EXPLORATION IN POLICY MIRROR DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy optimization is a core problem in reinforcement learning. In this paper, we investigate Reversed Entropy Policy Mirror Descent (REPMD), an on-line policy optimization strategy that improves exploration behavior while assuring monotonic progress in a principled objective. REPMD conducts a form of maximum entropy exploration within a mirror descent framework, but uses an alternative policy update with a reversed KL projection. This modified formulation bypasses undesirable mode seeking behavior and avoids premature convergence to sub-optimal policies, while still supporting strong theoretical properties such as guaranteed policy improvement. An experimental evaluation demonstrates that this approach significantly improves practical exploration and surpasses the empirical performance of state-of-the art policy optimization methods in a set of benchmark tasks.

1 INTRODUCTION

Model-free deep reinforcement learning (RL) has recently been shown to be remarkably effective in solving challenging sequential decision making problems (Schulman et al., 2015; Mnih et al., 2015; Silver et al., 2016). A central method of deep RL is *policy optimization* (aka policy gradient), which is based on formulating the problem as the optimization of a stochastic objective (expected return) with respect to the underlying policy parameters (Williams & Peng, 1991; Williams, 1992; Sutton et al., 1998). Unlike standard optimization, stochastic optimization requires the objective and gradient to be *estimated* from data, typically gathered from a process depending on current parameters, simultaneously with parameter updates. Such an interaction between estimation and updating complicates the optimization process, and often necessitates the introduction of variance reduction methods, leading to algorithms with subtle hyperparameter sensitivity. Joint estimation and updating can also create poor local optima whenever sampling neglect of some region can lead to further entrenchment of a current solution. For example, a non-exploring policy might fail to sample from high reward trajectories, preventing any further improvement since no useful signal is observed. In practice, it is well known that successful application of deep RL techniques requires a combination of extensive hyperparameter tuning, and a large, if not impractical, number of sampled trajectories. It remains a major challenge to develop methods that can reliably perform policy improvement while maintaining sufficient exploration and avoiding poor local optima, yet do so quickly.

Several ideas for improving policy optimization have been proposed in the literature, generally focusing on the goals of improving stability and data efficiency (Peters et al., 2010; Van Hoof et al., 2015; Fox et al., 2015; Schulman et al., 2015; Montgomery & Levine, 2016; Nachum et al., 2017b;c; Tangkaratt et al., 2017; Abdolmaleki et al., 2018; Haarnoja et al., 2018). Unfortunately, a notable gap remains between empirically successful methods and their underlying theoretical support. Current analyses typically assume a simplified setting that either ignores the policy parametrization or only considers linear models. These assumptions are hard to justify when current practice relies on complex function approximators, such as deep neural networks, that are highly nonlinear in their underlying parameters. This gulf between theory and practice is a barrier to wider adoption of model-free policy gradient methods.

In this paper, we focus on a setting where the policy can be a *non-convex* function of its parameters. We begin by considering the entropy-regularized expected reward objective, which has recently re-emerged as a foundation for state-of-the-art RL methods (Williams & Peng, 1991; Fox et al., 2015; Schulman et al., 2017a; Nachum et al., 2017b; Haarnoja et al., 2017). Our first contribution is to reformulate the maximization of this objective as a lift-and-project procedure, following the lines of Mirror Descent (Nemirovskii et al., 1983; Beck & Teboulle, 2003). Such a reformulation achieves

two things. First, it makes it easier to analyze policy optimization in the parameter space: using this reformulation we can establish a monotonic improvement guarantee with a fairly simple proof, even in a non-convex setting. We also provide a study of the fixed point properties of this setup. Second, the proposed reformulation has practical algorithmic consequences, suggesting, for example, that multiple gradient descent updates should be performed in the projection step. These considerations lead to our first practical algorithm, Policy Mirror Descent (PMD), which first lifts the policy to the entire policy-simplex, ignoring the constraint induced by its parametrization, then approximately solves the projection step by multiple gradient descent updates to the policy in the parameter space.

We then investigate additional modifications to mitigate the potential deficiencies of PMD. The main algorithm we propose, Reversed Entropy PMD (REPMD), incorporates both an entropy and relative entropy regularizer, and uses the mean seeking direction of KL divergence for projection. The benefit of this approach is twofold. First, using just the mean seeking direction of KL divergence for the projection step helps avoids poor local optima; second, this specific problem can now be efficiently solved to global optimality in certain non-trivial *non-convex* scenarios, such as one-layer-softmax networks. Similar guarantees can be proved for REPMD, which additionally incorporates entropy regularization, with respect to a surrogate objective $\mathrm{SR}(\pi)$. We further study the properties of $\mathrm{SR}(\pi)$ and provide theoretical and empirical evidence that SR can effectively guide the search for good policies. Finally, we also show how this algorithm can be extended with a value function approximator, and develop an actor-critic version that is effective in practice.

1.1 Notation and Problem Setting

For simplicity, we only consider finite horizon settings with finite state and action spaces. The behavior of an agent is modelled by a policy $\pi(a|s)$ that specifies a probability distribution over a finite set of actions given an observed state. At each time step t, the agent takes an action a_t by sampling from $\pi(a_t|s_t)$. The environment then returns a reward $r_t = r(s_t, a_t)$ and the next state $s_{t+1} = f(s_t, a_t)$, where f is the transition function not revealed to the agent. Given a trajectory, a sequence of states and actions $\rho = (s_1, a_1, \ldots, a_{T-1}, s_T)$, the policy probability and the total reward of ρ are defined as $\pi(\rho) = \prod_{t=1}^{T-1} \pi(a_t|s_t)$ and $r(\rho) = \sum_{t=1}^{T-1} r(s_t, a_t)$. Given a set of parametrized policy functions $\pi_\theta \in \Pi$, policy optimization aims to find the optimal policy π_θ^* by maximizing the expected reward,

$$\pi_\theta^* \in \arg\max_{\pi_\theta \in \Pi} \mathop{\mathbb{E}}_{\rho \sim \pi_\theta} r(\rho), \tag{1}$$

We use $\Delta \triangleq \{\pi | \sum_\rho \pi(\rho) = 1, \pi(\rho) \geq 0, \forall \rho\}$ to refer to the probability simplex over all possible trajectories. Without loss of generality, we also assume that the state transition function is deterministic, and the discount factor $\gamma = 1$. This same simplification is also assumed in Nachum et al. (2017a). Results for the case of a stochastic state transition function are presented in Appendix D.

2 Policy Mirror Descent

We begin with the development of the basic Policy Mirror Descent (PMD) strategy, which will form the basis for our subsequent algorithms and their analyses. As mentioned in the introduction, our analysis of this and subsequent methods focuses on the *non-convex* setting.

Consider the following local optimization problem: given a *reference policy* $\bar{\pi}$ (usually the current policy), maximize the prox regularized expected reward, using relative entropy as the regularizer:

$$\pi_\theta = \arg\max_{\pi_\theta \in \Pi} \mathop{\mathbb{E}}_{\rho \sim \pi_\theta} r(\rho) - \tau D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}). \tag{2}$$

Relative entropy regularization has been widely investigated in online learning and constrained optimization (Nemirovskii et al., 1983; Beck & Teboulle, 2003), primarily as a component of the mirror descent algorithm. Observe that when $\bar{\pi}$ is the uniform distribution, Eq. (2) reduces to entropy regularized expected reward. It is important to note that, since we are interested in the non-convex setting and only assume that π is parametrized as a smooth function of $\theta \in \mathbb{R}^d$, Π is generally a non-convex subset of the simplex. Therefore, Eq. (2) is a difficult constrained optimization problem.

