
A general class of surrogate functions for stable and efficient reinforcement learning

Sharan Vaswani¹

Olivier Bachem²

Simone Totaro³

Robert Müller⁴

Shivam Garg⁵

Matthieu Geist²

Marlos C. Machado^{5,6}

Pablo Samuel Castro²

Nicolas Le Roux^{3,7,8}

¹Simon Fraser University

²Google Brain

³Mila, Université de Montréal

⁴TU Munich

⁵Amii, University of Alberta

⁶DeepMind

⁷Microsoft Research

⁸Mila, McGill

Abstract

Common policy gradient methods rely on the maximization of a sequence of surrogate functions. In recent years, many such surrogate functions have been proposed, most without strong theoretical guarantees, leading to algorithms such as TRPO, PPO or MPO. Rather than design yet another surrogate function, we instead propose a general framework (FMA-PG) based on functional mirror ascent that gives rise to an entire family of surrogate functions. We construct surrogate functions that enable policy improvement guarantees, a property not shared by most existing surrogate functions. Crucially, these guarantees hold regardless of the choice of policy parameterization. Moreover, a particular instantiation of FMA-PG recovers important implementation heuristics (e.g., using forward vs reverse KL divergence) resulting in a variant of TRPO with additional desirable properties. Via experiments on simple reinforcement learning problems, we evaluate the algorithms instantiated by FMA-PG. The proposed framework also suggests an improved variant of PPO, whose robustness and efficiency we empirically demonstrate on the MuJoCo suite.

1 INTRODUCTION

Policy gradient (PG) methods (Williams, 1992; Sutton et al., 2000; Konda and Tsitsiklis, 2000; Kakade, 2002) are an important class of model-free methods in reinforcement learning. They enable a differentiable policy parameterization and can easily handle function approximation and structured state-action spaces. PG methods based on REINFORCE (Williams and Peng, 1991) are equipped with strong theoretical guarantees in restricted settings (Agarwal et al., 2020; Mei et al., 2020; Cen et al., 2020). For these methods, each policy update requires recomputing the policy gradient. This in turn requires interacting with the environment or the simulator which can be computationally expensive.

On the other hand, methods such as TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017) and MPO (Abdolmaleki et al., 2018) support off-policy updates, i.e. they can update the policy without requiring additional interactions with the environment. These methods are efficiently implementable and have good empirical performance (Dhariwal et al., 2017). All of these methods rely on constructing *surrogate functions* of the policy, then updating the policy by maximizing these surrogates. Unfortunately, most of these surrogate functions (including those for PPO, TRPO and MPO) do not have strong theoretical guarantees. Consequently, this class of PG methods only has performance guarantees in the tabular setting (Kakade and Langford, 2002; Schulman et al., 2015; Neu et al., 2017; Geist et al., 2019; Shani et al., 2020), and some of these can even fail to converge in simple scenarios (Hsu et al., 2020). More importantly, *there is no systematic way to*

design theoretically principled surrogate functions, or a unified framework to analyze their properties. We address these issues through the following contributions.

Functional mirror ascent for policy gradient: In Section 3, we construct surrogate functions using mirror ascent on a functional representation of the policy itself, rather than on its parameters. We call this approach functional mirror ascent (FMA) and derive its update for policy gradient methods. The FMA update results in a surrogate function that is independent of the policy parameterization. We use it to propose FMA-PG (FMA for PG), a general framework for constructing surrogate functions and introduce a generic policy optimization algorithm that relies on approximately maximizing a sequence of surrogate functions.

Theoretical guarantees for FMA-PG: In Section 4, we explain the theoretical advantages of using FMA-PG. In particular, we describe a sufficient condition that guarantees that maximizing the sequence of surrogate functions instantiated by FMA-PG will result in monotonic policy improvement and ensure convergence to a stationary point. Crucially, *these guarantees hold regardless of the choice of policy parameterization.*

Instantiating the FMA-PG framework: In Section 5, we instantiate the FMA-PG framework with two common functional representations – direct and softmax representations. For each of these, we compare the resulting surrogate function to existing methods in the literature. For each representation, we prove that a specific surrogate function instantiated by FMA-PG satisfies the sufficient condition in Section 4. Consequently, maximizing it guarantees monotonic policy improvement for *arbitrarily complicated policy parameterizations* including neural networks. Such a property is not shared by existing surrogate functions including those for PPO, TRPO and MPO.

For the softmax functional representation, FMA-PG results in a surrogate function that is a more stable variant of TRPO (Schulman et al., 2015) and MDPO (Tomar et al., 2020). Moreover, it recovers implementation heuristics (e.g. using forward vs reverse KL divergence) in a principled manner. Additionally, in Appendix A, we show that FMA-PG can handle stochastic value gradients (Heess et al., 2015).

Experimental evaluation: Finally, in Section 6, we evaluate the performance of surrogate functions instantiated by FMA-PG on simple bandit and reinforcement learning settings. FMA-PG also suggests a variant of PPO (Schulman et al., 2017), whose robustness and efficiency we demonstrate on continuous control tasks in the MuJoco environment (Todorov et al., 2012).

2 PROBLEM FORMULATION

We consider an infinite-horizon discounted Markov decision process (MDP) (Puterman, 1994) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, d_0, \gamma \rangle$ where \mathcal{S} and \mathcal{A} is the set of states and actions respectively, $p : \mathcal{S} \times \mathcal{A} \rightarrow \Delta^{\mathcal{S}}$ the transition probability function, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function, d_0 the initial distribution over states, and $\gamma \in [0, 1)$ the discount factor. Each policy π induces a distribution $p^\pi(\cdot|s)$ over actions for each state s . It also induces a measure d^π over states such that $d^\pi(s) = \sum_{\tau=0}^{\infty} \gamma^\tau \mathbb{P}(s_\tau = s \mid s_0 \sim d_0, a_\tau \sim p^\pi(a_\tau|s_\tau))$. Similarly, we define μ^π as the induced measure over state-action pairs induced by policy π , implying that $\mu^\pi(s, a) = d^\pi(s)p^\pi(a|s)$ and $d^\pi(s) = \sum_a \mu^\pi(s, a)$. The expected discounted return for π is defined as $J(\pi) = \mathbb{E}_{s_0, a_0, \dots} [\sum_{\tau=0}^{\infty} \gamma^\tau r(s_\tau, a_\tau)]$, where $s_0 \sim d_0, a_\tau \sim p^\pi(a_\tau|s_\tau)$, and $s_{\tau+1} \sim p(s_{\tau+1}|s_\tau, a_\tau)$. Given a set of feasible policies Π , the objective is to compute the policy that maximizes $J(\pi)$. We define $\pi^* := \arg \max_{\pi \in \Pi} J(\pi)$ as the *optimal policy*.

We call the set of distributions $p^\pi(\cdot|s)$ for each s or the measure d^π *functional representations* of the policy π . Note that a single policy policy π can have multiple functional representations. In general, optimizing J directly with respect to any functional representation of π is intractable. Consequently, the standard approach is to parameterize π by a set of parameters $\theta \in \mathbb{R}^d$ and to directly optimize J with respect to θ . However, it is critical to remember that *the functional representation of a policy is independent of its parameterization*.

There are other possible functional representations of a policy besides the two mentioned above. For example, since $p^\pi(\cdot|s)$ is a probability distribution, one can write $p^\pi(a|s) = \exp(z^\pi(a, s)) / \sum_{a'} \exp(z^\pi(a', s))$, and represent π as the set of $z^\pi(a, s)$ for each (a, s) pair. We call this particular functional representation the *softmax representation*, as opposed to the set of $p^\pi(a|s)$ which we call the *direct representation*. In the next section, we describe how to use the functional representation of a policy to derive a surrogate function. Although multiple functional representations can be equivalent in the class of policies they define, they result in different surrogate functions (Sections 5.1 and 5.2). Finally, we note that functional representations are not limited to stochastic policies and one can, for instance, represent a deterministic, stationary policy by specifying the state-action mapping for each state (Appendix A).

3 FUNCTIONAL MIRROR ASCENT FOR POLICY GRADIENT

In the previous section, we defined the functional representation of a policy. However, as we mentioned, typically, one cannot optimize J with respect to these

representations directly, in which case the policy π is parameterized. While the functional representation defines a policy’s sufficient statistics, the *policy parameterization* specifies the practical realization of these statistics and defines the set Π of realizable (representable) policies. The parameterization is independent of the functional representation, is explicit and determined by a *model* with parameters θ . For example, we could represent a policy by its state-action occupancy measure and use a linear parameterization to realize this measure, implying $\mu^\pi(s, a|\theta) = \langle \theta, \phi(s, a) \rangle$, where θ is the parameter to be optimized and $\phi(s, a)$ are the known features providing information about the state-occupancy measures. Similarly, we could use a neural-network parameterization for the variables that define a policy in its softmax representation, rewriting $z^\pi(a, s) = z^\pi(a, s|\theta)$. In order to compare to existing methods (Agarwal et al., 2020; Mei et al., 2020), we also define a *tabular parameterization*. For a finite state-action MDP with S states and A actions, choosing a tabular parameterization with the softmax representation results in $\theta \in \mathbb{R}^{SA}$ such that $\forall s \in \mathcal{S}, a \in \mathcal{A}, z^\pi(a, s|\theta) = \theta_{s,a}$.

Next, we describe a form of mirror ascent to directly update a policy’s functional representation.

3.1 Functional Mirror Ascent Update

To state the functional mirror ascent (FMA) update, we define a strictly convex, differentiable function ϕ as the mirror map. We denote by $D_\phi(\pi, \mu)$ the Bregman divergence associated with the mirror map ϕ between policies π and μ . Each iteration $t \in [T]$ of FMA consists of the update and projection steps (Bubeck, 2015): Eq. (1) computes the gradient $\nabla_\pi J(\pi_t)$ with respect to the policy’s functional representation and updates π_t to $\pi_{t+1/2}$ using a step-size η ; Eq. (2) computes the Bregman projection of $\pi_{t+1/2}$ onto the class of realizable policies, obtaining π_{t+1} .

$$\pi_{t+1/2} = (\nabla\phi)^{-1}(\nabla\phi(\pi_t) + \eta\nabla J(\pi_t)), \quad (1)$$

$$\pi_{t+1} = \arg \min_{\pi \in \Pi} D_\phi(\pi, \pi_{t+1/2}). \quad (2)$$

The above FMA updates can also be written as (c.f. Bubeck, 2015):

$$\pi_{t+1} = \arg \max_{\pi \in \Pi} \left[\langle \pi, \nabla_\pi J(\pi_t) \rangle - \frac{1}{\eta} D_\phi(\pi, \pi_t) \right]. \quad (3)$$

Note that the FMA update is solely in the functional space, and is specified by the choice of the functional representation and mirror map. The update requires solving a sub-problem to project the updated policy onto the set Π . Since the policy parameterization defines the set Π of realizable policies, it influences the difficulty of solving this projection sub-problem as well

as the final policy π_{t+1} . For simple policy parameterizations such as tabular or when using a linear model, the set Π is convex and the minimization in Eq. (3) can be done exactly. When using more complex policy parameterizations (e.g. deep neural network), the set of realizable policies Π can become arbitrarily complicated and non-convex, making the projection in Eq. (3) infeasible. The FMA-PG framework overcomes this issue as follows.

3.2 FMA-PG Framework

We assume that Π consists of policies that are realizable by a model parameterized by $\theta \in \mathbb{R}^d$. Throughout the paper, we will use π to refer to a policy’s functional representation, whereas $\pi(\theta)$ will refer to the parametric realization of π . We do not impose any restriction on the parameterization and any generic model (e.g. neural network) can be used to parameterize π . The choice of the policy parameterization is implicit in the $\pi(\theta)$ notation. For the special case of the tabular parameterization, $\pi = \pi(\theta) = \theta$.

Solving Eq. (2) iteratively may be interpreted as finding a path that starts from $\pi_{t+1/2}$ and gradually gets closer to the set Π . In this view, an approximate solution would be a point along that path that is not in the set Π , and consequently not realizable by a vector θ . Another perspective is to interpret solving Eq. (2) as finding a path *within* Π that starts from π_t , the previous policy (already in Π), and gets closer to $\pi_{t+1/2}$ (potentially outside Π). Any point along such a path is within Π and is thus realizable. In other words, we replace (2) with another problem with the same solution:

$$\arg \min_{\pi \in \Pi} D_\phi(\pi, \pi_{t+1/2}) = \arg \min_{\theta \in \mathbb{R}^d} D_\phi(\pi(\theta), \pi_{t+1/2}). \quad (4)$$

With this reparameterization, no projection is required and the update in Eq. (3) can be written as a parametric, unconstrained optimization problem. This is a critical property as it makes FMA-PG applicable to any policy parameterization.

In particular, if $\pi_t = \pi(\theta_t)$, $\theta_{t+1} \in \mathbb{R}^d$ is the solution to the RHS of Eq. (4) and $\pi_{t+1} = \pi(\theta_{t+1})$, then Eq. (3) can be written as the maximization of a *surrogate function*, $\theta_{t+1} = \arg \max_{\theta \in \mathbb{R}^d} \ell_t^{\pi, \phi, \eta}(\theta)$, where

$$\begin{aligned} \ell_t^{\pi, \phi, \eta}(\theta) := & J(\pi(\theta_t)) + \langle \pi(\theta) - \pi(\theta_t), \nabla_\pi J(\pi(\theta_t)) \rangle \\ & - \frac{1}{\eta} D_\phi(\pi(\theta), \pi(\theta_t)). \end{aligned} \quad (5)$$

The surrogate function $\ell_t^{\pi, \phi, \eta}(\theta)$ is a function of θ , but it is specified by the choice of the functional representation, the mirror map Φ , and the step-size η . Note that as compared to Eq. (3), in Eq. (5), we added terms

independent of θ which do not change the arg max but will prove useful to prove guarantees in Section 4. We have thus used the FMA update in Eq. (3) to specify a family of surrogate functions that can be used with any policy parameterization. We refer to this general framework of constructing surrogates for policy gradient methods as FMA-PG.

The surrogate function in Eq. (5) is non-concave in general and can be maximized using a gradient-based algorithm. We will use m gradient steps with a step-size α to maximize $\ell_t^{\pi, \phi, \eta}(\theta)$. With this choice, we can now state a generic policy optimization algorithm (pseudo-code in Algorithm 1). We see that the surrogate function $\ell_t^{\pi, \phi, \eta}$ acts as a “guide” for the parametric updates in the inner loop, similar to the supervised learning method proposed by Johnson and Zhang (2020).

Algorithm 1: Generic policy optimization

Input: π (choice of functional representation), θ_0 (initial policy parameterization), T (PG iterations), m (inner-loops), η (step-size for functional update), α (step-size for parametric update)

for $t \leftarrow 0$ **to** $T - 1$ **do**

- Compute gradient $\nabla_{\pi} J(\pi_t)$ and form function $\ell_t^{\pi, \phi, \eta}(\theta)$ as in Eq. (5)
- Initialize inner-loop: $\omega_0 = \theta_t$
- for** $k \leftarrow 0$ **to** m **do**
- $\omega_{k+1} = \omega_k + \alpha \nabla_{\omega} \ell_t^{\pi, \phi, \eta}(\omega_k)$
- end**
- $\theta_{t+1} = \omega_m$
- $\pi_{t+1} = \pi(\theta_{t+1})$

end

Return θ_T

4 THEORETICAL GUARANTEES

In this section, we explain the theoretical advantage of using surrogate functions instantiated by FMA-PG. Recall that the policy is updated through the (potentially approximate) maximization of Eq. (5). To guarantee that maximizing the surrogate function improves the resulting policy, i.e. $J(\pi_{t+1}) \geq J(\pi_t)$, a sufficient condition is to have $\ell_t(\theta) \leq J(\pi(\theta))$ for all θ . Indeed, if ℓ_t is a uniform lower-bound on J , then,

$$\begin{aligned} J(\pi_{t+1}) &= J(\pi(\theta_{t+1})) \geq \ell_t(\theta_{t+1}) \\ &\geq \ell_t(\theta_t) \quad (\text{By maximizing the surrogate function}) \\ &= J(\pi(\theta_t)) = J(\pi_t) \quad (\text{From Eq. (5)}) \end{aligned}$$

For stating a more practical condition that guarantees that the surrogate function is a uniform lower-bound on J , we prove the following proposition in Appendix B.

Proposition 1 (Guarantee on surrogate function). *The surrogate function $\ell_t^{\pi, \phi, \eta}$ is a lower bound of J if and only if $J + \frac{1}{\eta}\phi$ is a convex function of π .*

The above proposition shows that the desired property is guaranteed by selecting an appropriate value of η that only depends on properties of J and the mirror map Φ in the functional space. Once again, we emphasize that the guarantees offered by the surrogate function are *independent of the parameterization*.

