L_{\pi}(\pi') - \epsilon D_{TV}(d^{\pi'}\|d^{\pi})
\end{aligned}
$$

where $\epsilon = \max_s|\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)]|$ and D_{TV} is total variation distance. The first inequality follows from Cauchy-Schwartz inequality.

A.2 Score Function Estimator of the gradient with respect to the policy

$$
\nabla_{\theta'}\mathbb{E}_{\substack{s\sim\rho\\a\sim\pi_{\theta'}}}[g(s,a)] = \nabla_{\theta'}\int g(s,a)\,\rho(s)\,\pi_{\theta'}(a|s) = \int g(s,a)\,\rho(s)\nabla_{\theta'}\pi_{\theta'}(a|s) = \mathbb{E}_{\substack{s\sim\rho\\a\sim\pi_{\theta'}}}[g(s,a)\nabla_{\theta'}\log\pi_{\theta'}(a|s)]
$$

$$
\begin{aligned}
\nabla_{\theta'}\mathbb{E}_{(s,a)\sim\rho_i}\phi^{?}(g - P^{\pi'}g)(s,a) &= \mathbb{E}_{(s,a)\sim\rho_i}\nabla_{\theta'}\phi^{?}(g - P^{\pi'}g)(s,a) \\
&= \mathbb{E}_{(s,a)\sim\rho_i}\frac{\partial\phi^{?}}{\partial t}(g - P^{\pi'}g)(s,a)\,\nabla_{\theta'}(g - P^{\pi'}g) \\
&= -\mathbb{E}_{(s,a)\sim\rho_i}\frac{\partial\phi^{?}}{\partial t}(g - P^{\pi'}g)(s,a)\,\nabla_{\theta'}\int g(s',a')P(s'|s,a)\,\pi'(a'|s') \\
&= -\mathbb{E}_{(s,a)\sim\rho_i}\frac{\partial\phi^{?}}{\partial t}(g - P^{\pi'}g)(s,a)\,\mathbb{E}_{\substack{s'\sim P(\cdot|s,a)\\a'\sim\pi'(\cdot|s')}}[g(s',a')\nabla_{\theta'}\log\pi'(a'|s')]
\end{aligned}
$$

B Comparison with AlgaeDICE

Both the recent AlgaeDICE (Nachum et al., 2019b) and our present work propose regularisation based on discounted state-action visitation distribution but in different ways. Firstly, AlgaeDICE is initially designed to find an optimal policy given a batch of training data. They alter the objective function itself i.e the policy performance $J(\pi)$ by adding the divergence between the discounted state-action visitation distribution and training distribution, while our approach adds the divergence term to $L_{\pi}(\pi')$. The latter is a first order Taylor approximation of the policy performance $J(\pi')$. Therefore, our approach could be seen as a mirror descent that uses the divergence as a proximity term. Secondly, their training objective is completely different from ours. Their method ends up being an off-policy version of the actor-critic method.

We implemented the AlgaeDICE min-max objective to replace our surrogate min-max objective in the PPO training procedure i.e at each iteration, we sample rollouts from the current policy and update the actor and the critic of AlgaeDICE for 10 epochs. Empirically, we observed that AlgaeDICE objective is very slow to train in this setting. This was expected as it is agnostic to training data while our method leverages the fact that the data is produced by the current policy and estimates advantage using on-policy multi-step Monte Carlo. So our approach is more suitable than AlgaeDICE in this setting. However, AlgaeDICE, as an off-policy method, would be better when storing all history of transitions and updating both actor and critic after each transition, as shown in Nachum et al. (2019b).

C Empirical Results

C.1 OpenAI Gym: MuJoCo

See Figure 5

C.2 Atari

See Figure 6

D Hyperparameters

D.1 OpenAI Gym: MuJoCo

For the OpenAI Gym environments we use the default hyperparameters in Kostrikov (2018).

Parameter name	Value
Number of minibatches	4
Discount	0:99
Optimizer	Adam
Learning rate	3e-4
PPO clip parameter	0:2
PPO epochs	10
GAE	0:95
Entropy coef	0
Value loss coef	0:5
Number of forward steps per update	2048

Table 2: A complete overview of used hyper parameters for all methods.

D.2 Atari

For the Atari hyperparameters, we again use the defaults set by Kostrikov (2018).

Parameter name	Value
Number of minibatches	4
Discount	0:99
Optimizer	Adam
Learning rate	2.5e-4
PPO clip parameter	0:1
PPO epochs	4
Number of processes	8
GAE	0:95
Entropy coef	0:01
Value loss coef	0:5
Number of forward steps per update	128

Table 3: A complete overview of used hyper parameters for all methods.

Figure 5: Our method with KL divergences in comparison to PPO and TRPO on MuJoCo, with 10 seeds. Standard error shaded.