A useful way to decompose this constrained optimization is to consider an alternating lift-and-project procedure that isolates the different computational challenges.

$$\begin{aligned} &\textbf{(Project Step)} \quad \arg\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}_\tau^*), \\ &\textbf{(Lift Step)} \quad \text{where } \bar{\pi}_\tau^* = \arg\max_{\pi \in \Delta} \mathop{\mathbb{E}}_{\rho \sim \pi} r(\rho) - \tau D_{\mathrm{KL}}(\pi \| \bar{\pi}). \end{aligned} \tag{3}$$

Crucially, the reformulation Eq. (3) remains equivalent to the original problem Eq. (2), in that it preserves the same set of solutions, as established in Proposition 1.

Proposition 1. *Given a* reference policy $\bar{\pi}$*,*

$$\arg\max_{\pi_\theta \in \Pi} \mathop{\mathbb{E}}_{\rho \sim \pi_\theta} r(\rho) - \tau D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}) = \arg\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}_\tau^*).$$

Note that this result holds even for the non-convex setting. The reformulation Eq. (3) immediately leads to our PMD algorithm: Lift the current policy π_{θ_t} to $\bar{\pi}_\tau^*$, then perform multiple steps of gradient descent in the Project Step to update $\pi_{\theta_{t+1}}$.[1]

When Π is a convex set, one can show that PMD asymptotically converges to the optimal policy (Nemirovskii et al., 1983; Beck & Teboulle, 2003). The next proposition shows that despite the non-convexity of Π, PMD still enjoys desirable properties.

Proposition 2. *Let π_{θ_t} denote the policy produced at step t of the update sequence. Then PMD satisfies the following properties for an arbitrary parametrization of π.*

1. **(Monotonic Improvement Guarantee)** *If the Project Step $\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}_\tau^*)$ can be solved globally optimally, then*

$$\mathbb{E}_{\rho \sim \pi_{\theta_{t+1}}} r(\rho) - \mathbb{E}_{\rho \sim \pi_{\theta_t}} r(\rho) \geq 0.$$

2. **(Global Optimum Guarantee)** *If the Project Step $\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}_\tau^*)$ can be solved globally optimally, then π_{θ_t} converges to the global optimum π^*, i.e.*

$$\lim_{t \to \infty} \mathbb{E}_{\rho \sim \pi_{\theta_t}} r(\rho) = \mathbb{E}_{\rho \sim \pi^*} r(\rho) \geq \mathbb{E}_{\rho \sim \pi} r(\rho), \quad \forall \pi \in \Pi.$$

3. **(Fixed Points)** *Assume that the Project Step is optimized by gradient descent, then the fixed points of PMD are the stationary points of the expected reward $\mathbb{E}_{\rho \sim \pi_\theta} r(\rho)$.*

Despite these desirable properties, Proposition 2 relies on the condition that the Project Step in PMD is solved to global optimality. It is usually not practical to achieve such a stringent requirement when π_θ is not convex in θ, limiting the applicability of Proposition 2.

Another shortcoming with this naive strategy is that PMD typically gets trapped in poor local optima. Indeed, while the relative entropy regularizer help prevent a large policy update, it also tends to limit exploration. Moreover, minimizing the KL divergence $D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}_\tau^*)$ is known to be *mode seeking* (Murphy, 2012), which can lead to mode collapse during the learning process. Once a policy $\bar{\pi}_\tau^*$ has lost important modes, learning can easily become trapped at a sub-optimal policy. Unfortunately, at such points, the relative entropy regularizer does not encourage further exploration.

3 Reversed Entropy Policy Mirror Descent

We now introduce two modifications to PMD that overcome its aforementioned deficiencies. These two modifications lead to our main proposed algorithm, Reversed Entropy Policy Mirror Descent (REPMD), which retains desirable theoretical properties while achieving vastly superior performance to PMD in practice. (We consider some additional refinements to REPMD in the next section.)

[1] To estimate this gradient one would need to use self-normalized importance sampling Owen (2013). We omit the details here since PMD is not our main algorithm; similar techniques can be found in the implementation of REPMD.

The first modification is to add an additional entropy regularizer to the Lift Step, to improve the exploration behavior of the algorithm. The second modification is to use a reversed, *mean seeking* direction of the KL divergence in the Project Step. In particular, the REPMD algorithm solves the following alternating optimization problems to update the policy $\pi_{\theta_{t+1}}$ at each iteration:

$$\begin{aligned} &\textbf{(Project Step)} \quad \arg\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}^*_{\boldsymbol{\tau},\boldsymbol{\tau}'} \| \boldsymbol{\pi}_{\boldsymbol{\theta}}), \\ &\textbf{(Lift Step)} \quad \text{where } \bar{\pi}^*_{\tau,\tau'} = \arg\max_{\pi \in \Delta} \mathop{\mathbb{E}}_{\rho \sim \pi} r(\rho) - \tau D_{\mathrm{KL}}(\pi \| \pi_{\theta_t}) + \boldsymbol{\tau' \mathcal{H}(\pi)}. \end{aligned} \tag{4}$$

The idea of optimizing the reverse direction of KL divergence has proved to be effective in previous work, such as reward augmented maximum likelihood (Norouzi et al., 2016) and UREX (Nachum et al., 2017a). Its *mean seeking* behavior further encourages exploration by the algorithm. As we will see in Section 5, REPMD outperforms PMD significantly in experimental evaluations.

Theorem 1 shows that REPMD still enjoys similar desirable properties to PMD in the non-convex setting, but with respect to a surrogate reward $\mathrm{SR}(\pi_\theta)$, which we analyze further below.

Theorem 1. *Let π_{θ_t} denote the policy produced at step t of the update sequence. REPMD satisfies the following properties for an arbitrary parametrization of π.*

1. **(Monotonic Improvement Guarantee)** *If the Project Step $D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'} \| \pi_\theta)$ can be solved globally optimally, then*

$$\textit{SR}(\pi_{\theta_{t+1}}) - \textit{SR}(\pi_{\theta_t}) \geq 0,$$

where

$$\textit{SR}(\pi_\theta) \triangleq (\tau + \tau') \log \sum_\rho \exp\left\{ \frac{r(\rho) + \tau \log \pi_\theta(\rho)}{\tau + \tau'} \right\}. \tag{5}$$

2. **(Global Optimum Guarantee)** *If the Project Step can be solved globally optimally, then π_{θ_t} converges to the global optimum π^*, i.e.*

$$\lim_{t \to \infty} \textit{SR}(\pi_{\theta_t}) = \textit{SR}(\pi^*) \geq \textit{SR}(\pi), \quad \forall \pi \in \Pi.$$

3. **(Fixed points)** *Assume that the Project Step is optimized by gradient descent, then the fixed points of REPMD are the stationary points of the expected reward $\textit{SR}(\pi_\theta)$.*

A key question is the feasibility of solving the Project Step to global optimality. It is obvious that when $\pi(\theta) = \theta \in \Delta$ the Project Step is a convex optimization problem, hence can be solved optimally. Furthermore, as shown in Proposition 3, for a one-layer-softmax neural network π, the Project Step $D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'} \| \pi_\theta)$ can also still be solved to global optimality, affording a significant computational advantage over PMD.