We have seen that if the surrogate is a uniform lower bound on J , then the equality of the two functions at $\theta = \theta_t$ (from Eq. (5)) guarantees that any improvement of the surrogate leads to an improvement of J . The following result states that improvement in the surrogate can be guaranteed provided that the parametric step-size α is chosen according to the smoothness of the surrogate function.

Theorem 1 (Guaranteed policy improvement for Algorithm 1). *Assume that ℓ_t is β -smooth w.r.t. the Euclidean norm and that η satisfies the condition of Proposition 1. Then, for any $\alpha \leq 1/\beta$, iteration t of Algorithm 1 guarantees $J(\pi_{t+1}) \geq J(\pi_t)$ for any number m of inner-loop updates.*

Note that Algorithm 1 and the corresponding theorem can be easily extended to handle stochastic parametric updates. This will guarantee that $\mathbb{E}[J(\pi_{t+1})] \geq J(\pi_t)$ where the expectation is over the sampling in the parametric SGD steps. Similarly, both the algorithm and theoretical guarantee can be generalized to incorporate the relative smoothness of $\ell_t(\theta)$ w.r.t. a general Bregman divergence (Lu et al., 2018).

For rewards in $[0, 1]$, $J(\pi)$ is upper-bounded by $\frac{1}{1-\gamma}$, and hence monotonic improvements to the policy guarantee convergence to a stationary point. We emphasize that the above result holds for *any arbitrarily complicated policy parameterization*. Hence, a successful PG method (one that reliably improves the policy) relies on appropriately setting two step-sizes: η at the functional level and α at the parametric level.

5 INSTANTIATING FMA-PG

We now instantiate the FMA-PG framework with two common functional representations: the direct representation (Section 5.1) and the softmax representation (Section 5.2), deriving values for η for each.

5.1 Direct Functional Representation

In the direct functional representation, the policy π is represented by the set of distributions $p^{\pi}(\cdot|s)$ over actions for each state $s \in \mathcal{S}$. Using the policy gradient theorem (Sutton and Barto, 2018), in this

case, $\frac{\partial J(\pi)}{\partial p^\pi(a|s)} = d^\pi(s)Q^\pi(s, a)$. Since $p^\pi(\cdot|s)$ is a set of distributions (one for each state), we define the mirror map as $\phi(\pi) = \sum_{s \in \mathcal{S}} w(s) \phi(p^\pi(\cdot|s))$, where $w(s)$ is a positive weighting on the states s . Note that the positive weights ensure that ϕ is a valid mirror-map. The resulting Bregman divergence is $D_\phi(\pi, \pi') = \sum_s w(s)D_\phi(p^\pi(\cdot|s), p^{\pi'}(\cdot|s))$, that is, the weighted sum of the Bregman divergences between the action distributions in state s . By choosing $w(s)$ equal to $d^{\pi_t}(s)$, and parameterizing the functional representation, i.e. $p^{\pi_t}(\cdot|s) = p^\pi(\cdot|s, \theta_t)$, we obtain the following form of the surrogate function:

$$\ell_t^{\pi, \phi, \eta}(\theta) = \mathbb{E}_{(s, a) \sim \mu^{\pi_t}} \left[\left(Q^{\pi_t}(s, a) \frac{p^\pi(a|s, \theta)}{p^\pi(a|s, \theta_t)} \right) \right] - \frac{1}{\eta} \mathbb{E}_{s \sim d^{\pi_t}} [D_\phi(p^\pi(\cdot|s, \theta), p^\pi(\cdot|s, \theta_t))] , \quad (6)$$

where the constants independent of θ were omitted. By choosing ϕ and η , the above surrogate function can be used with Algorithm 1. We now discuss how to set η that guarantees monotonic policy improvement when using the above surrogate function with the negative entropy mirror map, i.e. $\phi_{NE}(p^\pi(\cdot|s)) = -\sum_a p^\pi(a|s) \log p^\pi(a|s)$.

Proposition 2 (Improvement guarantees for direct functional representation). *Assuming that the rewards are in $[0, 1]$, when using the surrogate function in Eq. (6) with the mirror map chosen to be the negative entropy, then $J \geq \ell_t^{\pi, \phi, \eta}$ for $\eta \leq \frac{(1-\gamma)^3}{2\gamma|A|}$.*

This proposition is proved in Appendix C. Using the argument in Section 4, we can infer that using the direct functional representation with the negative entropy mirror map and $\eta \leq \frac{(1-\gamma)^3}{2\gamma|A|}$ ensures monotonic policy improvement for *any policy parameterization*.

Next, we discuss how the surrogate function in Eq. (6) and the resulting algorithm is related to existing methods. When using a tabular parameterization, i.e. when $\pi(\theta) = \theta$, we make the following connections:

Connection to uniform TRPO and MDPI: With the tabular parameterization, the proposed update is similar to the update in uniform TRPO (Shani et al., 2020) and Mirror Descent Modified Policy Iteration (Geist et al., 2019).

Connection to CPI: For finite states and actions, when using a tabular parameterization, the first term in Eq. (6) becomes the same as in conservative policy iteration (CPI) (Kakade and Langford, 2002). In CPI, the authors first derive the form $\sum_s d^\pi(s) \sum_a p^\pi(a|s) Q^{\pi_t}(s, a)$, then use a mixture policy to ensure that π is “close” to π_t and justify replacing $d^\pi(s)$ in the above expression by d^{π_t} . On the other hand, we use the FMA-PG framework to directly de-

rive Eq. (6) and allow for the use of any Bregman divergence to ensure the proximity between π and π_t . While we derive the CPI update from an unconstrained optimization viewpoint, CPI has also been connected to constrained optimization with an equivalence to functional Frank-Wolfe (Scherrer and Geist, 2014).

Connection to REINFORCE-based methods: For finite states and actions, when using a tabular parameterization and Algorithm 1 with $m = \infty$ (exact minimization of the surrogate), if we choose the (i) squared Euclidean distance as the mirror map, the proposed update is the same as standard REINFORCE (Williams and Peng, 1991; Agarwal et al., 2020) and (ii) negative entropy as the mirror map (implying that the resulting Bregman divergence is the KL divergence), the proposed update is equal to natural policy gradient (Kakade, 2001).

Comparison to MDPO: With a direct functional representation, negative entropy mirror map and a general policy parameterization, the resulting FMA-PG update is similar to MDPO (Tomar et al., 2020). The only difference between the two updates is that MDPO involves the advantage A^{π_t} instead of the Q^{π_t} term in Eq. (6). Since both A^{π_t} and Q^{π_t} are independent of p^π , this difference does not matter for gradient-based algorithms maximizing the surrogate (see caption of Table 1 in Appendix E.1). Hence, MDPO directly falls under the FMA-PG framework.

The above formulation has two main shortcomings. First, it involves $p^\pi(a|s, \theta)$, which means that for each parametric update, either (i) the actions need to be resampled on-policy, or (ii) the update involves an importance-sampling ratio $p^\pi(a|s, \theta)/p^\pi(a|s, \theta_t)$ like in Eq. (6). This requires clipping the ratio for stability, and can potentially result in overly conservative updates (Schulman et al., 2017). Moreover, with the mirror map as the negative entropy, the Bregman divergence is the *reverse* KL divergence, i.e. $D_\phi(p^\pi(\cdot|s, \theta), p^\pi(\cdot|s, \theta_t)) = \text{KL}(p^\pi(\cdot|s, \theta)||p^\pi(\cdot|s, \theta_t))$. The reverse KL divergence makes this objective *mode seeking*, in that the policy π might only capture a subset of the actions covered by π_t . Past works have addressed this issue either by adding entropy regularization (Geist et al., 2019; Shani et al., 2020), or by simply reversing the KL, using the *forward* KL: $\text{KL}(p^\pi(\cdot|s, \theta_t)||p^\pi(\cdot|s, \theta))$ (Mei et al., 2019). However, using entropy regularization results in a biased policy, whereas the forward KL does not correspond to a valid Bregman divergence in p^π and can converge to a sub-optimal policy. We now show how FMA-PG with the softmax representation addresses both these issues in a principled way, providing a theoretical justification to heuristics that are used to improve PG methods.

5.2 Softmax Functional Representation

Since $p^\pi(\cdot|s)$ is a distribution, it has an equivalent softmax representation that we study in this section. The softmax functional representation results in the FMA update on the logits $z^\pi(a, s)$ of the conditional distributions $p^\pi(a|s)$. Formally, $p^\pi(a|s) = \frac{\exp(z^\pi(a, s))}{\sum_{a'} \exp(z^\pi(a', s))}$ and the policy gradient theorem yields $\frac{\partial J(\pi)}{\partial z^\pi(a, s)} = d^\pi(s) A^\pi(s, a) p^\pi(a|s)$. Here, $A^\pi(s, a)$ is the advantage function equal to $Q^\pi(s, a) - V^\pi(s)$. Similar to Section 5.1, we use a mirror map $\phi_z(z)$ that decomposes across states, i.e. $\phi_z(z) = \sum_s w(s) \phi_z(z^\pi(\cdot, s))$ for some positive weighting w . We denote the corresponding Bregman divergence as D_{ϕ_z} and choose $w(s) = d^{\pi_t}(s)$. Parameterizing the logits as $z^\pi(a, s, \theta)$ and noting that $p^{\pi_t}(a|s) = p^\pi(a|s, \theta_t)$, we obtain the following form of the surrogate function:

$$\ell_t^{\pi, \phi, \eta}(\theta) = E_{(s, a) \sim \mu^{\pi_t}} [A^{\pi_t}(s, a) p^\pi(a|s, \theta_t)] - \frac{1}{\eta} \sum_s w(s) D_{\phi_z}(z^\pi(\cdot, s, \theta), z^\pi(\cdot, s, \theta_t)). \quad (7)$$

We now discuss how the surrogate function in Eq. (7) and the resulting algorithm relate to existing methods.

Connection to REINFORCE-based methods: For finite states and actions and when using a tabular parameterization and the squared Euclidean mirror map, Algorithm 1 with $m = 1$ leads to the same update as that of policy gradient with the softmax parameterization (Agarwal et al., 2020; Mei et al., 2020).

A more interesting surrogate emerges when ϕ is the *logsumexp*, i.e. $\phi_z(z) = \sum_s w(s) \log(\sum_a \exp(z^\pi(a, s)))$, and $w(s) = d^{\pi_t}(s)$. Then,

$$\ell_t^{\pi, \phi, \eta}(\theta) = E_{(s, a) \sim \mu^{\pi_t}} \left[\left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) \log \frac{p^\pi(a|s, \theta)}{p^\pi(a|s, \theta_t)} \right], \quad (8)$$

omitting the constant terms independent of θ . The full derivation of this computation can be found in Proposition 4 of Appendix C. We now discuss how to set η that guarantees monotonic policy improvement when using the above surrogate function.

Proposition 3 (Improvement guarantees for softmax functional representation). *Assuming that the rewards are in $[0, 1]$, then the surrogate function in Eq. (8) satisfies $J \geq \ell_t^{\pi, \phi, \eta}$ for $\eta \leq 1 - \gamma$.*

This proposition is proved in Appendix C. As before, we can infer that using the softmax functional representation with the logsumexp mirror map and $\eta \leq 1 - \gamma$ ensures monotonic policy improvement for *any policy parameterization*. Although we have used the same η for all states s , the updates in Eqs. (6) and (8) can

accommodate a different step-size $\eta(s)$ for each state. This is likely to yield tighter lower bounds and larger improvements in the inner loop. Determining such step-sizes is left for future work.

Unlike the formulation in Eq. (6), we see that Eq. (8) relies on the logarithm of the importance sampling ratios. Moreover, Eq. (8) can be written as

$$\ell_t^{\pi, \phi, \eta} = \mathbb{E}_{s \sim d^{\pi_t}} \left[\mathbb{E}_{a \sim p^{\pi_t}} \left(A^{\pi_t}(s, a) \log \frac{p^\pi(a|s, \theta)}{p^\pi(a|s, \theta_t)} \right) - \frac{1}{\eta} \text{KL}(p^\pi(\cdot|s, \theta_t) || p^\pi(\cdot|s, \theta)) \right]. \quad (9)$$

Comparing to Eq. (6), we observe that the KL divergence is in the *forward* direction and is *mode covering*. This naturally prevents a mode-collapse of the policy and encourages exploration. We thus see that FMA-PG is able to recover an implementation heuristic (forward vs reverse KL) in a principled manner. Moreover, we can interpret Eq. (9) as a variant of TRPO with desirable properties, as we discuss next.

Comparison to TRPO: Comparing Eq. (9) to the TRPO update (Schulman et al., 2015), $\arg \max_{\theta \in \mathbb{R}^d} \mathbb{E}_{(s, a) \sim \mu^{\pi_t}} [A^{\pi_t}(s, a) \frac{p^\pi(a|s, \theta)}{p^\pi(a|s, \theta_t)}]$, such that $\mathbb{E}_{s \sim d^{\pi_t}} [\text{KL}(p^{\pi_t}(\cdot|s, \theta_t) || p^\pi(\cdot|s, \theta))] \leq \delta$, we observe that Eq. (9) involves the logarithm of p^π , which can be interpreted as a form of soft clipping due to the narrower range of the log ratio. Additionally, when the policy is modeled by a deep network with a final softmax layer, this leads to an objective concave in the last layer, which is in general easier to optimize than the original TRPO objective. Unlike TRPO, the proposed update enforces the proximity between policies via a regularization rather than a constraint. This modification has been recently found to be beneficial (Lazić et al., 2021). Finally, the parameter δ in TRPO is a hyper-parameter that needs to be tuned. In contrast, the regularization strength $1/\eta$ in proposed update can be determined theoretically (Proposition 3).

6 EXPERIMENTAL EVALUATION

While this work focuses on providing a general framework for designing surrogate functions, we explore the behaviour of surrogates instantiated by the softmax functional representation in three different settings. **First**, to avoid dealing with local maxima of J , we explore a multi-armed bandit, where we compare it to the exponential weights algorithm (EXP3) (Auer et al., 2002) in Section 6.1. The simplicity of the environment allows us to get a clearer understanding of the behaviour of each algorithm. **Second**, we set up small-scale RL environments where the surrogates in Section 5 can be maximized exactly. We assume access to the exact MDP dynamics and rewards model to

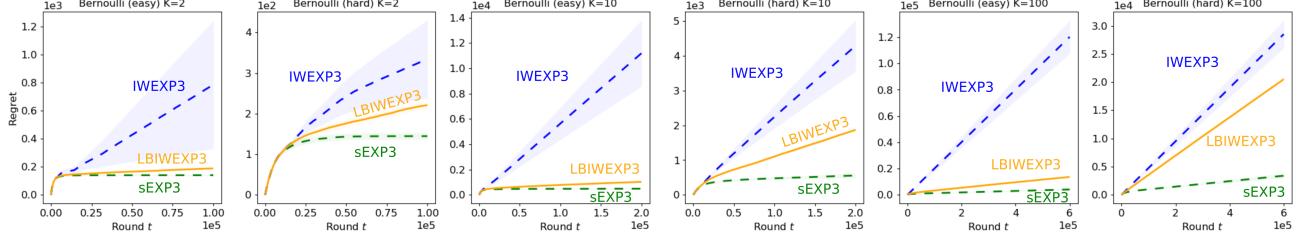


Figure 1: Comparing the average regret over 50 runs for two variants of EXP3 – with standard importance weights (IWEXP3) or loss-based importance weights (LBIWEXP3) to that of sEXP3. Both algorithms use a tuned step-size equal to 0.005. We observe that sEXP3 consistently achieves lower regret.

focus on the impact of the proposed surrogate, ignoring potential interactions with a critic and avoiding exploration and sampling issues. Finally, we tested the practical performance of FMA-PG using a larger-scale experiment on MuJoCo in Section 6.3. In addition to the increased complexity of the environments, this experiment allows us to explore how the surrogate behaves in the presence of a critic. Since the policies are parameterized as a deep network, the surrogate can only be maximized approximately.

6.1 Multi-armed Bandit

For a stochastic multi-armed bandit problem, we compare EXP3, which corresponds to the single-state, tabular parameterization of FMA-PG with the direct representation and the negative entropy mirror map; to softmax EXP3 (*sEXP3*), which uses the softmax parameterization and the logsumexp mirror map. For EXP3, we use the standard importance weighting procedure (denoted as IWEXP3 in the plots) as well as the loss-based variation (Lattimore and Szepesvári, 2020) (denoted as LBIWEXP3). We choose the step-size η that achieved the best average final regret for each algorithm over 50 runs (see Appendix D for details). Figure 1 shows that sEXP3 consistently achieves lower regret than the EXP3 variants, regardless of the number of arms (2, 10, 100) and the problem difficulty determined by the action gap.