Proposition 3. *Assume $\pi_\theta(s) = \mathrm{softmax}(\phi_s^\top \theta)$. Given a reference policy $\bar{\pi}$, the Projection Step $\min_{\theta \in \mathbb{R}^d} D_{\mathrm{KL}}(\bar{\pi} \| \pi_\theta)$ is a convex optimization problem in θ.*

3.1 Learning Algorithms

We now provide some of the learning algorithms for REPMD. Despite its non-convexity, the Lift Step has an analytic solution,

$$\bar{\pi}^*_{\tau,\tau'}(\rho) \triangleq \frac{\bar{\pi}(\rho) \exp\left\{ \frac{r(\rho) - \tau' \log \bar{\pi}(\rho)}{\tau + \tau'} \right\}}{\sum_{\rho'} \bar{\pi}(\rho') \exp\left\{ \frac{r(\rho') - \tau' \log \bar{\pi}(\rho')}{\tau + \tau'} \right\}}. \tag{6}$$

The Project Step in Eq. (4), $\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}^*_{\boldsymbol{\tau},\boldsymbol{\tau}'} \| \boldsymbol{\pi}_{\boldsymbol{\theta}})$, can be optimized via stochastic gradient descent, given that one can sample trajectories from $\bar{\pi}^*_{\tau,\tau'}$ to estimate its gradient. To see this, note that $\arg\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'} \| \pi_\theta) = \arg\min_{\pi_\theta \in \Pi} \mathbb{E}_{\rho \sim \bar{\pi}^*_{\tau,\tau'}} - \log \pi_\theta(\rho)$. The next lemma shows that sampling from $\bar{\pi}^*_{\tau,\tau'}$ can be done using self-normalized importance sampling (Owen, 2013) when it is possible to draw multiply samples, following the idea of UREX (Nachum et al., 2017a).

Algorithm 1 The REPMD algorithm

Input: temperature parameters τ and τ', number of samples for computing gradient K

1: Random initialized π_θ
2: **For** $t = 1, 2, \dots$ **do**
3: Set $\bar{\pi} = \pi_\theta$
4: **Repeat**
5: Sample a mini-batch of K trajectories from $\bar{\pi}$
6: Compute the gradient according to Eq. (7)
7: Update π_θ by gradient descent
8: **Until** converged or reach maximum of training steps

Lemma 1. *Let* $\omega_k = \frac{r(\rho_k) - \tau' \log \bar{\pi}(\rho_k)}{\tau + \tau'}$. *Given* K i.i.d. *samples* $\{\rho_1, \dots, \rho_K\}$ *from the* reference policy $\bar{\pi}$*, we have the following unbiased gradient estimator,*

$$\nabla_\theta D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'} \| \pi_\theta) \approx -\sum_{k=1}^{K} \frac{\exp\{\omega_k\}}{\sum_{j=1}^{K} \exp\{\omega_j\}} \nabla_\theta \log \pi_\theta(\rho_k), \tag{7}$$

Derivation for the analytic solution of the Lift Step and above Lemma as well as other implementation details can be found in Appendix B.

3.2 Behavior of $\mathrm{SR}(\pi)$

Theorem 1 only establishes desirable properties for REPMD with respect to $\mathrm{SR}(\pi)$, but not necessarily $r(\rho)$. In this section we present theoretical and empirical evidence that $\mathrm{SR}(\pi_\theta)$ is in fact a reasonable surrogate that can provide good guidance for learning, even when targeting desirable behavior with respect to $r(\rho)$. In fact, by properly adjusting the two temperature parameters τ and τ' in REPMD, the resulting surrogate objective $\mathrm{SR}(\pi_\theta)$ recovers existing performance measures.

Proposition 4. *SR*(π_θ) *satisfies the following properties:*

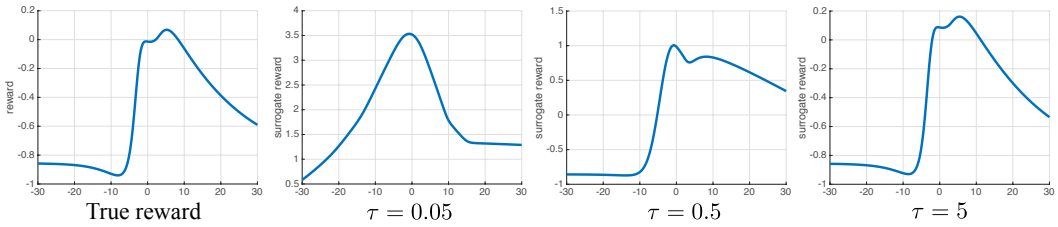
(i) *SR*$(\pi_\theta) \to \max_\rho r(\rho)$*, as* $\tau \to 0, \tau' \to 0$.

(ii) *SR*$(\pi_\theta) \to \mathbb{E}_{\rho \sim \pi_\theta} r(\rho)$*, as* $\tau \to \infty, \tau' \to 0$.

Remark 1. *Note that SR*(π_θ) *also resembles the "softmax value function" that has become popular in value based RL (Nachum et al., 2017b; Haarnoja et al., 2018; Ding & Soricut, 2017). The standard softmax value can be recovered by SR*(π_θ) *as a special case when* $\tau = 0$ *or* $\tau' = 0$.

According to Proposition 4, one should gradually decrease τ' to reduce the level of exploration as sufficient reward landscape information is collected during the learning process. Different choices can be made for τ however, depending on the policy constraint set Π. Given $\tau' \to 0$ and a sufficiently explored reward landscape, the resulting unconstrained policy must satisfy $\bar{\pi}^*_{\tau,\tau'} \to \pi^*$ as $\tau \to 0$, where π^* is the globally optimal deterministic policy. Therefore, in the Project Step, π_θ is obtained by directly projecting π^* into Π. When the policy constraint Π has nice properties that support good behavior of KL projection, such as convexity, π_θ will achieve good performance. However, in practice, Π is typically non-convex, and setting $\tau \to 0$ might not work very well, since directly projecting π^* into Π will not always lead to a π_θ with large expected reward.

On the other hand, as $\tau \to \infty$, the stationary point of $\mathrm{SR}(\pi_\theta)$ will approach the stationary point of $\sum_\rho \pi_\theta(\rho) r(\rho)$. Fig. 1 shows the a simulation investigating the behavior of $\mathrm{SR}(\pi_\theta)$ for different τ. Note that there is a poor local maximum for $\theta < 0$, where naive gradient method will converge to if initialization is about $\theta < -10$. When $\tau = 0.05$, the reward landscape of $\mathrm{SR}(\pi_\theta)$ guides θ to converge to the neighbourhood of 0, thus helping avoid the poor local maximum. Later in training, when τ increases to 5, $\mathrm{SR}(\pi_\theta)$ recovers the true expected reward landscape, ensuring that θ will converge to a good local (if not the global) maximum. While it is hard for a simulation study to be comprehensive, these results demonstrate how $\mathrm{SR}(\pi_\theta)$ can offer reasonable guidance for maximizing the true expected reward. Further investigation on the behavior of $\mathrm{SR}(\pi_\theta)$ is left for future work.



True reward $\tau = 0.05$ $\tau = 0.5$ $\tau = 5$

Figure 1: Simulation results for the true reward landscape and $\text{SR}(\pi_\theta)$ with different values of τ in a bandit setting with 10,000 actions. Each action is represented by a feature $\omega \in \mathbb{R}$. Let $\Omega = (\omega_1, \ldots, \omega_{10,000})$ be the feature vector. The policy is parameterized by a weight scalar $\theta \in \mathbb{R}$. The policy is defined by $\text{softmax}(\Omega^\top \theta)$. True Reward landscape shows the expected reward as a function of θ. The rest figures show $\text{SR}(\pi_\theta)$ as a function of θ with different value of τ.

4 An Actor-Critic Extension

Finally, we develop a natural extension of REPMD to an actor-critic formulation by incorporating a value function approximator. We refer to this final algorithm as Policy Mirror Actor-Critic (PMAC).

It is well known that data efficiency of policy-based methods can be generally improved by adding a value-based critic. For a reference policy $\bar{\pi}$ and an initial state s, recall that the objective in the Lift Step of REPMD is

$$\mathcal{O}_{\text{REPMD}}(\pi, s) = \mathop{\mathbb{E}}_{\rho\sim\pi} r(\rho) - \tau D_{\text{KL}}(\pi\|\bar{\pi}) + \tau'\mathcal{H}(\pi),$$

where $\rho = (s_1 = s, a_1, s_2, a_2, \ldots)$. To incorporate value function approximation, we need to derive the temporal consistency structure for this objective, which can be done as follows. First, write

$$\mathcal{O}_{\text{REPMD}}(\pi, s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[r(s,a) + \mathcal{O}_{\text{REPMD}}(\pi, s') + \tau \log \bar{\pi}(a|s) - (\tau + \tau')\log \pi(a|s)\right].$$

Let $\bar{\pi}^*_{\tau,\tau'}(\cdot|s) = \arg\max_\pi \mathcal{O}_{\text{REPMD}}(\pi, s)$ denote the optimal policy on state s. Further denote the soft optimal state-value function $\mathcal{O}_{\text{REPMD}}(\bar{\pi}^*_{\tau,\tau'}(\cdot|s), s)$ by $\bar{V}^*_{\tau,\tau'}(s)$, and let $\bar{Q}^*_{\tau,\tau'}(s,a) = r(s,a) + \gamma \bar{V}^*_{\tau,\tau'}(s')$ be the soft-Q function. It can then be verified that

$$\begin{aligned}
\bar{V}^*_{\tau,\tau'}(s) &= (\tau + \tau')\log \sum_a \exp\left\{\frac{\bar{Q}^*_{\tau,\tau'}(s,a) + \tau \log \bar{\pi}(a|s)}{\tau + \tau'}\right\}; \\
\bar{\pi}^*_{\tau,\tau'}(a|s) &= \exp\left\{\frac{\bar{Q}^*_{\tau,\tau'}(s,a) + \tau \log \bar{\pi}(a|s) - \bar{V}^*_{\tau,\tau'}(s)}{\tau + \tau'}\right\}.
\end{aligned} \tag{8}$$

We propose to train a soft state-value function V_ϕ parameterized by ϕ, a soft Q-function Q_ψ parameterized by ψ, and a policy π_θ parameterized by θ, based on Eq. (4). The update rules for these parameters can be derived as follows.

The soft state-value function approximates the soft optimal state-value $\bar{V}^*_{\tau,\tau'}$. Note that we can re-express $\bar{V}^*_{\tau,\tau'}$ by

$$\bar{V}^*_{\tau,\tau'}(s) = (\tau + \tau')\log \mathbb{E}_{a\sim\bar{\pi}}\left[\exp\left\{\frac{\bar{Q}^*_{\tau,\tau'}(s,a) - \tau' \log \bar{\pi}(a|s)}{\tau + \tau'}\right\}\right].$$

This suggests a Monte-Carlo estimate for $\bar{V}^*_{\tau,\tau'}(s)$: by sampling one single action a according to the reference policy $\bar{\pi}$, we have $\bar{V}^*_{\tau,\tau'}(s) \approx \bar{Q}^*_{\tau,\tau'}(s,a) - \tau' \log \bar{\pi}(a|s)$. Then, given a replay buffer $\mathcal{D}$, the soft state-value function can be trained to minimize the mean squared error,

$$L(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[\frac{1}{2}\left(V_\phi(s) - [Q_\psi(s,a) - \tau' \log \bar{\pi}(a|s)]\right)^2\right]. \tag{9}$$

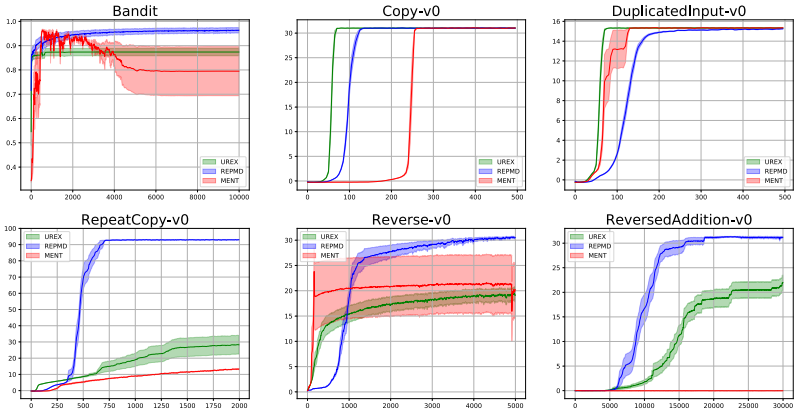


Figure 2: Results of MENT (red), UREX (green), and REPMD (blue) on synthetic bandit problem and algorithmic tasks. Plots show average reward with standard error during training. Synthetic bandit results averaged over 5 runs. Algorithmic task results averaged over 25 random training runs (5 runs $\times$ 5 random seeds for neural network initialization). X-axis is number of sampled trajectories.

One might note that, in principle, there is no need to include a separate state-value approximation, since it can be directly computed from a soft-Q function and reference policy, using Eq. (8). However, including a separate function approximator for the state-value can help stabilize the training (Haarnoja et al., 2018). The soft Q-function parameters ψ is then trained to minimize the soft Bellman error using the state-value network,

$$L(\psi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\psi(s,a) - [r(s,a) + \gamma V_\phi(s')]\right)^2\right] \tag{10}$$

The policy parameters are updated by performing the Project Step in Eq. (4) with stochastic gradient descent,

$$L(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\exp\left\{\frac{Q_\psi(s,\cdot) + \tau\log\bar{\pi}(\cdot|s) - V_\phi(s)}{\tau+\tau'}\right\}\middle\|\pi_\theta(\cdot|s)\right)\right] \tag{11}$$

where we approximate $\bar{\pi}^*_{\tau,\tau'}$ by the soft-Q and state-value function approximations.

Finally, we also use a target state-value network (Lillicrap et al., 2015) and the trick of maintaining two soft-Q functions (Haarnoja et al., 2018; Fujimoto et al., 2018). Implementation details for PMAC are given in Appendix C.

5 Experiments

We evaluate REPMD and PMAC on a number of benchmark tasks. We first introduce the experimented tasks, then present the experiment results. Implementation details are provided in Appendix E.2.