6.2 Tabular MDP

We use two tabular environments: CliffWorld (Sutton and Barto, 2018) and DeepSeaTreasure (Osband et al., 2019), and a tabular softmax policy parameterization (one parameter for each state and action). We study the performance of four algorithms, two of which are instantiated by the FMA-PG framework – (i) sMDPO (maximizing the objective given in Eq. (8)), (ii) MDPO (objective given in (7) with a negative entropy mirror map), and two commonly used PG methods – (iii) PPO (Schulman et al., 2017) and (iv) TRPO (Schulman et al., 2015). Of these, sMDPO and MDPO

have two hyper-parameters: η (outer loop stepsize) and α (the inner loop step-size); PPO has two hyper-parameters, ϵ (clipping factor) and α (the inner-loop step-size) whereas TRPO has a single hyper-parameter δ , the magnitude of the KL-constraint. For all the algorithms, we use the true action-value functions. The complete experimental setup, implementation details, and additional experiments are in Appendix E and F.

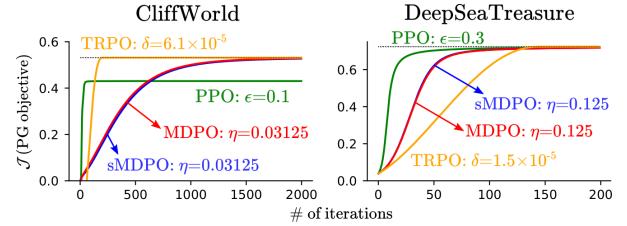


Figure 2: Comparing PG algorithms on CliffWorld and DeepSeaTreasure environments for 100 inner-loop updates and best set of hyper-parameters.

Figure 2 shows the algorithm performance with the number of outer-loops (interactions with the environment) for $m = 100$ inner-loop updates. We show the performance for the best set of hyper-parameters for each algorithm and environment. We observe that (i) with exact computation of action-value functions, MDPO and sMDPO have similar performance, and (ii) for both environments, sMDPO, MDPO and TRPO are able to reach the performance of the optimal policy, whereas PPO (with the best hyper-parameter) converges to a sub-optimal policy for CliffWorld. For both sMDPO and MDPO, the theoretically derived step-sizes in Proposition 2 and Proposition 3 are much smaller than the best tuned step-sizes (see Appendix E.4 for exact calculations). Using theoretically derived step-sizes result in slow (but monotonic) convergence, verifying Theorem 1. Our results show that sMDPO and MDPO are competitive with popular PG algorithms, and demonstrate the effectiveness of FMA-PG in designing theoretically sound and practical PG methods.

Ablation Study: In Figure 3, we study the effect of

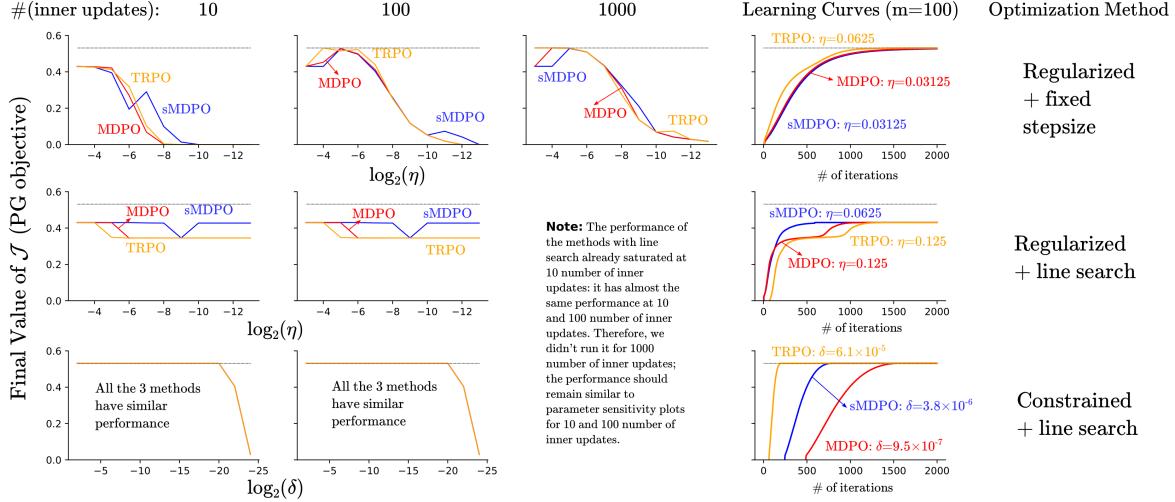


Figure 3: Parameter sensitivity for sMDPO, MDPO and TRPO on CliffWorld for 2000 environment interactions with different number of inner loop updates (first 3 columns) and different algorithmic choices (rows) (see Appendix E for exact expressions). For each plot, the X-axis shows the sensitivity towards the corresponding hyper-parameter (the other hyper-parameters are set to best-tuned values). The first row shows the regularized (with parameter η) variants (default variant of sMDPO and MDPO used in Figure 2). The second row also shows the regularized variants but uses a line-search to set the step-size for each inner-loop. Instead of enforcing the proximity between consecutive policies via regularization, the variants in the third row use a constraint with parameter δ (default variant of TRPO used in Figure 2). For each row, the fourth column shows the algorithm performance vs the number of environment interactions. Black lines correspond to the value of the optimal policy.

different algorithmic choices and sensitivity towards the corresponding hyper-parameter (see the caption for details) for sMDPO, MDPO, and TRPO for CliffWorld (DeepSeaTreasure results in Appendix E). We observe that (i) increasing the number of inner-loops (marginally) improves the performance of each method, demonstrating the effect of data reuse (ii) in the first row, all methods perform worse as the regularization increases from left to right, and the regularized variant of TRPO (Lazić et al., 2021) has similar performance as sMDPO and MDPO, (iii) in the second row, using a line-search for the inner-loop makes all methods more robust to η , but the aggressive (using large step-sizes) inner-loop updates can result in convergence to a sub-optimal policy, (iv) in the third row, the constrained variants of all methods are quite robust to the constraint hyper-parameter δ , with all methods converging to the optimal policy. Hence, for each method, using a constraint to enforce proximity between consecutive policies can result in superior performance over its regularized counterpart (with or without line-search).

6.3 Large-scale Continuous Control Tasks

Since PPO (Schulman et al., 2017) requires clipping the importance sampling ratio, in order to make the resulting algorithm similar to PPO for ease of implementation, we included clipping with the surrogate

function instantiated by FMA-PG. In particular, we modify Eq. (8) and the resulting surrogate given by:

$$\ell_t^{\pi, \phi, \eta}(\theta) = \mathbb{E}_{(s, a) \sim \mu^{\pi_t}} \left[A^{\pi_t}(s, a) \times \log \left(\text{clip} \left(\frac{p^\pi(a|s, \theta)}{p^\pi(a|s, \theta_t)}, \frac{1}{1+\epsilon}, 1+\epsilon \right) \right) \right],$$

We denote the above surrogate function and the resulting algorithm as *sPPO*. We investigate its performance on five continuous control environments from the OpenAI Gym (Brockman et al., 2016): Hopper-v1, Walker2d-v1, HalfCheetah-v1, Ant-v1, and Humanoid-v1. As a baseline, we use the PPO implementation from Andrychowicz et al. (2021) with their standard configuration and default hyperparameters values. We implement sPPO by adding a binary flag (`use_softmax`). We re-emphasize that both algorithms use a critic and that the hyper-parameters of the critic are tuned using PPO to avoid favoring our framework.

We investigate the differences between PPO and sPPO by training 180 different policies for each environment and all combinations of `use_softmax` $\in \{\text{True}, \text{False}\}$, $m \in \{10, 100\}$ and the importance weight capping value $\epsilon \in \{0.1, 0.3, 0.5, 0.7\}$ (a total compute of 1400 days with TPUs). We evaluate each policy 18 times during training, using the action with largest probability rather than a sample. We compute the average return

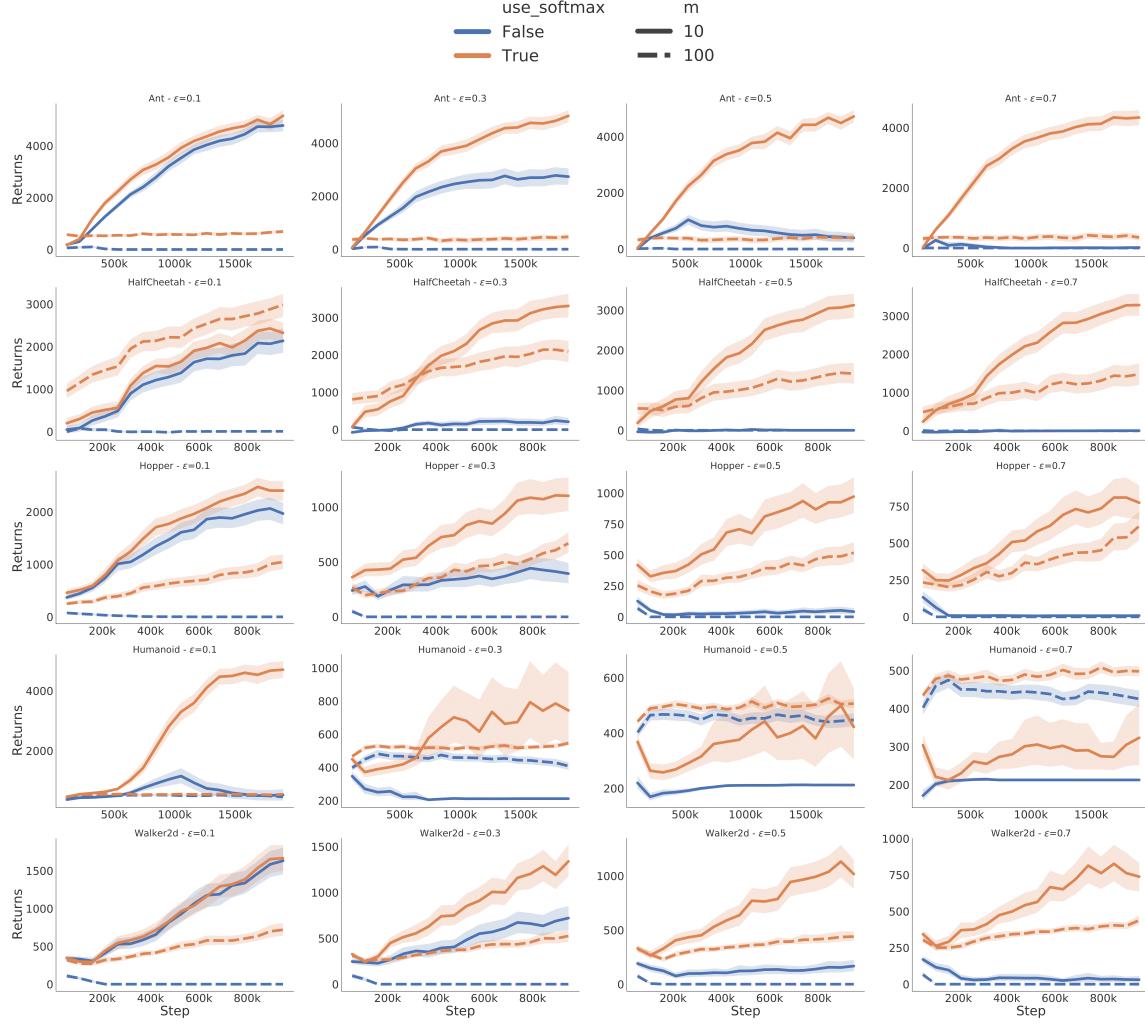


Figure 4: Average return and 95% confidence intervals (over 180 runs) for PPO and sPPO on 5 environments (rows) and for four different clipping values (columns). sPPO is more robust to large values of clipping, even more so when the number of updates in the inner loop grows (linestyle).

and 95% confidence intervals for each of the settings. The results are presented in Figure 4, where we see that sPPO outperforms PPO across all environments. Furthermore, we see that the difference is more pronounced when the number of iterations m in the inner loop is increased (linestyles) or when less capping is used (columns). In Appendix G, we show additional results but with learning rate decay and gradient clipping disabled, two commonly used techniques to stabilize PPO training (Engstrom et al., 2019). In this setting, sPPO only suffers a mild degradation while PPO fails completely, confirming sPPO’s additional robustness.

7 CONCLUSION

We proposed FMA-PG, a general framework to design computationally efficient policy gradient methods. By disentangling the functional representation of a policy from its parameterization, we unified different PG per-

spectives, recovering several existing algorithms and implementation heuristics in a principled manner. By using the appropriate theoretically-determined hyperparameters, FMA-PG guarantees policy improvement (and hence convergence to a stationary point) for the resulting PG method, even with arbitrarily complex policy parameterizations and for arbitrary number of inner-loop steps. We demonstrated that FMA-PG enables the design of new, improved surrogate functions that can lead to improved empirical results. We believe that our framework will further enable the systematic design of sample-efficient PG methods.

Our theoretical results assume the exact computation of the action-value and advantage functions, and are thus limited in practice. In the future, we aim to handle sampling errors and extend these results to the actor-critic framework.

8 Acknowledgements

We would like to thank Veronica Chelu for suggesting the use of the log-sum-exp mirror map in Section 5. Nicolas Le Roux and Marlos C. Machado are funded by a CIFAR chair. Sharan Vaswani and Shivam Garg gratefully acknowledge support from Csaba Szepesvari during the duration of this project.

References

- Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. A. (2018). Maximum a posteriori policy optimisation. In *International Conference on Learning Representations (ICLR)*.
- Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. (2020). Optimality and approximation with policy gradient methods in Markov decision processes. In *Conference on Learning Theory (COLT)*, pages 64–66.
- Andrychowicz, M., Raichuk, A., Stanczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. (2021). What matters for on-policy deep actor-critic methods? a largescale study. In *International conference on learning representations*.
- Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.
- Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.
- Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.
- Cen, S., Cheng, C., Chen, Y., Wei, Y., and Chi, Y. (2020). Fast global convergence of natural policy gradient methods with entropy regularization. *arXiv preprint arXiv:2007.06558*.
- Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. (2017). Openai baselines. <https://github.com/openai/baselines>.
- Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. (2019). Implementation matters in deep RL: A case study on PPO and TRPO. In *International conference on learning representations*.
- Geist, M., Scherrer, B., and Pietquin, O. (2019). A theory of regularized Markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR.
- Ghosh, D., C Machado, M., and Le Roux, N. (2020). An operator view of policy gradient methods. *Advances in Neural Information Processing Systems*, 33.
- Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. (2015). Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952.
- Hsu, C. C.-Y., Mendler-Dünner, C., and Hardt, M. (2020). Revisiting design choices in proximal policy optimization. *arXiv preprint arXiv:2009.10897*.
- Johnson, R. and Zhang, T. (2020). Guided learning of nonconvex models through successive functional gradient optimization. In *International Conference on Machine Learning*, pages 4921–4930. PMLR.
- Kakade, S. (2001). A natural policy gradient. In *NIPS*, volume 14, pages 1531–1538.
- Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 267–274.
- Kakade, S. M. (2002). A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538.
- Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.
- Lattimore, T. and Szepesvari, C. (2020). *Bandit algorithms*. Cambridge University Press.
- Lažić, N., Hao, B., Abbasi-Yadkori, Y., Schuurmans, D., and Szepesvari, C. (2021). Optimization issues in kl-constrained approximate policy iteration. *arXiv preprint arXiv:2102.06234*.
- Lu, H., Freund, R. M., and Nesterov, Y. (2018). Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354.
- Mei, J., Xiao, C., Huang, R., Schuurmans, D., and Müller, M. (2019). On principled entropy exploration in policy optimization. In *IJCAI*, pages 3130–3136.
- Mei, J., Xiao, C., Szepesvari, C., and Schuurmans, D. (2020). On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pages 6820–6829. PMLR.
- Neu, G., Jonsson, A., and Gómez, V. (2017). A unified view of entropy-regularized Markov decision processes. *CoRR*, abs/1705.07798.

- Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, second edition.
- Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., et al. (2019). Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*.
- Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA.
- Scherrer, B. and Geist, M. (2014). Local policy search in a convex space and conservative policy iteration as boosted policy search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer.
- Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.
- Shani, L., Efroni, Y., and Mannor, S. (2020). Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5668–5675.
- Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. *Journal of Machine Learning Research*.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2 edition.
- 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 (NeurIPS)*, pages 1057–1063.
- Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.
- Tomar, M., Shani, L., Efroni, Y., and Ghavamzadeh, M. (2020). Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*.
- Vaswani, S., Mehrabian, A., Durand, A., and Kveton, B. (2020). Old dog learns new tricks: Randomized ucb for bandit problems.
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.
- Williams, R. J. and Peng, J. (1991). Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268.

Supplementary Material: A general class of surrogate functions for stable and efficient reinforcement learning

Organization of the Appendix

- A Handling stochastic value gradients
- B Proofs for Section 4
- C Proofs for Section 5
- D Experimental details in the bandit setting
- E Experiments in the tabular setting
- F Analytical Updates and Gradient Expressions for tabular PG Algorithms
- G Additional experiments on MuJoCo environments

A Handling stochastic value gradients

Thus far we have worked with the original formulation of policy gradients where a policy is a distribution over actions given states. An alternative approach is that taken by stochastic value gradients (Heess et al., 2015), that rely on the reparametrization trick. In this case, a policy is not represented by a distribution over actions but rather by a set of actions. Formally, if ε are random variables drawn from a fixed distribution ν , then policy π is a deterministic map from $\mathcal{S} \times \nu \rightarrow \mathcal{A}$. This corresponds to the functional representation of the policy. The action a chosen by π in state s (when fixing the random variable $\varepsilon = \varepsilon$) is represented as $\pi(s, \varepsilon)$ and

$$J(\pi) = \sum_s d^\pi(s) \int_\varepsilon \nu(\varepsilon) r(s, \pi(s, \varepsilon)) d\varepsilon \quad (10)$$

and Silver et al. (2014) showed that $\frac{\partial J(\pi)}{\partial \pi(s, \varepsilon)} = d^\pi(s) \nabla_a Q^\pi(s, a)|_{a=\pi(s, \varepsilon)}$.

If the policy π is parameterized by model f with parameters θ , then $\pi(s, \varepsilon) = f(\theta, s, \varepsilon)$. If $f(\theta_t, \varepsilon)$ and $f(\theta, \varepsilon)$ are S -dimensional vectors, then Eq. (3) is given as

$$\theta_{t+1} = \arg \min \mathbb{E}_{\varepsilon \sim \nu} \left[- \sum_s d^{\pi_t}(s) f(\theta, s, \varepsilon) \nabla_a Q^{\pi_t}(s, a)|_{a=f(\theta_t, s, \varepsilon)} + \frac{1}{\eta} D_\phi(f(\theta, \varepsilon), f(\theta_t, \varepsilon)) \right]. \quad (11)$$

Similar to Sections 5.1 and 5.2, we will use a mirror map that decomposes across states. Specifically, we choose $D_\phi(\pi, \mu) = \sum_{s \in \mathcal{S}} d^{\pi_t}(s) \|\pi(s) - \mu(s)\|^2$. With this choice, Eq. (11) can be written as:

$$\theta_{t+1} = \arg \max \left[\mathbb{E}_{s \sim d^{\pi_t}} \left[\mathbb{E}_{\varepsilon \sim \nu} \left[f(\theta, s, \varepsilon) \nabla_a Q^{\pi_t}(s, a)|_{a=f(\theta_t, s, \varepsilon)} - \frac{1}{\eta} \|f(\theta, \varepsilon) - f(\theta_t, \varepsilon)\|^2 \right] \right] \right] \quad (12)$$

This formulation is similar to Eq (15) of (Silver et al., 2014), with Q^{π_t} instead of Q^π . Additionally, while the authors justified the off-policy approach with an approximation, our formulation offers guarantees provided η satisfies the condition of Proposition Proposition 1.

B Proofs for Section 4

Proposition 1 (Guarantee on surrogate function). *The surrogate function $\ell_t^{\pi, \phi, \eta}$ is a lower bound of J if and only if $J + \frac{1}{\eta}\phi$ is a convex function of π .*

Proof.

$$\begin{aligned} J(\pi) - \ell_t^{\pi, \phi, \eta}(\pi) &= J(\pi) - J(\pi_t) - \langle \pi - \pi_t, \nabla_\pi J(\pi_t) \rangle + \frac{1}{\eta} D_\phi(\pi, \pi_t) \\ &= J(\pi) - J(\pi_t) - \langle \pi - \pi_t, \nabla_\pi J(\pi_t) \rangle + \frac{1}{\eta} (\phi(\pi) - \phi(\pi_t) - \langle \nabla_\pi \phi(\pi_t), \pi - \pi_t \rangle) \\ &= \left(J + \frac{1}{\eta} \phi \right) (\pi) - \left(J + \frac{1}{\eta} \phi \right) (\pi_t) - \langle \pi - \pi_t, \nabla_\pi \left(J + \frac{1}{\eta} \phi \right) (\pi_t) \rangle. \end{aligned}$$

The last equation is positive for all π and all π_t if and only if $J + \frac{1}{\eta}\phi$ is convex. \square

Theorem 1 (Guaranteed policy improvement for Algorithm 1). *Assume that ℓ_t is β -smooth w.r.t. the Euclidean norm and that η satisfies the condition of Proposition 1. Then, for any $\alpha \leq 1/\beta$, iteration t of Algorithm 1 guarantees $J(\pi_{t+1}) \geq J(\pi_t)$ for any number m of inner-loop updates.*

Proof. Using the update in Algorithm 1 with $\alpha = \frac{1}{\beta}$ and the β -smoothness of $\ell_t(\omega)$, for all $k \in [m-1]$,

$$\ell_t(\omega_{k+1}) \geq \ell_t(\omega_k) + \frac{1}{2\beta} \|\nabla \ell_t(\omega_k)\|^2$$

After m steps,

$$\ell_t(\omega_m) \geq \ell_t(\omega_0) + \frac{1}{2\beta} \sum_{k=0}^{m-1} \|\nabla \ell_t(\omega_k)\|^2$$

Since $\theta_{t+1} = \omega_m$ and $\omega_0 = \theta_t$ in Algorithm 1,

$$\implies \ell_t(\theta_{t+1}) \geq \ell_t(\theta_t) + \frac{1}{2\beta} \|\nabla \ell_t(\theta_t)\|^2 + \sum_{k=1}^{m-1} \|\nabla \ell_t(\omega_k)\|^2$$

Note that $J(\pi_t) = \ell_t(\theta_t)$ and if η satisfies Proposition 1, then $J(\pi_{t+1}) \geq \ell_t(\theta_{t+1})$. Using these relations,

$$J(\pi_{t+1}) \geq J(\pi_t) + \underbrace{\frac{1}{2\beta} \|\nabla \ell_t(\theta_t)\|^2 + \sum_{k=1}^{m-1} \|\nabla \ell_t(\omega_k)\|^2}_{+\text{ve}} \implies J(\pi_{t+1}) \geq J(\pi_t).$$

\square

C Proofs for Section 5

In this section, we first prove the equivalence of the formulations in terms of the logits and in terms of $\log \pi$.

Lemma 1. *Let*

$$\phi(z) = \log \left(\sum_a \exp(z(a)) \right) \quad (13)$$

$$p^\pi(a) = \frac{\exp(z(a))}{\sum_{a'} \exp(z(a'))} \cdot \quad (14)$$

Then

$$D_\phi(z, z') = KL(p^{\pi'} || p^\pi) \cdot \quad (15)$$

where p^π and $p^{\pi'}$ use z and z' respectively.

Proof.

$$\begin{aligned} D_\phi(z, z') &= \log \left(\sum_a \exp(z(a)) \right) - \log \left(\sum_a \exp(z'(a)) \right) - \frac{\sum_a \exp(z'(a))(z(a) - z'(a))}{\sum_a \exp(z'(a))} \\ &= \sum_a p^{\pi'}(a) \left(z(a) - z'(a) + \log \left(\sum_a \exp(z(a)) \right) - \log \left(\sum_a \exp(z'(a)) \right) \right) \\ &= \sum_a p^{\pi'}(a) \log \frac{p^\pi(a)}{p^{\pi'}(a)} . \end{aligned}$$

□

Proposition 4.

$$\ell_t^{z^\pi, \phi, \eta}(\theta) = J(\pi_t) + E_{(s,a) \sim \mu^{\pi_t}} \left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) \log \frac{p^\pi(a|s, \theta)}{p^{\pi_t}(a|s, \theta)} \quad (16)$$

Proof. Because $\sum_a p^{\pi_t}(a|s) A^{\pi_t}(s, a) = 0$, we can shift all values of z by a term that does not depend on a without changing the sum, in particular by $\log(\sum_{a'} \exp(z^\pi(a', s|\theta)))$. Thus,

$$\begin{aligned} \ell_t^{z^\pi, \phi, \eta}(\theta) &= J(\pi_t) + E_{(s,a) \sim \mu^{\pi_t}} A^{\pi_t}(s, a) \left(z^\pi(a, s|\theta) - \log \left(\sum_{a'} \exp(z^\pi(a', s|\theta)) \right) \right) \\ &\quad - \frac{1}{\eta} \sum_s d^{\pi_t}(s) D_{\phi_z}(z^\pi(\cdot, s|\theta), z^\pi(\cdot, s, \theta_t)) \\ &= J(\pi_t) + E_{(s,a) \sim \mu^{\pi_t}} A^{\pi_t}(s, a) \log p^\pi(a|s, \theta) - \frac{1}{\eta} \sum_s d^{\pi_t}(s) D_{\phi_z}(z^\pi(\cdot, s|\theta), z^\pi(\cdot, s, \theta_t)) \\ &= J(\pi_t) + E_{(s,a) \sim \mu^{\pi_t}} A^{\pi_t}(s, a) \log p^\pi(a|s, \theta) - \frac{1}{\eta} \sum_s d^{\pi_t}(s) KL((p^{\pi'}(\cdot|s)||p^\pi(\cdot|s))) , \end{aligned}$$

where the last line is obtained using Lemma 1. Expanding the KL leads to the desired result. □

Proposition 2 (Improvement guarantees for direct functional representation). *Assuming that the rewards are in $[0, 1]$, when using the surrogate function in Eq. (6) with the mirror map chosen to be the negative entropy, then $J \geq \ell_t^{\pi, \phi, \eta}$ for $\eta \leq \frac{(1-\gamma)^3}{2\gamma|A|}$.*

Proof. Agarwal et al. (2020) show that, when using the direct parameterization, J is $\left(\frac{2\gamma|A|}{(1-\gamma)^3}\right)$ -smooth w.r.t. the Euclidean distance. By using the properties of relative smoothness (Lu et al., 2018), if the mirror map ϕ is μ -strongly convex w.r.t. Euclidean distance, then J is L -smooth with $L = \frac{(2\gamma|A|)/(1-\gamma)^3}{\mu}$. Using the fact that negative entropy is 1-strongly convex w.r.t. the 1-norm, we can set $\eta = \frac{(1-\gamma)^3}{2\gamma|A|}$ in Eq. (6). \square

To prove the value of η guaranteeing improvement for the softmax parameterization, we first need to extend a lower bound result from Ghosh et al. (2020):

Lemma 2. *Let us assume that the rewards are lower bounded by $-c$ for some $c \in \mathbb{R}$. Then we have*

$$J(\pi) \geq J(\pi_t) + E_{(s,a) \sim \mu^{\pi_t}} \left[\left(Q^{\pi_t}(s, a) + \frac{c}{1-\gamma} \right) \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} \right]. \quad (17)$$

Proof. Let us define the function J_ν for a policy ν as

$$J_\nu(\pi) = \sum_{h=0}^{+\infty} \gamma^h \int_{\tau_h} (r(s_h, a_h) + c) \left(1 + \log \frac{\pi_h(\tau_h)}{\nu_h(\tau_h)} \right) \nu_h(\tau_h) d\tau_h - \frac{c}{1-\gamma},$$

where τ_h is a trajectory of length h that is a prefix of a full trajectory τ and π_h is the policy restricted to trajectories of length h . We first show that it satisfies $J_\nu(\pi) \leq J(\pi)$ for any ν and any π such that the support of ν covers that of π .

Indeed, we can rewrite

$$\begin{aligned} J(\pi) &= \int_{\tau} \left(R(\tau) + \frac{c}{1-\gamma} \right) \pi(\tau) d\tau - \frac{c}{1-\gamma} \\ &= \int_{\tau} \left(\sum_h \gamma^h (r(a_h, s_h) + c) \right) \pi(\tau) d\tau - \frac{c}{1-\gamma} \quad (\text{using } \sum_h \gamma^h c = c/(1-\gamma)) \\ &= \sum_h \gamma^h \int_{\tau_h} (r(a_h, s_h) + c) \pi_h(\tau_h) d\tau_h - \frac{c}{1-\gamma}, \end{aligned}$$

where the last line is obtained by marginalizing over steps $h+1, \dots, +\infty$ for all h and all trajectories τ . Because $r(a_h, s_h) + c$ is positive, as the rewards are lower bounded by $-c$, we have

$$\begin{aligned} J(\pi) &= \sum_h \gamma^h \int_{\tau_h} (r(a_h, s_h) + c) \frac{\pi_h(\tau_h)}{\nu_h(\tau_h)} \nu_h(\tau_h) d\tau_h - \frac{c}{1-\gamma} \\ &\geq \sum_h \gamma^h \int_{\tau_h} (r(a_h, s_h) + c) \left(1 + \log \frac{\pi_h(\tau_h)}{\nu_h(\tau_h)} \right) \nu_h(\tau_h) d\tau_h - \frac{c}{1-\gamma} \quad (\text{using } x \geq 1 + \log x) \\ &= J_\nu(\pi). \end{aligned}$$

Let us denote J_ν^{SA} the right-hand side of Eq. (17), i.e.:

$$J_\nu^{SA}(\pi) = J(\nu) + E_{(s,a) \sim \mu^\nu} \left[\left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) \log \frac{p^\pi(a|s)}{p^\nu(a|s)} \right].$$

We now prove that J_ν has the same gradient as J_ν^{SA} :

$$\begin{aligned} \nabla_\theta J_\nu(\pi) &= \nabla_\theta \left(\sum_h \gamma^h \int_{\tau_h} (r(a_h, s_h) + c) \left(1 + \log \frac{\pi_h(\tau_h)}{\nu_h(\tau_h)} \right) \nu_h(\tau_h) d\tau_h \right) \\ &= \nabla_\theta \left(\sum_h \gamma^h \int_{\tau_h} (r(a_h, s_h) + c) \log \pi_h(\tau_h) \nu_h(\tau_h) d\tau_h \right) \\ &= \sum_h \gamma^h \int_{\tau_h} (r(a_h, s_h) + c) \nabla_\theta \log \pi_h(\tau_h) \nu_h(\tau_h) d\tau_h, \end{aligned}$$

where all terms independent of θ were moved outside of the gradient. As the log probability of a trajectory decomposes into a sum of the probabilities of actions given states and of the transition probabilities, and as the latter are independent of θ , we get

$$\begin{aligned}\nabla_\theta J_\nu(\pi) &= \sum_h \gamma^h \int_{\tau_h} (r(a_h, s_h) + c) \nabla_\theta \log \pi_h(\tau_h) \nu_h(\tau_h) d\tau_h \\ &= \sum_h \gamma^h \int_{\tau_h} (r(a_h, s_h) + c) \left(\sum_{h'} \nabla_\theta \log p^\pi(a_{h'}|s_{h'}) \right) \nu_h(\tau_h) d\tau_h \\ &= \int_\tau \sum_{h'} \nabla_\theta \log p^\pi(a_{h'}|s_{h'}) \left(\sum_{h=h'}^{+\infty} \gamma^h (r(a_h, s_h) + c) \right) \nu(\tau) d\tau.\end{aligned}$$

But

$$\begin{aligned}\sum_{h=h'}^{+\infty} \gamma^h (r(a_h, s_h) + c) &= \gamma^{h'} \left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) \\ \int_\tau \nu(\tau) d\tau 1_{a_{h'}=a} 1_{s_{h'}=s} &= d_\nu^{h'}(s) \nu(a|s),\end{aligned}$$

with $d_\nu^{h'}(s)$ the undiscounted probability of reaching state s at timestep h' . Hence, we have

$$\begin{aligned}\nabla_\theta J_\nu(\pi) &= \int_\tau \sum_{h'} \nabla_\theta \log p^\pi(a_{h'}|s_{h'}) \left(\sum_{h=h'}^{+\infty} \gamma^h (r(a_h, s_h) + c) \right) \nu(\tau) d\tau \\ &= \sum_{h'} \sum_s \sum_a \nabla_\theta \log p^\pi(a|s) d_\nu^{h'}(s) \nu(a|s) \gamma^{h'} \left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) \\ &= \sum_{h'} \gamma^{h'} \sum_s d_\nu^{h'}(s) \sum_a \nabla_\theta \log p^\pi(a|s) \nu(a|s) \left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) \\ &= \sum_s d^\nu(s) \sum_a \left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) \nu(a|s) \nabla_\theta \log p^\pi(a|s) \\ &= \nabla_\theta \left(\sum_s d^\nu(s) \sum_a \left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) \nu(a|s) \log p^\pi(a|s) \right) \\ &= \nabla_\theta \left(J(\nu) + E_{(s,a) \sim \mu^\nu} \left[\left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) \log \frac{p^\pi(a|s)}{p^\nu(a|s)} \right] \right) \\ &= \nabla_\theta J_\nu^{SA}(\pi),\end{aligned}$$

with $d^\nu(s)$ the unnormalized probability of s under the *discounted* stationary distribution.