5.1 Settings

We test the performance of REPMD on a synthetic bandit problem and the algorithmic tasks from OpenAI gym library (Brockman et al., 2016). The synthetic multi-armed bandit problem has 10,000 distinct actions. The reward of each action i is initialized by $r_i = s_i^8$, where s_i is randomly sampled from a uniform distribution over $[0, 1)$. Each action i is represented by a randomly sampled feature vector $\omega_i \in \mathbb{R}^{20}$ from standard normal distribution. Note that these features are fixed during training. We further test our method on five algorithmic tasks from the OpenAI gym library, in rough order of difficulty: Copy, DuplicatedInput, RepeatCopy, Reverse, and ReversedAddition (Brockman et al., 2016). Lastly, we test PMAC using standard continuous-control benchmarks from OpenAI Gym utilizing the Mujoco environment (Brockman et al., 2016; Todorov et al., 2012), including Hopper, Walker2d, HalfCheetah, Ant and Humanoid. The details of algorithmic and mujoco tasks are provided in Appendix E.1.

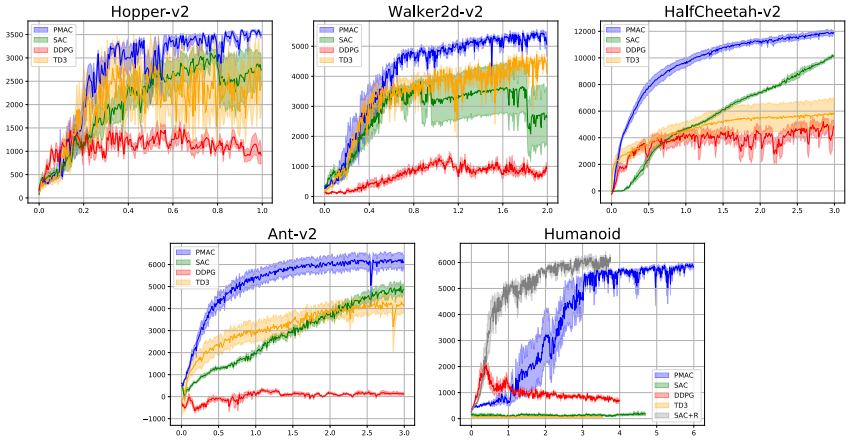


Figure 3: Learning curves of DDPG (red), TD3 (yellow), SAC (green) and PMAC (blue) on mujoco tasks. Plots show mean reward with standard error during training, averaged over five different instances with different random seeds. X-axis is millions of environment steps. We observe that PMAC is consistently able to math and, in many cases, outperform all baseline algorithms across all tasks, both in terms of final performance and sample efficiency.

Note that only cumulative rewards are available in both synthetic bandit problem and algorithmic tasks. Therefore value-based RL algorithms can not be applied in this setting. We compare REPMD against REINFORCE with entropy regularization (MENT) (Williams, 1992) and under-appreciated reward exploration (UREX) (Nachum et al., 2017a), which are the state-of-the-art policy-based algorithms in algorithmic tasks to our best knowledge. For continuous control tasks, we compare PMAC to deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015), an efficient off-policy deep RL methods, twin delayed deep deterministic policy gradient algorithm (TD3) (Fujimoto et al., 2018), a recent extension to DDPG by applying the double Q-learning trick to address over-estimation problem when function approximations are adopted, and Soft-Actor-Critic (SAC) (Haarnoja et al., 2018), a recent state-of-the-art off-policy algorithm on a number of benchmarks. All these algorithms are implemented in rlkit [2]. We do not include TRPO and PPO as our baselines in the experiments since their performance are dominated by SAC and TD3 as shown in (Haarnoja et al., 2018; Fujimoto et al., 2018).

5.2 Comparative Evaluation

The results on synthetic bandit problem and algorithmic tasks are reported in Fig. 2. It is clear that REPMD substantially outperforms the competitors on all of these benchmark tasks. REPMD is able to consistently achieve the highest score and learn substantially faster than UREX. We also find the performance of UREX is very unstable. On the difficult tasks, including RepeatCopy, Reverse and ReversedAddition, UREX can only successfully find appropriate solutions a few times out of 5 runs for each random seed, which brings the overall scores down. This observation creates the gap between our presented results with the ones reported in the paper[3]. Note that the performance of REPMD is sill significantly better than UREX even compared with the results reported in Nachum et al. (2017a).

Fig. 3 presents the total mean returns of evaluation rollouts during training of all algorithms. Results of each algorithm are averaged over five different instances with different random seeds. The solid curves corresponds to the mean and the shaded region to the standard errors over the five trials. For fair comparison with PMAC, we implement the double-Q but not the reparameterization trick (see equation (11)-(13) in (Haarnoja et al., 2018)) for SAC, which expains the discrepancy with the reported results in (Haarnoja et al., 2018). We observe that without the reparameterization trick SAC

[2]https://github.com/vitchyr/rlkit

[3]The results reported in Nachum et al. (2017a) averages over 5 runs of random restart, while our results are averaged over 25 random training runs (5 runs $\times$ 5 random seed for neural network initialization).

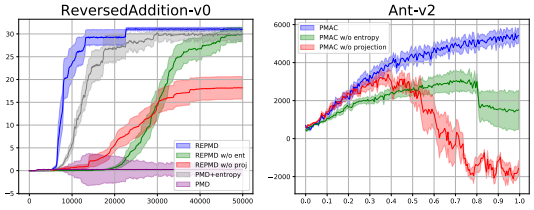


Figure 4: Ablation Study by comparing REPMD and PMAC with designed baselines on ReversedAddition and Ant.

cannot make any progress in the hardest problem Humanoid. To clarify this we also report the result of SAC with such trick on Humanoid using the author's GitHub implementation. The results show that PMAC matches or, in many cases, surpasses all other baseline algorithms in both final performance and sample efficiency across all tasks. In Humanoid, although PMAC is outperformed in the learning speed, its final performance is still comparable with SAC. Note that the reparameterization trick could also be applied in PMAC. We will include this in PMAC in the future work.

5.3 Ablation Study

The comparative evaluations provided in the previous sections suggest that our proposed algorithms based on the policy optimization method 4 outperform conventional RL methods on a number of challenging benchmarks. In this section, we further investigate how each novel component of Eq. (4) boosts the learning performance, by performing an ablation study on ReversedAddition and Ant.

Importance of entropy regularizer. The main difference between the function objective in Eq. (4) with the PMD objective Eq. (3) is to add another entropy regularizer. We demonstrate the importance of this choice by presenting the results of REPMD and PMAC without the extra entropy regularizer, i.e. $\tau' = 0$.

Importance of KL divergence projection. Another important difference between Eq. (4) with other RL methods is to use a project step to optimize policy, rather than direct SGD of the objective function. To show the importance of the project step, we test REPMD and PMAC without projection, which only performs one step of gradient update at each iteration of training.

Importance of direction of KL divergence. We choose PMD 3 as another baseline to prove the effectiveness of using the *mean seeking* direction of KL divergence in the project step. Similar as in REPMD, we add a separate tempreture parameter $\tau' > 0$ to the original objective function in (3) to encourage policy exploration, which gives $\arg\max_{\pi_\theta \in \Pi} \mathbb{E}_{\rho\sim\pi_\theta}\, r(\rho) - \tau \mathrm{KL}(\pi_\theta \| \bar{\pi}) + \tau' \mathcal{H}(\pi_\theta)$. We name this algorithm PMD+entropy, The corresponded algorithms in actor-critic setting, named PMD-AC and PMD-AC+entropy, are also implemented for comparison.

Results are presented in Fig. 4, which clearly indicate all of the three major components of Eq. (4) are helpful for yielding better performance.

6 Related Work

The lift-and-project approach is distinct from the previous literature on policy search, with the exception of a few recent works: Mirror Descent Guided Policy Search (MDGPS) (Montgomery & Levine, 2016), Guide Actor-Critic (GAC) (Tangkaratt et al., 2017), Maxmimum aposteriori (MPO) (Abdolmaleki et al., 2018), and Soft Actor-Critic (SAC) (Haarnoja et al., 2018). These approaches also adopt a mirror descent framework, but differ from the proposed method in key aspects. MDGPS (Montgomery & Levine, 2016) follows a different learning principle, using the Lift Step to learn multiple local policies (rather than a single policy) then aligning these with a global policy in the Project Step. MDGPS also does not include the entropy term in the Lift objective, which we have found to be essential for exploration. MPO (Abdolmaleki et al., 2018) also neglects to add the additional entropy term; Section 5.3 shows that entropy regularization with an appropriate annealing of τ' significantly improves learning efficiency. Both GAC and SAC use the mode seeking direction of KL divergence in the Project Step, reversed from the mean seeking direction we consider here

(Tangkaratt et al., 2017; Haarnoja et al., 2018). Additionally, SAC only uses entropy regularization in the Lift Step, neglecting the proximal relative entropy. The benefits of regularizing with relative entropy has been discussed in TRPO (Schulman et al., 2015) and MPO (Abdolmaleki et al., 2018), where it is noted that proximal regularization significantly improves learning stability. GAC seeks to match the mean of Gaussian policies under second order approximation in the Project Step, instead of directly minimizing the KL divergence with gradient descent. Although one might also attempt to interpret "one-step" methods in terms of lift-and-project, these approaches would obliviously still differ from REPMD, given that we use different directions of the KL divergence for the Lift and Project steps respectively.

Regarding the optimization objective, several existing methods have considered related approaches, either by considering (relative) entropy regularization during policy search, or directly using KL divergence as the target objective. As noted in Section 2, REPMD resembles policy gradient methods that maximize expected reward with an additional entropy regularizer (Williams & Peng, 1991; Fox et al., 2015; Nachum et al., 2017b). Using KL divergence, or Bregman divergences more generally, as regularizers has also been explored in Liu et al. (2015); Thomas et al. (2013); Mahadevan & Liu (2012). However, these approaches differ from the proposed method in important ways. In particular, they apply regularization to the parameters of the *linear* approximated value functions, whereas here KL regularization is applied directly to the policy space. The literature on relative entropy policy search also uses a similar KL divergence regularization scheme (Peters et al., 2010; Van Hoof et al., 2015), but on joint state-action distributions. Instead of KL divergence, Reward-Weighted Regression (RWR) uses a log of the correlation between π_τ^* and π_θ, which is then approximated similar to a cross entropy loss (Peters & Schaal, 2007; Wierstra et al., 2008).

TRPO/PPO also has a similar formulation to Eq. (2) as a constrained version with a mean seeking KL divergence Schulman et al. (2015; 2017b). Our proposed method includes additional modifications that, as shown in Section 5, significantly improve performance. UREX also uses the same mean seeking KL divergence for regularization, which encourages exploration but also complicates the optimization; as shown in Section 5, UREX is significantly less efficient than the method proposed here.

Trust PCL adopts the same objective defined in Eq. (4), including both entropy and relative entropy regularization (Nachum et al., 2017c). However, the policy update strategy is substantially different: while REPMD uses KL projection, Trust PCL minimizes a path inconsistency error (inherited from PCL) between the value and policy along observed trajectories (Nachum et al., 2017b). Although policy optimization by minimizing path inconsistency error can efficiently utilize off-policy data, this approach loses the desirable monotonic improvement guarantee.

In terms of existing theoretical analyses, similar monotonic improvement guarantee exists for TRPO, but only for an impractical formulation.[4] Here, by contrast, we use the lift-and-project reformulation to establish a monotonic improvement guarantee in a simple and direct way. MPO provides a guarantee on the regularized reward of a non-parametric policy, but this depends on the assumption that a non-convex optimization problem can be solved globally. SAC obtains a similar result with respect to the optimal achievable Q-values, again relying on an assumption that a non-convex optimization problem can be solved globally. We have shown that similar results hold in the case of PMD/REPMD, but here we have also established something stronger: When the projection step cannot be efficiently solved to global optimality, we have shown that PMD/REPMD still preserve stationary points in their respective, principled objectives. MPO and SAC have no such guarantee, as far as we are aware.

7 Conclusion and Future Work

We have proposed reversed entropy policy mirror descent (REPMD) as an effective new approach for policy based reinforcement learning that also guarantees monotonic improvement in a well motivated objective. We show that the resulting method achieves better exploration than both a directed exploration strategy (UREX) and undirected maximum entropy exploration (MENT). It will be interesting to further extend the follow-on PMAC actor-critic framework with further development of the value function learning approach.

[4] For the monotonic improvement guarantee to hold, TRPO must use $D_{\mathrm{KL}}^{\max}$ rather than the stanard KL divergence.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *ICLR*, 2018.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Nan Ding and Radu Soricut. Cold-start reinforcement learning with softmax policy gradient. In *Advances in Neural Information Processing Systems*, pp. 2817–2826, 2017.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Finite-sample analysis of proximal gradient td algorithms. In *UAI*, pp. 504–513. Citeseer, 2015.

Sridhar Mahadevan and Bo Liu. Sparse q-learning with mirror descent. *arXiv preprint arXiv:1210.4893*, 2012.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

William H Montgomery and Sergey Levine. Guided policy search via approximate mirror descent. In *Advances in Neural Information Processing Systems*, pp. 4008–4016, 2016.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA, 2012.

Ofir Nachum, Mohammad Norouzi, and Dale Schuurmans. Improving policy gradient by exploring under-appreciated rewards. In *ICLR*, 2017a.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2772–2782, 2017b.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-pcl: An off-policy trust region method for continuous control. In *ICLR*, 2017c.

Arkadii Nemirovskii, David Borisovich Yudin, and Edgar Ronald Dawson. Problem complexity and method efficiency in optimization. 1983.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pp. 1723–1731, 2016.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750. ACM, 2007.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, pp. 1607–1612. Atlanta, 2010.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

Voot Tangkaratt, Abbas Abdolmaleki, and Masashi Sugiyama. Guide actor-critic for continuous control. *arXiv preprint arXiv:1705.07606*, 2017.

Philip S Thomas, William C Dabney, Stephen Giguere, and Sridhar Mahadevan. Projected natural actor-critic. In *Advances in neural information processing systems*, pp. 2337–2345, 2013.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Herke Van Hoof, Jan Peters, and Gerhard Neumann. Learning of non-parametric control policies with high-dimensional state features. In *Artificial Intelligence and Statistics*, pp. 995–1003, 2015.

Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Episodic reinforcement learning by logistic reward-weighted regression. In *International Conference on Artificial Neural Networks*, pp. 407–416. Springer, 2008.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

A PROOFS FOR ??

This section is devoted to the omitted proofs in **??**.