Because J_ν and J_ν^{SA} have the same gradient, they differ by a constant, i.e. $J_\nu^{SA} = J_\nu + C$ for some C . But we also know that $J_\nu(\nu) = J(\nu)$, which means that

$$\begin{aligned}C &= J_\nu^{SA}(\nu) - J_\nu(\nu) \\ &= J_\nu^{SA}(\nu) - J(\nu) \\ &= E_{(s,a) \sim \mu^\nu} \left[\left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) \log \frac{p^\nu(a|s)}{p^\nu(a|s)} \right] \\ &= 0.\end{aligned}$$

Hence, $J_\nu = J_\nu^{SA}$ and, becomes J_ν is a lower bound of J , we have

$$J(\pi) \geq J(\nu) + \sum_s d^\nu(s) \sum_a \left(Q^\nu(s, a) + \frac{c}{1-\gamma} \right) p^\nu(a|s) \log \frac{p^\pi(a|s)}{p^\nu(a|s)}. \quad (18)$$

Setting $\nu = \pi_t$ concludes the proof. \square

Proposition 3 (Improvement guarantees for softmax functional representation). *Assuming that the rewards are in $[0, 1]$, then the surrogate function in Eq. (8) satisfies $J \geq \ell_t^{\pi, \phi, \eta}$ for $\eta \leq 1 - \gamma$.*

Proof. Assume

$$\eta = \frac{1 - \gamma}{r_m - r_l} . \quad (19)$$

We know from Proposition 4 that

$$\ell_t^{z^\pi, \phi, \eta}(\theta) \leq J(\pi_t) + E_{(s, a) \sim \mu^{\pi_t}} \left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) \log \frac{p^\pi(a|s, \theta)}{p^{\pi_t}(a|s, \theta)} .$$

Since the rewards are between r_l and r_m , we have

$$\begin{aligned} \ell_t^{z^\pi, \phi, \eta}(\pi) &\leq J(\pi_t) + E_{(s, a) \sim \mu^{\pi_t}} \left[\left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} \right] \\ &= J(\pi_t) + E_{(s, a) \sim \mu^{\pi_t}} \left[\left(A^{\pi_t}(s, a) + \frac{r_m - r_l}{1 - \gamma} \right) \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} \right] \\ &= J(\pi_t) + E_{(s, a) \sim \mu^{\pi_t}} \left[\left(A^{\pi_t}(s, a) + V^{\pi_t}(s) + \left(\frac{r_m}{1 - \gamma} - V^{\pi_t}(s) \right) - \frac{r_l}{1 - \gamma} \right) \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} \right] \\ &= J(\pi_t) + E_{(s, a) \sim \mu^{\pi_t}} \left[\left(Q^{\pi_t}(s, a) - \frac{r_l}{1 - \gamma} \right) \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} \right] \\ &\quad - E_{s \sim d^{\pi_t}} \left[\left(\frac{r_m}{1 - \gamma} - V^{\pi_t}(s) \right) KL(p^{\pi_t}(\cdot|s) || p^\pi(\cdot|s)) \right] . \end{aligned}$$

The last term on the RHS of the last equation is negative. Indeed, because the rewards are less than r_m , the value functions are less than $r_m/(1 - \gamma)$ and $r_m/(1 - \gamma) - V^{\pi_t}(s)$ is positive. As the KL divergences are positive, the product of the two is positive and the whole term is negative because of the minus term. Thus, we have

$$\begin{aligned} \ell_t^{z^\pi, \phi, \eta}(\pi) &\leq J(\pi_t) + E_{(s, a) \sim \mu^{\pi_t}} \left[\left(Q^{\pi_t}(s, a) - \frac{r_l}{1 - \gamma} \right) \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} \right] \\ &\leq J(\pi) . \end{aligned} \quad (\text{by Lemma 2})$$

Hence, choosing $\eta = \frac{1 - \gamma}{r_m - r_l}$ leads to an improvement guarantee. Because our rewards are bounded between 0 and 1, setting $r_m = 1$ and $r_l = 0$ gives $\eta = 1 - \gamma$. This concludes the proof. \square

D Experimental details in the bandit setting

In this section, we detail the experimental setup for the bandit experiments in Section 6.1.

We consider different K -armed Bernoulli bandit problems. For sEXP3, we specialising the update rule in Eq. (8) to this multi-armed bandit case yielding: $p^{\pi_{t+1}}(a) = p^{\pi_t}(a)(1 + \eta A^{\pi_t}(a))$, where η needs to be chosen such that the probabilities are always positive. However, the computing the advantage either requires knowledge of the rewards of all arms, or an estimate thereof. Since EXP3 is an adversarial bandit algorithm and does not exploit the stochasticity in the rewards, to ensure a fair comparison, we cannot use such an estimate and thus replace the advantage with the immediate reward, leading to the final sEXP3 update:

$$p^{\pi_{t+1}}(a) = p^{\pi_t}(a)(1 + \eta \hat{r}_t(a)) ,$$

where $\hat{r}_t(a)$ an estimator of the reward $r_t(a)$ obtained at round t .

For sEXP3, if A_t is the action taken at round t , then we use the importance weighted estimator $\hat{r}_t(a) = \mathbb{I}\{A_t = 1\}r_t(a)/\pi_t(a)$. For EXP3, we consider both the standard importance weighted estimator (referred to as IWEXP3 in the plots) and the loss based importance weighted estimator (referred to as LBIWEXP3 in the plots) for which $\hat{r}_t(a) = \mathbb{I}\{A_t = 1\}(1 - r_t(a))/\pi_t(a)$.

Before describing our experimental setup, we emphasize that there are two different sources of randomness in our experiments. First, we have the *environment seed* that controls the mean rewards in the bandit problem. Considering different environment seeds guarantees that our results are not specific to a particular choice of the rewards. Given a specific bandit problem, since EXP3 and sEXP3 are randomized bandit algorithms, there is a stochasticity in the actions chosen. We can use different *agent seeds* to control the algorithm randomness.

Following the evaluation protocol of (Vaswani et al., 2020), we consider two classes of bandits with different action gaps (difference in the mean rewards) – hard instances ($\Delta = 0.5$) and easy instances ($\Delta = 0.1$). The mean vector defining a Bernoulli bandit is then sampled entry wise (for each arm) from $\mathcal{U}(0.5 - \Delta/2, 0.5 + \Delta/2)$. To obtain the plot in Section 6.1, we run the experiment for 50 different environment seeds and one agent seed. We evaluated the three algorithms for Bernoulli bandits with $K \in \{2, 10, 100\}$ arms and the difficulty of the problem, as determined by the action gap. For each algorithm, we set the step-size via a grid search over $\eta \in \{0.5, 0.05, 0.005, 0.0005, 0.00005\}$. The plot shows the regret corresponding to the step-size with lowest final average regret.

E Experiments in the tabular setting

In this section¹, we study the performance of four different policy gradient (PG) algorithms. Two of these can be directly obtained from the FMA-PG framework: sMDPO (FMA-PG with a softmax policy and log-sum-exp mirror map; see Eq. 8 in the main text) and MDPO (FMA-PG with direct parameterization and a negative entropy mirror map; see Eq. 7 in the main text). And the other two are the existing popular PG algorithms: TRPO and PPO.

Further, to better understand the reason behind the performance of each of these methods, in addition to studying the objective functions used by these PG algorithms, we will also consider the impact of the optimization techniques used to implement them. In particular, we will look at three different variants of sMDPO, MDPO, and TRPO based on whether they use a regularized objective with a fixed step-size (similar to the conventional sMDPO and MDPO), a regularized objective with Armijo line search, or a constrained objective with line search (similar to the conventional TRPO).

E.1 Algorithmic Details

We begin by specifying the different surrogate objectives used by the different algorithms and the two optimization procedures we use for maximizing these objectives. The sMDPO and MDPO algorithms motivated by the FMA-PG framework can be considered as regularized algorithms, which can be summarized by

$$\max_{\theta} \mathcal{J}_{\text{PG-Alg}} - \frac{1}{\eta} \mathcal{C}_{\text{PG-Alg}}, \quad (20)$$

where the terms $\mathcal{J}_{\text{PG-Alg}}$ and $\mathcal{C}_{\text{PG-Alg}}$ are given in Table 1. One way of solving this objective is by gradient descent using a fixed step-size α , as specified in Algorithm 1; we call this setting as **Regularized + fixed step-size**. We can equivalently solve such an unconstrained optimization problem by using an Armijo-style backtracking line search, which we call as **Regularized + line search**. Note that, we can use this same form to obtain a regularized version of the TRPO algorithm² as well.

| PG Alg. | Objective (\mathcal{J}) | Constraint (\mathcal{C}) |
|---------|---|--|
| sMDPO | $\sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a s) A^{\pi_t}(s, a) \log \frac{p^{\pi_\theta}(s, a)}{p^{\pi_t}(s, a)}$ | $\sum_s d^{\pi_t}(s) \cdot \text{KL}(p^{\pi_t}(\cdot s) \ p^{\pi_\theta}(\cdot s))$ |
| TRPO | $\sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a s) Q^{\pi_t}(s, a) \frac{p^{\pi_\theta}(s, a)}{p^{\pi_t}(s, a)}$ | (same as above) |
| MDPO | $\sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a s) A^{\pi_t}(s, a) \frac{p^{\pi_\theta}(s, a)}{p^{\pi_t}(s, a)}$ | $\sum_s d^{\pi_t}(s) \cdot \text{KL}(p^{\pi_\theta}(\cdot s) \ p^{\pi_t}(\cdot s))$ |

Table 1: The objectives and the constraints corresponding to the different PG algorithms. Note that the objective \mathcal{J} for both TRPO and MDPO is essentially equivalent to each other, since maximizing either of them would lead to the same solution. The reason for this is that the difference between the two objectives is $\sum_s d^{\pi_t} V^{\pi_t}(s)$, which is independent of the policy weight θ .

On the other hand, the conventional TRPO algorithm instead solves a constrained optimization problem given by the equation

$$\max_{\theta} \mathcal{J}_{\text{PG-Alg}} \quad \text{subject to} \quad \mathcal{C}_{\text{PG-Alg}} \leq \delta, \quad (21)$$

with the terms $\mathcal{J}_{\text{PG-Alg}}$ and $\mathcal{C}_{\text{PG-Alg}}$ again given in Table 1. The regularized program of Eq. 20 can be considered a “softer” version of the constrained program of Eq. 21. To solve the constrained optimization problem, we use the exact same process used by the TRPO paper (Schulman et al., 2015): we use line search to find the maximal step-size that increases the objective value in the direction of maximum ascent while satisfying the constraint; see Section F.3 for details. We call this setting as **Constrained + line search**. Further, using Eq. 21, we can also obtain constrained versions of sMDPO and MDPO.

¹The code implementation for the algorithms and the environment corresponding to experiments presented in this section is available at <https://github.com/svngrg/fma-pg>.

²For TRPO, this objective is almost the same as PPO with KL penalty (Eq. 8, Schulman et al. (2017)) except that PPO uses the advantage function and we used the action value function (which, as we discussed in the caption of Table 1, doesn’t really matter). It is also similar to the objective stated in the TRPO paper (Section 4, Schulman et al. (2015)) except that this has an average KL divergence instead of the max KL divergence given in the original paper.

The motivation behind considering these three different variants for sMDPO, MDPO, and TRPO is to figure out how much of the performance difference between these algorithms comes from their exact objectives (Table 1) and how much of it comes from the optimization techniques employed. We also summarize the gradient of these objectives in Table 2. The corresponding gradient derivations for the algorithms (including PPO) are presented in Appendix F.

| PG Alg. | Grad. objective ($\nabla_{\theta(s,a)} \mathcal{J}$) | Grad. constraint ($\nabla_{\theta(s,a)} \mathcal{C}$) |
|---------|--|--|
| sMDPO | $d^{\pi_t}(s)p^{\pi_t}(a s)A^{\pi_t}(s,a)$ | $d^{\pi_t}(s)[p^{\pi}(a s) - p^{\pi_t}(a s)]$ |
| TRPO | $d^{\pi_t}(s)p^{\pi}(a s)[Q^{\pi_t}(s,a) - \sum_b p^{\pi}(b s)Q^{\pi_t}(s,b)]$ | (same as above) |
| MDPO | $d^{\pi_t}(s)p^{\pi}(a s)[A^{\pi_t}(s,a) - \sum_b p^{\pi}(b s)A^{\pi_t}(s,b)]$ | $d^{\pi_t}(s)p^{\pi}(a s) \times [\log \frac{p^{\pi}(a s)}{p^{\pi_t}(a s)} - \text{KL}(p^{\pi}(\cdot s) \ p^{\pi_t}(\cdot s))]$ |

Table 2: The gradients of the objectives and constraints w.r.t. the policy parameter corresponding to the different PG algorithms. Note that the gradient of the objective for both TRPO and MDPO is exactly equal to each other.

E.2 Empirical Details

In all our tabular experiments, we assumed full access to the environment dynamics and used the analytically calculated expected gradient updates for all the algorithms, and therefore the results closely follow the theoretical properties of the PG methods. Doing so, essentially made this a study of the optimization properties of the four PG algorithms considered. We use a policy gradient agent with a tabular softmax policy parameterization, and evaluate the algorithms on two tabular episodic environments: CliffWorld (environment description and its properties are discussed in Figure 5) and DeepSeaTreasure (Osband et al. (2019); with $n = 5$, discount factor $\gamma = 0.9$, 25 different states, and two actions).

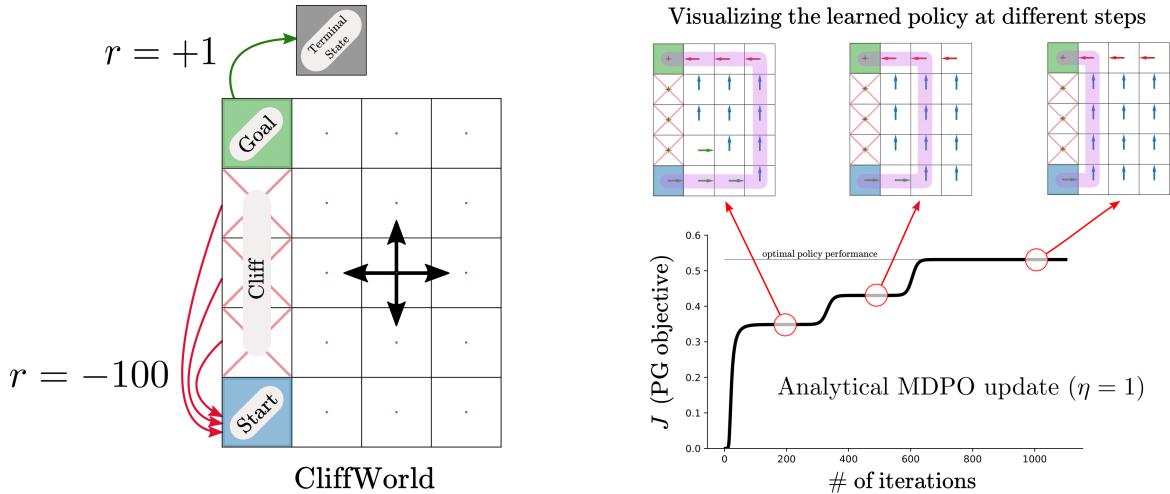


Figure 5: The episodic CliffWorld environment and the learning curve for MDPO on it illustrating three different locally optimal policies. **(Left)** We consider a variant of the CliffWorld environment (Example 6.6, Sutton and Barto (2018)) containing 21 different states and four actions per state. The agent starts in the **Start** state and has four cardinal actions which deterministically move it into the corresponding next state. The objective is to reach the **Goal** state as quickly as possible. If the agent falls into a state marked by **Cliff**, any subsequent action taken by it moves it back to the start state and yields a reward of -100 . Similarly, once in the goal state, any action takes the agent into the terminal state and yields a reward of $+1$. All the other transitions have zero reward and the discount factor is $\gamma = 0.9$. It is easy to see that the optimal policy will have a value of $v^*(s_0) = 0 + \gamma \cdot 0 + \dots + \gamma^5 \cdot 0 + \gamma^6 \cdot 1 = 0.9^6 = 0.53$. **(Right)** We show the learning curve for the analytical MDPO update using $\eta = 1$. This curve shows three different locally optimal policies. We later show in our experiments, that the different PG agents often get stuck on one of these policies.