A.1 PROOF OF PROPOSITION 1

Proof. Note that $-\tau D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}_\tau^*) = -\tau \sum_\rho \pi_\theta(\rho) \log \pi_\theta(\rho) + \tau \sum_\rho \pi_\theta(\rho)(\log \bar{\pi}(\rho) + r(\rho)/\tau) - Z_{\bar{\pi}} = \mathbb{E}_{\rho\sim\pi_\theta} r(\rho) - \tau D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}) - Z_{\bar{\pi}}$. Note the fact that $Z_{\bar{\pi}} \triangleq \tau \log \sum_\rho \bar{\pi}(\rho) \exp\{r(\rho)/\tau\}$ is indenpendent of π_θ given the reference policy $\bar{\pi}$. □

A.2 PROOF OF PROPOSITION 2

Proof. **(Monotonic Improvement Guarantee)** By the definition of $\pi_{\theta_{t+1}}$, note that $D_{\mathrm{KL}}(\pi_{\theta_{t+1}} \| \bar{\pi}_\tau^*) = \min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}_\tau^*) \le D_{\mathrm{KL}}(\pi_{\theta_t} \| \bar{\pi}_\tau^*)$. By expanding the KL divergence and rearranging terms, we have $\tau D_{\mathrm{KL}}(\pi_{\theta_{t+1}} \| \pi_{\theta_t}) - \sum_\rho \pi_{\theta_{t+1}}(\rho) r(\rho) \le -\sum_\rho \pi_{\theta_t}(\rho) r(\rho)$, which gives $\mathbb{E}_{\rho\sim\pi_{\theta_{t+1}}} r(\rho) - \mathbb{E}_{\rho\sim\pi_{\theta_t}} r(\rho) \ge \tau D_{\mathrm{KL}}(\pi_{\theta_{t+1}} \| \pi_{\theta_t}) \ge 0$.

(Global Optimum Guarantee) By the **Monotonic Improvement Guarantee** proved above, the sequence $\mathbb{E}_{\rho\sim\pi_{\theta_t}} r(\rho)$ is monotonically increased. Let π^* be the converged fixed policy, we show π^* is the optimal policy. According to the algorithm, at convergence it must be that $\forall \pi \in \Pi, \pi \ne \pi^*, D_{\mathrm{KL}}(\pi^* \| \bar{\pi}_\tau^*) \le D_{\mathrm{KL}}(\pi \| \bar{\pi}_\tau^*)$. Using the same argument above we have $\forall \pi \in \Pi, \pi \ne \pi^*, \mathbb{E}_{\rho\sim\pi^*} r(\rho) \ge \mathbb{E}_{\rho\sim\pi} r(\rho)$. Hence π^* is optimal in Π.

(Fixed Points) The stationary point of $\mathbb{E}_{\rho\sim\pi_\theta} r(\rho)$ is the π_θ which satisfies,

$$\begin{aligned} & \sum_\rho r(\rho) \cdot \frac{d\pi_\theta(\rho)}{d\theta} = 0 \\ \iff & \sum_\rho \left[\log \pi_\theta(\rho) - \log \pi_\theta(\rho) - \frac{r(\rho)}{\tau} \right] \cdot \frac{d\pi_\theta(\rho)}{d\theta} = 0 \qquad (\tau > 0) \\ \iff & \sum_\rho \left[\log \pi_\theta(\rho) - \log \{\pi_\theta(\rho) \exp\{r(\rho)/\tau\}\} \right] \cdot \frac{d\pi_\theta(\rho)}{d\theta} = 0. \end{aligned} \tag{12}$$

On the other hand, the fixed point of PMD indicates at some iteration t,

$$\begin{aligned} & \pi_{\theta_t} = \pi_{\theta_{t+1}}, \\ & \text{where } \pi_{\theta_t} \xrightarrow{\text{Lift Step}} \bar{\pi}_\tau^* \xrightarrow{\text{Project Step}} \pi_{\theta_{t+1}} \text{ in Eq. (3)}, \end{aligned} \tag{13}$$

which means π_{θ_t} is the solution of the Project Step,

$$\begin{aligned} & \left. \frac{dD_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}_\tau^*)}{d\theta} \right|_{\theta=\theta_t} = 0 \\ \iff & \left. \sum_\rho \left[\log \pi_\theta(\rho) - \log \bar{\pi}_\tau^*(\rho) + 1 \right] \cdot \frac{d\pi_\theta(\rho)}{d\theta} \right|_{\theta=\theta_t} = 0 \\ \iff & \left. \sum_\rho \left[\log \pi_\theta(\rho) - \log \left\{ \frac{\pi_{\theta_t}(\rho) \exp\{r(\rho)/\tau\}}{\sum_{\rho'} \pi_{\theta_t}(\rho') \exp\{r(\rho')/\tau\}} \right\} + 1 \right] \cdot \frac{d\pi_\theta(\rho)}{d\theta} \right|_{\theta=\theta_t} = 0 \\ & \qquad \text{(by Lift Step in Eq. (13))} \\ \iff & \left. \sum_\rho \left[\log \pi_\theta(\rho) - \log \{\pi_{\theta_t}(\rho) \exp\{r(\rho)/\tau\}\} + c \right] \cdot \frac{d\pi_\theta(\rho)}{d\theta} \right|_{\theta=\theta_t} = 0, \end{aligned} \tag{14}$$

where we denote $c = 1 + \log \sum_{\rho'} \pi_{\theta_t}(\rho') \exp\{r(\rho')/\tau\}$. Note that for c independent of ρ, we have,

$$\left. \sum_\rho c \cdot \frac{d\pi_\theta(\rho)}{d\theta} \right|_{\theta=\theta_t} = c \cdot \left. \frac{d \sum_\rho \pi_\theta(\rho)}{d\theta} \right|_{\theta=\theta_t} = c \cdot \left. \frac{d1}{d\theta} \right|_{\theta=\theta_t} = 0.$$

Therefore, Eq. (14) is equivalent with,

$$\iff \sum_{\rho} \left[\log \pi_\theta(\rho) - \log \left\{\pi_{\theta_t}(\rho) \exp\left\{r(\rho)/\tau\right\}\right\}\right] \cdot \left.\frac{d\pi_\theta(\rho)}{d\theta}\right|_{\theta=\theta_t} = 0. \tag{15}$$

Comparing Eq. (15) with Eq. (12), we have the fixed point condition of PMD is the same as the definition of the stationary point of $\mathbb{E}_{\rho\sim\pi_\theta}\, r(\rho)$. □

A.3 Proof of Theorem 1

Proof. **(Monotonic Improvement Guarantee)** Using $D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_{\theta_{t+1}}) = \min_{\pi_\theta\in\Pi} D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_\theta) \le D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_{\theta_t})$ and Jensen's inequality,

$$\begin{aligned}
&\mathrm{SR}(\pi_{\theta_{t+1}}) - \mathrm{SR}(\pi_{\theta_t}) = (\tau+\tau')\log \sum_\rho \frac{\exp\left\{\frac{r(\rho)+\tau\log\pi_{\theta_{t+1}}(\rho)}{\tau+\tau'}\right\}}{\sum_\rho \exp\left\{\frac{r(\rho)+\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\}} \\
=&(\tau+\tau')\log\sum_\rho \frac{\exp\left\{\frac{r(\rho)+\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\}}{\sum_\rho \exp\left\{\frac{r(\rho)+\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\}} \cdot \exp\left\{\frac{\tau\log\pi_{\theta_{t+1}}(\rho)-\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\} \\
=&(\tau+\tau')\log\sum_\rho \bar{\pi}^*_{\tau,\tau'}(\rho)\cdot\exp\left\{\frac{\tau\log\pi_{\theta_{t+1}}(\rho)-\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\} \\
\ge&(\tau+\tau')\sum_\rho \bar{\pi}^*_{\tau,\tau'}(\rho)\log\exp\left\{\frac{\tau\log\pi_{\theta_{t+1}}(\rho)-\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\} \\
=&\tau\sum_\rho \bar{\pi}^*_{\tau,\tau'}(\rho)\log\frac{\pi_{\theta_{t+1}}(\rho)}{\pi_{\theta_t}(\rho)} = \tau\left[D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_{\theta_t}) - D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_{\theta_{t+1}})\right] \ge 0.
\end{aligned}$$

(Global Optimum Guarantee) By the **Monotonic Improvement Guarantee** proved above, the sequence $SR(\pi_{\theta_t})$ is monotonically increased. Let π^* be the converged fixed policy, we show π^* is the optimal policy. According to the algorithm, at convergence it must be that $\forall\pi\in\Pi, \pi\neq\pi^*, D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi^*) \le D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi)$. Using the same argument above we have $SR(\pi^*)\ge SR(\pi)$. Hence π^* is optimal in Π.