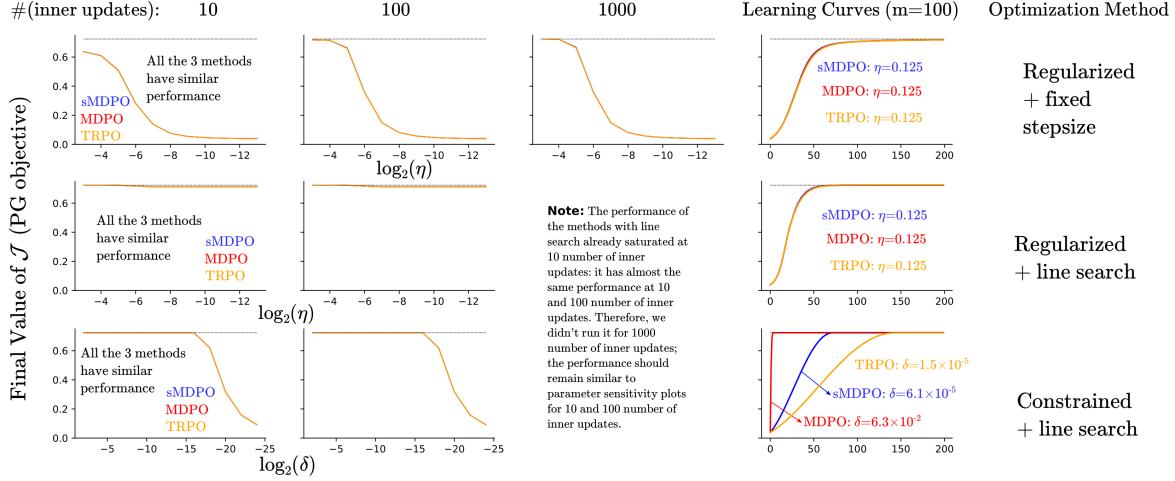


Figure 6: The parameter sensitivity plots for the PG algorithms on the DeepSeaTreasure environment for different number of inner loop updates. The x axis shows sweep over one parameter of the corresponding PG algorithm. And for each point on the x -axis, we chose the best performing second parameter of the algorithm: the inner loop step-size α for the first row, the Armijo constant for the second row, and there is no additional parameter for the last row. The faint black line near the top of each subplot depicts the value of the optimal policy. The last column shows the learning curves for the best performing parameter configuration for each method.

Hyperparameter configurations: We trained each of the method for 2000 iterations for CliffWorld (200 iterations for DeepSeaTreasure). Each iteration consisted of multiple inner loop updates (we represent this number by m); these updates are performed in an off-policy fashion that is typical of all these algorithms (also see Algorithm 1 in the main paper). We also swept over the relevant parameters of the PG algorithms. For the **Regularized** variants (both with and without line search) of sMDPO, MDPO, and TRPO, this was $\eta \in \{2^{-13}, 2^{-12}, \dots, 2^{-1}\}$. For **fixed step-size** variant of sMDPO, MDPO, TRPO, and PPO, we swept over the inner loop step-size $\alpha \in \{2^{-13}, 2^{-12}, \dots, 2^3\}$ for CliffWorld (and $\alpha \in \{2^{-13}, 2^{-12}, \dots, 2^{-2}\}$ for DeepSeaTreasure). For PPO, we additionally considered the clipping parameter $\epsilon \in \{0.01, 0.1, 0.2, 0.3 \dots, 0.8, 0.9, 0.99\}$. For the **Regularized + line search** variant of sMDPO, MDPO, TRPO, we also considered different Armijo constants in the set $\{0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99\}$, used a decay factor of 0.9, initialized the maximal step-size to 10.0 and fixed the warm-start factor to 2.0. Finally, for the **Constrained + line search** variant of sMDPO, MDPO, and TRPO, we swept over the trust region size $\delta \in \{2^{-24}, 2^{-22}, \dots, 2^{-2}\}$, used a fixed backtracking decay parameter of 0.9, an analytically obtained maximal step-size (see Appendix F.3), and an Armijo constant of 0.0 (i.e. no Armijo line search).

E.3 Experimental Results

Learning Curves: We show the learning curves corresponding to the best performing hyperparameters for the four algorithms conventional sMDPO and MDPO (**Regularized + fixed step-size**), conventional TRPO (**Constrained + line search**), and PPO in Figure 2 (main paper). To select the hyperparameters for each setting, we ran sweeps over different configurations and chose the ones that resulted in the best final performance at the end of 2000 iterations for CliffWorld (and 200 iterations for DeepSeaTreasure). From Figure 2, we see that for CliffWorld, all the methods except PPO (PPO got stuck in a “safe” sub-optimal policy) were able to converge to the optimal policy, and TRPO had the fastest convergence (learned the optimal policy in less than 200 iterations). On the other hand for DeepSeaTreasure, we note that all the methods converged to the optimal policy, with PPO having the fastest convergence and TRPO the slowest. Additionally, we should also mention that the TRPO’s update was the costliest (more than two times slower than the rest of the methods) in terms of wall time, likely because of the backtracking from the line-search.

Parameter Sensitivity and Ablation Study: We show the final performance for sMDPO, MDPO, and TRPO in Figure 3 (after 2000 iterations for CliffWorld; main paper) and Figure 6 (after 200 iterations for DeepSeaTreasure). The different rows correspond to the variants **Regularized + fixed step-size**, **Regularized + line search**, and **Constrained + line search** for each of the methods. And the different columns correspond to different

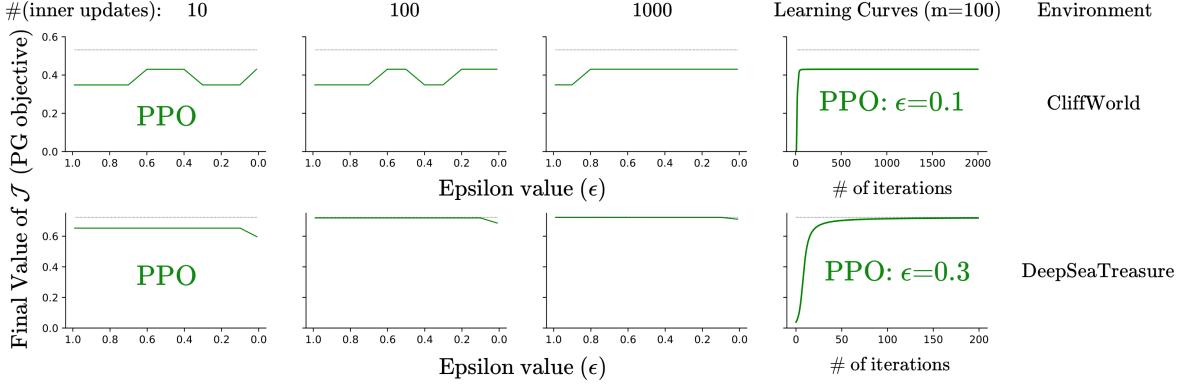


Figure 7: The parameter sensitivity plots for PPO on the CliffWorld and DeepSeaTreasure environments for different number of inner loop updates. The x axis shows sweep over the clipping parameter ϵ . The curve shows the final performance of the method for the best performing inner loop stepsize α given the ϵ value.

number of inner loop updates³. The last column in each row shows the learning curves for the best performing parameter setting. The x -axis on each subplot of the first two rows shows the regularization strength η . For the **Regularization + fixed stepsize** variant, we chose the best performing α for each η , and for **Regularized + line search** variant, we chose the best performing Armijo constant for each η . The last row (constrained variant) had only a single parameter, the trust region magnitude, δ that is shown on the x -axis.

From these figures, we see that as the value of m increased, the performance of the fixed step-size algorithms improved. We also note that adding line search to regularized methods improved their parameter sensitivity to a large extent. Although, for CliffWorld, none of the **Regularized + line search** variant were able to achieve the optimal policy. We believe that the reason for this is that with warm-start the algorithms started using very large stepsizes (as large as 1000), which lead to an early convergence to a locally optimal policy. To verify this further, we tried running these algorithms (experiments not shown here) without warm start and a maximal stepsize of 1.0; this allowed the methods to achieve the optimal policy for a small range of η values, but also made them much more sensitive different values of η . For the constrained version, we see that all the three algorithms achieved the optimal policy and were generally insensitive to the δ values. This is likely because the constrained variant used the (near) optimal steepest ascent direction with the maximal stepsize, achieved via a backtracking line search. Finally, we note that for DeepSeaTreasure, all the methods had essentially the same performance and achieved the optimal policy in each case; we attribute this to the simplicity of the environment coupled with access to the true gradient updates.

We also provide the sensitivity plot for PPO for the two environments in Figure 7. We again see that increasing the number of inner loop updates helps the performance of PPO on both the environments. We also note that for no value of the parameters we tested, did PPO achieve the optimal policy on CliffWorld.

E.4 Discussion

These experiments served to demonstrate three major points:

1. The optimization methods might matter as much as the policy gradient objectives being considered. We found that much of TRPO’s performance came from formalizing the optimization problem as a constrained program and solving it using the optimal descent direction and a stepsize found using line search. In particular, not only did TRPO’s performance suffer when we replaced the constraint with regularization, but the performance of both sMDPO and MDPO also improved significantly when we used TRPO style of optimization on their objectives. Additionally, we found that line search greatly improved the parameter sensitivity of all the algorithms.
2. The optimal η values chosen by the **Regularized + fixed stepsize** variants of sMDPO and MDPO were

³For the **Constrained + line search** of each method, we observed that the performance saturated after $m = 10$; in particular the sensitivity plots are identical for $m = 10$ and $m = 100$. Therefore, the performance at $m = 1000$ should be exactly equivalent to the performance given at $m = 100$, and consequently we skipped running that experiment.

much larger than the values predicted by our theoretical results. For instance, the maximal η values for CliffWorld, as studied by the FMA-PG framework, are

$$\begin{aligned}\eta_{\text{sMDPO}} &= \frac{1 - \gamma}{r_m - r_l} = \frac{1 - 0.9}{100 - (-1)} = 9.9 \times 10^{-4}, \\ \eta_{\text{MDPO}} &= \frac{(1 - \gamma)^3}{(r_m - r_l) \cdot 2\gamma|\mathcal{A}|} = \frac{(1 - 0.9)^3}{101 \times 2 \times 0.9 \times 4} = 1.4 \times 10^{-6}.\end{aligned}$$

Similarly for DeepSeaTreasure, they are

$$\begin{aligned}\eta_{\text{sMDPO}} &= \frac{1 - 0.9}{1 - (-0.01/5)} = 1.0 \times 10^{-2} \\ \eta_{\text{MDPO}} &= \frac{(1 - 0.9)^3}{(1 - (-0.01/5)) \times 2 \times 0.9 \times 2} = 2.8 \times 10^{-4}.\end{aligned}$$

Note that these values of η are extremely small, and while the FMA-PG framework still guarantees policy improvement with these values, the convergence would be much slower than that shown in our experiments. This is natural since these bounds on η are based on the smoothness of the policy objective \mathcal{J} and from optimization literature, we know that such bounds are usually loose. Finally, also note that the optimal η for sMDPO found by the experiments (for instance, that given in Figure 2) is closer to that predicted by the theory, as compared to MDPO.

3. Each of the algorithms benefited from increasing the number of inner loop updates. These off-policy type of updates enables the PG algorithms to “maximally squeeze” out all the information present in the data they have already collected, thereby allowing them to improve their performance without any additional interaction with the environment. This demonstrates the strength of these methods over simpler algorithms, such as REINFORCE (Williams, 1992), which have only a single update per batch of sampled data.

To conclude, our experiments suggest that the FMA-PG framework provides general purpose surrogate functions with policy improvement guarantees, which when combined with existing optimization techniques can yield policy gradient algorithms that are competitive to existing state-of-the-art methods.

F Analytical Updates and Gradient Expressions for tabular PG Algorithms

In this section, we give the calculations for the closed form analytical solutions for sMDPO and MDPO, and the gradient expressions for all the four algorithms employed in our implementation for tabular PG algorithms given in Appendix E.

F.1 sMDPO with Tabular Parameterization

We begin by considering the conventional sMDPO algorithm with a regularized objective.

F.1.1 Closed Form Update with Direct Representation

Our goal is to find the closed form solution to the following optimization problem (from Eq. 8, main paper):

$$\pi_{t+1} = \arg \max_{\pi \in \Pi} \underbrace{\left[\sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a|s) \left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) \log \frac{p^\pi(s, a)}{p^{\pi_t}(s, a)} \right]}_{=: \ell_{\text{sMDPO}}^{\pi_t}}, \quad (22)$$

subject to the constraints on policy p^π . We will solve this problem by assuming the policy $\pi \equiv p^\pi$ as an $|\mathcal{S}| \times |\mathcal{A}|$ table satisfying the standard constraints

$$\begin{aligned} \sum_a p^\pi(a|s) &= 1, \quad \forall s \in \mathcal{S} \\ p^\pi(a|s) &\geq 0, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \end{aligned}$$

We begin by formulating this problem using Lagrange multipliers $\{\lambda_s\}_{s \in \mathcal{S}}$ and $\{\lambda_{s,a}\}_{s,a \in \mathcal{S} \times \mathcal{A}}$ for all states s and actions a :

$$\begin{aligned} \mathcal{L}(p^\pi, \lambda_s, \lambda_{s,a}) &= \sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a|s) \left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} \\ &\quad - \sum_{s,a} \lambda_{s,a} p^\pi(a|s) - \sum_s \lambda_s \left(\sum_a p^\pi(a|s) - 1 \right), \end{aligned} \quad (23)$$

where we abused the notation, in $\mathcal{L}(p^\pi, \lambda_s, \lambda_{s,a})$, by using λ_s to represent the set $\{\lambda_s\}_{s \in \mathcal{S}}$ and $\lambda_{s,a}$ to represent the set $\{\lambda_{s,a}\}_{s,a \in \mathcal{S} \times \mathcal{A}}$. The KKT conditions (Theorem 12.1, Nocedal and Wright (2006)) for this constrained optimization problem can be written as:

$$\nabla_{p^\pi(b|x)} \mathcal{L}(p^\pi, \lambda_s, \lambda_{s,a}) = 0, \quad \forall x \in \mathcal{S}, \forall b \in \mathcal{A} \quad (\text{C1})$$

$$\sum_a p^\pi(a|s) = 1, \quad \forall s \in \mathcal{S} \quad (\text{C2})$$

$$p^\pi(a|s) \geq 0, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \quad (\text{C3})$$

$$\lambda_s \geq 0, \quad \forall s \in \mathcal{S} \quad (\text{C4})$$

$$\lambda_s \left(\sum_a p^\pi(a|s) - 1 \right) = 0, \quad \forall s \in \mathcal{S} \quad (\text{C5})$$

$$\lambda_{s,a} p^\pi(a|s) = 0, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \quad (\text{C6})$$

We now solve this system. Simplifying Eq. C1 for an arbitrary state-action pair (x, b) gives us:

$$\begin{aligned} \nabla_{p^\pi(b|x)} \mathcal{L}(p^\pi, \lambda_s, \lambda_{s,a}) &= d^{\pi_t}(x) p^{\pi_t}(b|x) \left(A^{\pi_t}(x, b) + \frac{1}{\eta} \right) \frac{1}{p^\pi(b|x)} - \lambda_{x,b} - \lambda_x = 0 \\ \Rightarrow p^\pi(b|x) &= \frac{d^{\pi_t}(x) p^{\pi_t}(b|x) (1 + \eta A^{\pi_t}(x, b))}{\eta(\lambda_x + \lambda_{x,b})}. \end{aligned} \quad (24)$$

Let us set

$$\lambda_{s,a} = 0, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \quad (25)$$

Combining Eq. 24 with the second KKT condition gives us

$$\lambda_s = \frac{1}{\eta} \sum_a d^{\pi_t}(s) p^{\pi_t}(a|s)(1 + \eta A^{\pi_t}(s, a)). \quad (26)$$

Therefore, with the standard coverage assumption $d^{\pi_t}(s) > 0$, $p^{\pi}(a|s)$ becomes

$$p^{\pi}(a|s) = \frac{p^{\pi_t}(a|s)(1 + \eta A^{\pi_t}(s, a))}{\sum_b p^{\pi_t}(b|s)(1 + \eta A^{\pi_t}(s, b))}. \quad (27)$$

Note that $d^{\pi_t}(s), p^{\pi_t}(a|s) \geq 0$ for any state-action pair, since they are proper measures. We also need to ensure that

$$1 + \eta A^{\pi_t}(s, a) \geq 0$$

to satisfy the third and fourth KKT conditions. One straightforward way to achieve this is to define $p^{\pi}(a|s) = 0$ whenever $1 + \eta A^{\pi_t}(s, a) < 0$, and accordingly re-define λ_s . This gives us the final solution to our original optimization problem (Eq. 22):

$$\pi_{t+1} = p^{\pi}(s, a) = \frac{p^{\pi_t}(a|s) \max(1 + \eta A^{\pi_t}(s, a), 0)}{\sum_b p^{\pi_t}(b|s) \max(1 + \eta A^{\pi_t}(s, b), 0)}. \quad (28)$$

However, it leaves us one last problem to deal with: ensuring that for any state s , there always exists at least one action a , such that $1 + \eta A^{\pi_t}(s, a) > 0$. This is not a problem since we can put a condition on η in order to fulfill this constraint.

F.1.2 Gradient of the Loss Function with Softmax Policy Representation

Consider the softmax policy representation

$$p^{\pi}(b|x) = \frac{e^{\theta(x, b)}}{\sum_c e^{\theta(x, c)}}, \quad (29)$$

where $\theta(x, b)$ for all state-action pairs (x, b) are action preferences maintained in a table (tabular parameterization). Also note that the derivative of the policy with respect to the action preferences is given by

$$\frac{\partial}{\partial \theta(s, a)} p^{\pi}(b|x) = \mathbb{I}(x = s) (\mathbb{I}(b = a) - p^{\pi}(a|x)) p^{\pi}(b|x), \quad (30)$$

where $\mathbb{I}(a = b)$ is the identity function when $a = b$ and zero otherwise. We will use gradient ascent to approximately solve Eq. 22; to do that, the quantity of interest is

$$\begin{aligned} \frac{\partial}{\partial \theta(s, a)} \ell_{\text{sMDPO}}^{\pi_t} &= \sum_{x \in \mathcal{S}} \sum_{b \in \mathcal{A}} \left[\frac{\partial}{\partial \theta(s, a)} p^{\pi}(b|x) \right] \left[\frac{\partial}{\partial p^{\pi}(b|x)} \ell_{\text{sMDPO}}^{\pi_t} \right] \quad (\text{using total derivative}) \\ &= \sum_{x, b} \left[\mathbb{I}(x = s) (\mathbb{I}(b = a) - p^{\pi}(a|x)) p^{\pi}(b|x) \right] \left[d^{\pi_t}(x) p^{\pi_t}(b|x) \left(A^{\pi_t}(x, b) + \frac{1}{\eta} \right) \frac{1}{p^{\pi}(b|x)} \right] \\ &= \mathbb{E}_{X \sim d^{\pi_t}, B \sim p^{\pi_t}(\cdot|X)} \left[\mathbb{I}(X = s) (\mathbb{I}(B = a) - p^{\pi}(a|x)) \left(A^{\pi_t}(X, B) + \frac{1}{\eta} \right) \right] \quad (31) \\ &= d^{\pi_t}(s) \sum_b (\mathbb{I}(b = a) - p^{\pi}(a|s)) p^{\pi_t}(b|s) \left(A^{\pi_t}(s, b) + \frac{1}{\eta} \right) \\ &= d^{\pi_t}(s) \left[p^{\pi_t}(a|s) \left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) - p^{\pi}(a|s) \sum_b p^{\pi_t}(b|s) \left(A^{\pi_t}(s, b) + \frac{1}{\eta} \right) \right] \\ &= d^{\pi_t}(s) \left[p^{\pi_t}(a|s) \left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) - \frac{p^{\pi}(a|s)}{\eta} \right], \end{aligned}$$

Now we can simply update the inner loop of FMA-PG (Algorithm 1, main paper) via gradient ascent:

$$\theta(s, a) \leftarrow \theta(s, a) + \alpha d^{\pi_t}(s) \left[p^{\pi_t}(a|s) \left(A^{\pi_t}(s, a) + \frac{1}{\eta} \right) - \frac{p^{\pi}(a|s)}{\eta} \right]. \quad (32)$$

F.2 Mirror Descent Policy Optimization (MDPO)

In this section, we study the MDPO type FMA-PG update (Eq. 7 in main paper). We first calculate the analytical solution to that optimization problem, and then calculate its gradient which we use in the experiments. However, in the analysis that follows, we replace the advantage function A^{π_t} with the action-value function Q^{π_t} to make it exactly same as the original MDPO (Tomar et al., 2020) update.

F.2.1 Closed Form Update with Direct Parameterization

While giving the MDPO type FMA-PG equation (Eq. 7), the paper considers the direct representation along with tabular parameterization of the policy, albeit with a small change in notation as compared to the previous subsection: $\pi(a|s) \equiv p^\pi(a|s, \theta)$. However, since this notation is more cumbersome, we will stick with our the notation of the previous subsection: $\pi(a|s) \equiv p^\pi(a|s)$. The constraints on the parameters $p^\pi(s, a)$ are the same as before: $\sum_a p^\pi(a|s) = 1$, $\forall s \in \mathcal{S}$; and $p^\pi(a|s) \geq 0$, $\forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}$. Our goal, this time, is to solve the following optimization problem (from Eq. 6, main paper)

$$\pi_{t+1} = \arg \max_{\pi \in \Pi} \underbrace{\left[\sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a|s) \left(Q^{\pi_t}(s, a) \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} - \frac{1}{\eta} D_\phi(p^\pi(\cdot|s), p^{\pi_t}(\cdot|s)) \right) \right]}_{=: \ell_{\text{MDPO}}^{\pi_t}}, \quad (33)$$

with the mirror map as the negative entropy (Eq. 5.27, Beck and Teboulle (2003)). This particular choice of the mirror map simplifies the Bregman divergence as follows

$$D_\phi(p^\pi(\cdot|s), p^{\pi_t}(\cdot|s)) = \text{KL}(p^\pi(\cdot|s) \| p^{\pi_t}(\cdot|s)) := \sum_a p^\pi(a|s) \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)}. \quad (34)$$

The optimization problem (Eq. 33) then simplifies to

$$\pi_{t+1} = \arg \max_{\pi \in \Pi} \left[\sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a|s) \left(Q^{\pi_t}(s, a) \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} - \frac{1}{\eta} \sum_{a'} p^\pi(a'|s) \log \frac{p^\pi(a'|s)}{p^{\pi_t}(a'|s)} \right) \right]. \quad (35)$$

Proceeding analogously to the previous subsection, we use Lagrange multipliers λ_s , $\lambda_{s,a}$ for all states s and actions a to obtain the function

$$\begin{aligned} \mathcal{L}(p^\pi, \lambda_s, \lambda_{s,a}) &= \sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a|s) Q^{\pi_t}(s, a) \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} - \frac{1}{\eta} \sum_s d^{\pi_t}(s) \sum_{a'} p^\pi(a'|s) \log \frac{p^\pi(a'|s)}{p^{\pi_t}(a'|s)} \\ &\quad - \sum_{s,a} \lambda_{s,a} p^\pi(a|s) - \sum_s \lambda_s \left(\sum_a p^\pi(a|s) - 1 \right). \end{aligned} \quad (36)$$

The KKT conditions are exactly the same as before (Eq. C1 to Eq. C6).

Again, we begin by solving the first KKT condition:

$$\begin{aligned} \nabla_{p^\pi(b|x)} \mathcal{L}(p^\pi, \lambda_s, \lambda_{s,a}) &= d^{\pi_t}(x) p^{\pi_t}(b|x) \frac{Q^{\pi_t}(x, b)}{p^{\pi_t}(b|x)} - \frac{d^{\pi_t}(x)}{\eta} \left[\log \frac{p^\pi(b|x)}{p^{\pi_t}(b|x)} + 1 \right] - \lambda_{x,b} - \lambda_x \\ &= \frac{d^{\pi_t}(x)}{\eta} \left[\eta Q^{\pi_t}(x, b) - \log \frac{p^\pi(b|x)}{p^{\pi_t}(b|x)} - 1 - \frac{\eta(\lambda_{x,b} + \lambda_x)}{d^{\pi_t}(x)} \right] \\ &= 0 \\ \Rightarrow \quad \log \frac{p^\pi(b|x)}{p^{\pi_t}(b|x)} &= \eta Q^{\pi_t}(x, b) - \frac{\eta(\lambda_{x,b} + \lambda_x)}{d^{\pi_t}(x)} - 1 \\ \Rightarrow \quad p^\pi(b|x) &= p^{\pi_t}(b|x) \cdot e^{\eta Q^{\pi_t}(x, b)} \cdot e^{-\frac{\eta(\lambda_{x,b} + \lambda_x)}{d^{\pi_t}(x)} - 1}, \end{aligned} \quad (37)$$

where in the fourth line, we used the assumption that $d^{\pi_t}(x) > 0$ for all states x . We again set

$$\lambda_{s,a} = 0, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \quad (38)$$

And, we put Eq. 37 in the second KKT condition to get

$$e^{-\frac{\eta \lambda_x}{d^{\pi_t}(x)} - 1} = \left(\sum_b p^{\pi_t}(b|x) \cdot e^{\eta Q^{\pi_t}(x,b)} \right)^{-1}. \quad (39)$$

Therefore, we obtain

$$p^\pi(a|s) = \frac{p^{\pi_t}(a|s) \cdot e^{\eta Q^{\pi_t}(s,a)}}{\sum_b p^{\pi_t}(b|s) \cdot e^{\eta Q^{\pi_t}(s,b)}}. \quad (40)$$

This leaves us one last problem to deal with: ensuring $\lambda_s \geq 0$ for all states s . Again, we can set the step-size η to ensure this constraint.

F.2.2 Gradient of the MDPO Loss Function with Tabular Softmax Representation

We again take the softmax policy representation given by Eq. 29, and compute $\nabla_{\theta(s,a)} \ell_{\text{MDPO}}^{\pi_t}$ for the MDPO loss (we substitute Q^{π_t} with A^{π_t} in this calculation):

$$\begin{aligned} \frac{\partial}{\partial \theta(s,a)} \ell_{\text{MDPO}}^{\pi_t} &= \sum_{x,b} \left[\frac{\partial}{\partial \theta(s,a)} p^\pi(b|x) \right] \left[\frac{\partial}{\partial p^\pi(b|x)} \ell_{\text{MDPO}}^{\pi_t} \right] \quad (\text{using total derivative}) \\ &= \sum_{x,b} \left[\mathbb{I}(x=s) \left(\mathbb{I}(b=a) - p^\pi(a|x) \right) p^\pi(b|x) \right] \left[\frac{d^{\pi_t}(x)}{\eta} \left(\eta A^{\pi_t}(x,b) - \log \frac{p^\pi(b|x)}{p^{\pi_t}(b|x)} - 1 \right) \right] \\ &= \frac{d^{\pi_t}(s)}{\eta} \sum_b \left(\mathbb{I}(b=a) - p^\pi(a|s) \right) p^\pi(b|s) \left[\eta A^{\pi_t}(s,b) - \log \frac{p^\pi(b|s)}{p^{\pi_t}(b|s)} - 1 \right] \\ &= \frac{d^{\pi_t}(s)}{\eta} p^\pi(a|s) \left[\eta A^{\pi_t}(s,a) - \eta \sum_b p^\pi(b|s) A^{\pi_t}(s,b) - \log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} + \text{KL}(p^\pi(\cdot|s) \| p^{\pi_t}(\cdot|s)) \right], \end{aligned}$$

where in the last line, we used the fact that

$$\sum_b p^\pi(b|s) \left[\eta A^{\pi_t}(s,b) - \log \frac{p^\pi(b|s)}{p^{\pi_t}(b|s)} - 1 \right] = \eta \sum_b p^\pi(b|s) A^{\pi_t}(s,b) - \text{KL}(p^\pi(\cdot|s) \| p^{\pi_t}(\cdot|s)) - 1.$$

F.3 Trust Region Policy Optimization (TRPO)

At each step of the policy update, TRPO (Eq. 14, Schulman et al. (2015)) solves the following problem:

$$\max_{\theta} \underbrace{\sum_s d^{\pi_t}(s) \sum_a p^{\pi_\theta}(a|s) Q^{\pi_t}(s,a)}_{=: \mathcal{J}_{\text{TRPO}}} \quad \text{subject to} \quad \underbrace{\sum_s d^{\pi_t}(s) \cdot \text{KL}(p^{\pi_t}(\cdot|s) \| p^{\pi_\theta}(\cdot|s))}_{=: \mathcal{C}_{\text{TRPO}}} \leq \delta. \quad (41)$$

Unlike the sMDPO and the MDPO updates, an analytical solution cannot be derived for this update (since it would require solving a system of non-trivial non-linear equations). Therefore, we will use gradient based methods to approximately solve this problem. From Appendix C of Schulman et al. (2015), the descent direction is given by $s \approx A^{-1}g$ where the vector g is defined as $g_{(s,a)} := \frac{\partial}{\partial \theta(s,a)} \mathcal{J}_{\text{TRPO}}$, and the matrix A is defined as $A_{(s,a),(s',a')} := \frac{\partial}{\partial \theta(s,a)} \frac{\partial}{\partial \theta(s',a')} \mathcal{C}_{\text{TRPO}}$. We analytically compute the expression for this direction assuming a softmax policy (Eq. 29). The vector g can be readily calculated as

$$\begin{aligned} \frac{\partial}{\partial \theta(s,a)} \mathcal{J}_{\text{TRPO}} &= \sum_x d^{\pi_t}(x) \sum_b Q^{\pi_t}(x,b) \frac{\partial p^{\pi_\theta}(b|x)}{\partial \theta(s,a)} \\ &= \sum_x d^{\pi_t}(x) \sum_b Q^{\pi_t}(x,b) \mathbb{I}(x=s) \left(\mathbb{I}(b=a) - p^{\pi_\theta}(a|x) \right) p^{\pi_\theta}(b|x) \\ &= \sum_x d^{\pi_t}(x) \mathbb{I}(x=s) \left[\sum_b \mathbb{I}(b=a) p^{\pi_\theta}(b|x) Q^{\pi_t}(x,b) - p^{\pi_\theta}(a|x) \sum_b p^{\pi_\theta}(b|x) Q^{\pi_t}(x,b) \right] \\ &= d^{\pi_t}(s) p^{\pi_\theta}(a|s) \left[Q^{\pi_t}(s,a) - \sum_b p^{\pi_\theta}(b|s) Q^{\pi_t}(s,b) \right]. \end{aligned} \quad (42)$$

For calculating the matrix A , we use the law of total derivative to obtain

$$\begin{aligned}
\frac{\partial}{\partial \theta(s, a)} \mathcal{C}_{\text{TRPO}} &= \sum_{x, b} \left[\frac{\partial}{\partial \theta(s, a)} p^{\pi_\theta}(b|x) \right] \left[\frac{\partial}{\partial p^{\pi_\theta}(b|x)} \sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a|s) \log \frac{p^{\pi_t}(a|s)}{p^{\pi_\theta}(a|s)} \right] \\
&= \sum_{x, b} \left[\mathbb{I}(x = s) \left(\mathbb{I}(b = a) - p^{\pi_\theta}(a|x) \right) p^{\pi_\theta}(b|x) \right] \left[-d^{\pi_t}(x) \frac{p^{\pi_t}(b|x)}{p^{\pi_\theta}(b|x)} \right] \\
&= -d^{\pi_t}(s) \sum_b \left(\mathbb{I}(b = a) - p^{\pi_\theta}(a|s) \right) p^{\pi_t}(b|s) \\
&= -d^{\pi_t}(s) \left[\sum_b \mathbb{I}(b = a) p^{\pi_t}(b|s) - p^{\pi_\theta}(a|s) \sum_b p^{\pi_t}(b|s) \right] \\
&= d^{\pi_t}(s) \left[p^{\pi_\theta}(a|s) - p^{\pi_t}(a|s) \right]. \tag{43}
\end{aligned}$$

Finally, using the above result yields

$$\begin{aligned}
\frac{\partial}{\partial \theta(s, a)} \frac{\partial}{\partial \theta(s', a')} \mathcal{C}_{\text{TRPO}} &= \frac{\partial}{\partial \theta(s, a)} d^{\pi_t}(s') \left[p^{\pi_\theta}(a'|s') - p^{\pi_t}(a'|s') \right] \\
&= d^{\pi_t}(s') \cdot \frac{\partial}{\partial \theta(s, a)} p^{\pi_\theta}(a'|s') \\
&= \mathbb{I}(s' = s) \cdot d^{\pi_t}(s') \left(\mathbb{I}(a' = a) - p^{\pi_\theta}(a|s') \right) p^{\pi_\theta}(a'|s') \tag{44}
\end{aligned}$$

$$\Rightarrow A_{(s,:),(s,:)} = d^{\pi_t}(s) \left(\text{diag}(p^{\pi_\theta}(\cdot|s)) - p^{\pi_\theta}(\cdot|s) p^{\pi_\theta}(\cdot|s)^\top \right), \tag{45}$$

where $p^{\pi_\theta}(\cdot|s) \in \mathbb{R}^{|\mathcal{A}|}$ is the vector defined as $[p^{\pi_\theta}(\cdot|s)]_a = p^{\pi_\theta}(a|s)$ and $A_{(s,:),(s,:)}$ denotes the square sub-block of the matrix A corresponding to the given state s and all the actions. In our experiments, since our A matrix is small, we directly take its inverse to compute the update direction, thereby bypassing the conjugate method. Once we have the update direction, we then compute the maximal stepsize β and perform a backtracking line search similar to the TRPO paper.

F.4 Proximal Policy Optimization (PPO)

The Proximal Policy Optimization algorithm (Schulman et al., 2017) solves the following optimization problem at each iteration step:

$$\max_{\theta} \underbrace{\sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a|s) \cdot \min \left(\frac{p^{\pi_\theta}(a|s)}{p^{\pi_t}(a|s)} A^{\pi_t}(s, a), \text{clip} \left[\frac{p^{\pi_\theta}(a|s)}{p^{\pi_t}(a|s)}, 1 - \epsilon, 1 + \epsilon \right] A^{\pi_t}(s, a) \right)}_{=: \mathcal{J}_{\text{PPO}}}. \tag{46}$$

The gradient of the objective \mathcal{J}_{PPO} can be shown to be equivalent to

$$\nabla \mathcal{J}_{\text{PPO}} = \sum_s d^{\pi_t}(s) \sum_a p^{\pi_t}(a|s) \cdot \mathbb{I}(\text{cond}(s, a)) \frac{\nabla p^{\pi_\theta}(a|s)}{p^{\pi_t}(a|s)} A^{\pi_t}(s, a), \tag{47}$$

where

$$\text{cond}(s, a) = \left(A^{\pi_t}(s, a) > 0 \wedge \frac{p^{\pi_\theta}(a|s)}{p^{\pi_t}(a|s)} < 1 + \epsilon \right) \vee \left(A^{\pi_t}(s, a) < 0 \wedge \frac{p^{\pi_\theta}(a|s)}{p^{\pi_t}(a|s)} > 1 - \epsilon \right). \tag{48}$$

Repeating our usual drill, we assume a softmax policy to obtain:

$$\begin{aligned}
 & \frac{\partial}{\partial \theta(s, a)} \mathcal{J}_{\text{PPO}} \\
 &= \sum_x d^{\pi_t}(x) \sum_b \mathbb{I}(\text{cond}(x, b)) \frac{\partial p^{\pi_\theta}(b|x)}{\partial \theta(s, a)} A^{\pi_t}(x, b) \\
 &= \sum_x d^{\pi_t}(x) \sum_b \mathbb{I}(\text{cond}(x, b)) \mathbb{I}(x = s) (\mathbb{I}(b = a) - p^{\pi_\theta}(a|x)) p^{\pi_\theta}(b|x) A^{\pi_t}(x, b) \\
 &= d^{\pi_t}(s) \left[\sum_b \mathbb{I}(b = a) \mathbb{I}(\text{cond}(s, b)) p^{\pi_\theta}(b|s) A^{\pi_t}(s, b) - p^{\pi_\theta}(a|s) \sum_b \mathbb{I}(\text{cond}(s, b)) p^{\pi_\theta}(b|s) A^{\pi_t}(s, b) \right] \\
 &= d^{\pi_t}(s) p^{\pi_\theta}(a|s) \left[\mathbb{I}(\text{cond}(s, a)) A^{\pi_t}(s, a) - \sum_b p^{\pi_\theta}(b|s) \mathbb{I}(\text{cond}(s, b)) A^{\pi_t}(s, b) \right]. \tag{49}
 \end{aligned}$$

The PPO gradient (Eq. 49) is exactly the same as the TRPO gradient (Eq. 42) except for the additional condition on choosing only specific state-action pairs while calculating the difference between advantage under the current policy and the approximate change in advantage under the updated policy.

F.5 MDPO with Constraints

In this section, we calculate the second derivative of the MDPO constraint as given in Table 1. This will allow us compute the Hessian A^{MDPO} , which is the analog of the A matrix from TRPO implementation, and help us implement MDPO with a constrained objective and line search.

Continuing from the gradient of the MDPO constraint given in Table 2, we get

$$\begin{aligned}
 \frac{\partial}{\partial p^\pi(b|x)} \frac{\partial \mathcal{C}_{\text{MDPO}}}{\partial \theta(s', a')} &= \frac{\partial}{\partial p^\pi(b|x)} d^{\pi_t}(s') p^\pi(a'|s') \left(\log \frac{p^\pi(a'|s')}{p^{\pi_t}(a'|s')} - \sum_c p^\pi(c|s') \log \frac{p^\pi(c|s')}{p^{\pi_t}(c|s')} \right) \\
 &= \mathbb{I}(x = s') d^{\pi_t}(s') \left[\mathbb{I}(b = a') \left(\log \frac{p^\pi(a'|s')}{p^{\pi_t}(a'|s')} - \sum_c p^\pi(c|s') \log \frac{p^\pi(c|s')}{p^{\pi_t}(c|s')} \right) \right. \\
 &\quad \left. + p^\pi(a'|s') \frac{\mathbb{I}(b = a')}{p^\pi(a'|s')} - p^\pi(a'|s') \left(\log \frac{p^\pi(b|s')}{p^{\pi_t}(b|s')} + 1 \right) \right] \\
 &= \mathbb{I}(x = s') d^{\pi_t}(s') \left[\mathbb{I}(b = a') \cdot T_{\text{itd}}(s', a') - p^\pi(a'|s') \left(\log \frac{p^\pi(b|s')}{p^{\pi_t}(b|s')} + 1 \right) \right], \tag{50}
 \end{aligned}$$

where we introduced an intermediate variable $T_{\text{itd}}(s', a') := \log \frac{p^\pi(a'|s')}{p^{\pi_t}(a'|s')} - \text{KL}(p^\pi(\cdot|s') \| p^{\pi_t}(\cdot|s')) + 1$. Now, using the law of total derivative, we obtain

$$\begin{aligned}
 \frac{\partial}{\partial \theta(s, a)} \frac{\partial \mathcal{C}_{\text{MDPO}}}{\partial \theta(s', a')} &= \sum_{x,b} \frac{\partial p^\pi(b|x)}{\partial \theta(s, a)} \times \frac{\partial}{\partial p^\pi(b|x)} \frac{\partial \mathcal{C}_{\text{MDPO}}}{\partial \theta(s', a')} \\
 &= \sum_{x,b} \mathbb{I}(x = s) [\mathbb{I}(b = a) - p^\pi(a|x)] p^\pi(b|x) \times \mathbb{I}(x = s') d^{\pi_t}(s') \\
 &\quad \times \left[\mathbb{I}(b = a') \cdot T_{\text{itd}}(s', a') - p^\pi(a'|s') \left(\log \frac{p^\pi(b|s')}{p^{\pi_t}(b|s')} + 1 \right) \right] \\
 &= \mathbb{I}(s = s') d^{\pi_t}(s) \cdot T_{\text{aux}}, \tag{51}
 \end{aligned}$$

where the auxillary term T_{aux} is

$$\begin{aligned}
T_{\text{aux}} &:= \sum_b [\mathbb{I}(b = a) - p^\pi(a|s)] p^\pi(b|s) \left[\mathbb{I}(b = a') \cdot T_{\text{itd}}(s, a') - p^\pi(a'|s) \left(\log \frac{p^\pi(b|s)}{p^{\pi_t}(b|s)} + 1 \right) \right] \quad (52) \\
&= T_{\text{itd}}(s, a') \sum_b \mathbb{I}(b = a) p^\pi(b|s) \mathbb{I}(b = a') - p^\pi(a'|s) \sum_b \mathbb{I}(b = a) p^\pi(b|s) \left(\log \frac{p^\pi(b|s)}{p^{\pi_t}(b|s)} + 1 \right) \\
&\quad - p^\pi(a|s) T_{\text{itd}}(s, a') \sum_b p^\pi(b|s) \mathbb{I}(b = a') + p^\pi(a'|s) p^\pi(a|s) \sum_b p^\pi(b|s) \left(\log \frac{p^\pi(b|s)}{p^{\pi_t}(b|s)} + 1 \right) \\
&= T_{\text{itd}}(s, a') p^\pi(a|s) \mathbb{I}(a = a') - p^\pi(a'|s) p^\pi(a|s) \left(\log \frac{p^\pi(a|s)}{p^{\pi_t}(a|s)} + 1 \right) \\
&\quad - p^\pi(a|s) T_{\text{itd}}(s, a') p^\pi(a'|s) + p^\pi(a'|s) p^\pi(a|s) (\text{KL}(p^\pi(\cdot|s) \| p^{\pi_t}(\cdot|s)) + 1) \\
&= \mathbb{I}(a = a') p^\pi(a|s) T_{\text{itd}}(s, a') - p^\pi(a'|s) p^\pi(a|s) [T_{\text{itd}}(s, a') + T_{\text{itd}}(s, a)] + p^\pi(a'|s) p^\pi(a|s). \quad (53)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\partial}{\partial \theta(s, a)} \frac{\partial \mathcal{C}_{\text{MDPO}}}{\partial \theta(s', a')} &= \mathbb{I}(s = s') d^{\pi_t}(s) \left[\mathbb{I}(a = a') p^\pi(a|s) T_{\text{itd}}(s, a') - p^\pi(a|s) p^\pi(a'|s) T_{\text{itd}}(s, a') \right. \\
&\quad \left. - p^\pi(a'|s) p^\pi(a|s) T_{\text{itd}}(s, a) + p^\pi(a'|s) p^\pi(a|s) \right] \quad (54)
\end{aligned}$$

$$\Rightarrow A_{(s,:),(s,:)}^{\text{MDPO}} = d^{\pi_t}(s) \cdot \left[\text{diag}(T_{\text{vec}}(s)) - p^\pi(\cdot|s) T_{\text{vec}}(s)^\top \right. \\
\left. - T_{\text{vec}}(s) p^\pi(\cdot|s)^\top + p^\pi(\cdot|s) p^\pi(\cdot|s)^\top \right], \quad (55)$$

where we introduced yet another intermediate term $T_{\text{vec}}(s)$, defined as

$$T_{\text{vec}}(s) := p^\pi(\cdot|s) \odot T_{\text{itd}}(s, \cdot) \quad (56)$$

$$= p^\pi(\cdot|s) \odot \left[\log \left(p^\pi(\cdot|s) \oslash p^{\pi_t}(\cdot|s) \right) - \text{KL}(p^\pi(\cdot|s) \| p^{\pi_t}(\cdot|s)) \mathbf{1}_{|\mathcal{A}|} + \mathbf{1}_{|\mathcal{A}|} \right], \quad (57)$$

and \oslash in the above equation represents the elementwise vector division defined as $[a \oslash b]_i := a_i/b_i$ for any two vectors a and b . As a sanity check, note that the matrix A^{MDPO} is symmetric, as any Hessian matrix should be.

G Additional experiments on MuJoCo environments

In this section, we present results on a series of MuJoCo environments where learning rate decay and gradient clipping have *not* been applied. Figure 8 shows that, while sPPO (in orange) still learns something, PPO is unable to make progress, regardless of the capping (ϵ) and the number of inner loop steps (m), further reinforcing our intuition that the softmax parameterization leads to a more robust optimization.

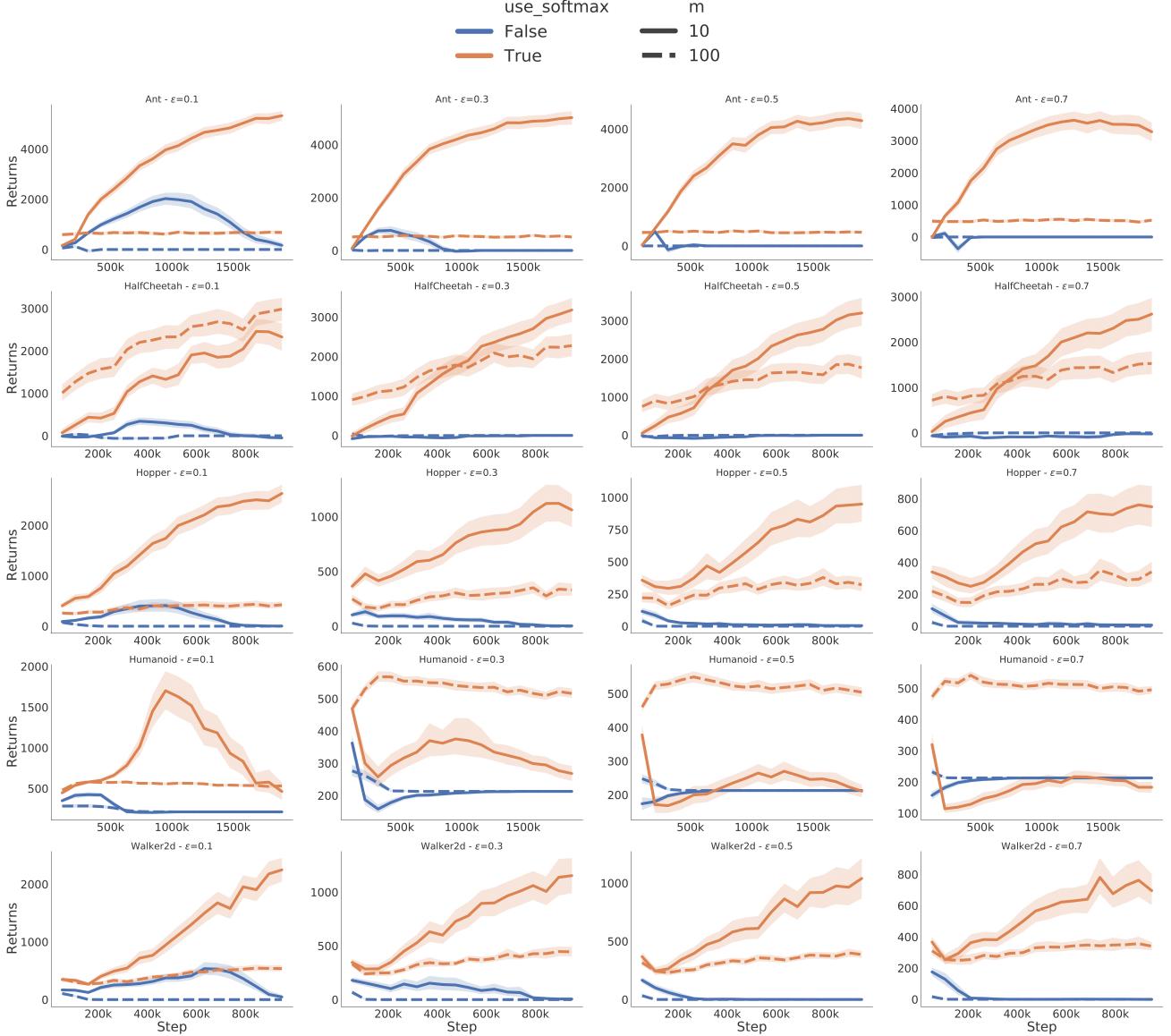


Figure 8: Average discounted return and 95% confidence interval (over 180 runs) for **PPO** and **softmax PPO** on 4 environments (*env* - rows) and for four different clipping strengths (*epsilon* - columns). We see that sPPO is more robust to large values of clipping, even more so when the number of updates in the inner loop grows (linestyle).