(Fixed Points) The stationary point of $\mathrm{SR}(\pi_\theta)$ is the π_θ which satisfies,

$$\begin{aligned}
&\frac{d\mathrm{SR}(\pi_\theta)}{d\theta} = 0 \\
\iff\ &(\tau+\tau')\cdot\frac{\sum_\rho \exp\left\{\frac{r(\rho)+\tau\log\pi_\theta(\rho)}{\tau+\tau'}\right\}\cdot\frac{\tau}{\tau+\tau'}\cdot\frac{1}{\pi_\theta(\rho)}\cdot\frac{d\pi_\theta(\rho)}{d\theta}}{\sum_{\rho'}\exp\left\{\frac{r(\rho')+\tau\log\pi_\theta(\rho')}{\tau+\tau'}\right\}} = 0 \\
\iff\ &-\sum_\rho \frac{\pi_\theta(\rho)\exp\left\{\frac{r(\rho)-\tau'\log\pi_\theta(\rho)}{\tau+\tau'}\right\}}{\sum_{\rho'}\pi_\theta(\rho')\exp\left\{\frac{r(\rho')-\tau'\log\pi_\theta(\rho')}{\tau+\tau'}\right\}}\cdot\frac{1}{\pi_\theta(\rho)}\cdot\frac{d\pi_\theta(\rho)}{d\theta} = 0. \qquad (\tau>0)
\end{aligned} \tag{16}$$

On the other hand, the fixed point of REPMD indicates at some iteration t,

$$\begin{aligned}
&\pi_{\theta_t} = \pi_{\theta_{t+1}}, \\
&\text{where } \pi_{\theta_t} \xrightarrow{\text{Lift Step}} \bar{\pi}^*_{\tau,\tau'} \xrightarrow{\text{Project Step}} \pi_{\theta_{t+1}} \text{ in Eq. (4)},
\end{aligned} \tag{17}$$

which means π_{θ_t} is the solution of the Project Step,

$$\begin{aligned} & \frac{dD_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_\theta)}{d\theta}\bigg|_{\theta=\theta_t} = 0 \\ \iff & -\sum_\rho \bar{\pi}^*_{\tau,\tau'}(\rho)\cdot\frac{1}{\pi_\theta(\rho)}\cdot\frac{d\pi_\theta(\rho)}{d\theta}\bigg|_{\theta=\theta_t} = 0 \\ \iff & -\sum_\rho \frac{\pi_{\theta_t}(\rho)\exp\left\{\frac{r(\rho)-\tau'\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\}}{\sum_{\rho'}\pi_{\theta_t}(\rho')\exp\left\{\frac{r(\rho')-\tau'\log\pi_{\theta_t}(\rho')}{\tau+\tau'}\right\}}\cdot\frac{1}{\pi_\theta(\rho)}\cdot\frac{d\pi_\theta(\rho)}{d\theta}\bigg|_{\theta=\theta_t} = 0 \\ & \qquad\qquad \text{(by Lift Step in Eq. (17))} \end{aligned} \tag{18}$$

Comparing Eq. (18) with Eq. (16), we have the fixed point condition of REPMD is the same as the definition of the stationary point of $\mathrm{SR}(\pi_\theta)$. □

A.4 Proof of Proposition 3

Proof. Note that $\pi_\theta = \frac{\exp\{\Phi^\top\theta\}}{\mathbf{1}^\top\exp\{\Phi^\top\theta\}}$, where Φ is the feature matrix and θ is the policy parameter. Simply compute the Hessian matrix of the objective,

$$\frac{d^2 D_{\mathrm{KL}}(\bar{\pi}\|\pi_\theta)}{d\theta^2} = \Phi(\Delta(\pi_\theta) - \pi_\theta\pi_\theta^\top)\Phi^\top \succeq 0.$$

Thus $D_{\mathrm{KL}}(\bar{\pi}\|\pi_\theta)$ is convex in θ. □

A.5 Proof of Proposition 4

Proof. To prove (i), note that as $\tau \to 0$, $\mathrm{SR}(\pi_\theta) \to \tau'\log\sum_\rho \exp\left\{\frac{r(\rho)}{\tau'}\right\}$, the standard softmax value. Taking limit on τ' gives the hardmax value $\max_\rho r(\rho)$ as $\tau' \to 0$.

To prove (ii), we have

$$\lim_{\tau\to\infty}(\tau+\tau')\log\sum_\rho\exp\left\{\frac{r(\rho)+\tau\log\pi_\theta(\rho)}{\tau+\tau'}\right\} = \lim_{\tau\to\infty}\frac{\sum_\rho \pi_\theta(\rho)\exp\{\frac{r(\rho)-\tau'\log\pi_\theta(\rho)}{\tau+\tau'}\}\big(r(\rho)-\tau'\log\pi_\theta(\rho)\big)}{\sum_\rho\pi_\theta(\rho)\exp\{\frac{r(\rho)-\tau'\log\pi_\theta(\rho)}{\tau+\tau'}\}}$$

$$= \sum_\rho \pi_\theta(\rho)\left[r(\rho)-\tau'\log\pi_\theta(\rho)\right] = \mathop{\mathbb{E}}_{\rho\sim\pi_\theta} r(\rho) + \tau'\mathcal{H}(\pi_\theta)$$

As $\tau' \to 0$, $\mathrm{SR}(\pi_\theta) \to \mathbb{E}_{\rho\sim\pi_\theta} r(\rho)$. □

B Details of REPMD Learning

This section provides some of the details of learning algorithms for REPMD. We first show the derivation of the analytic solution of the Lift Step.

Lemma 2. *The lift step of Eq.* (4) *has the following closed form expression:*

$$\bar{\pi}^*_{\tau,\tau'}(\rho) \triangleq \frac{\bar{\pi}(\rho)\exp\left\{\frac{r(\rho)-\tau'\log\bar{\pi}(\rho)}{\tau+\tau'}\right\}}{\sum_{\rho'}\bar{\pi}(\rho')\exp\left\{\frac{r(\rho')-\tau'\log\bar{\pi}(\rho')}{\tau+\tau'}\right\}}. \tag{19}$$

Proof. Rewrite the objective function defined in Eq. (4),

$$\mathop{\mathbb{E}}_{\rho\sim\pi} r(\rho) - \tau D_{\mathrm{KL}}(\pi\|\bar{\pi}) + \tau'\mathcal{H}(\pi) = \mathop{\mathbb{E}}_{\rho\sim\pi}[r(\rho)+\tau\log\bar{\pi}(\rho)] + (\tau+\tau')\mathcal{H}(\pi), \tag{20}$$

which is an entropy regularized reshaped reward objective. The optimal policy of this objective can be obtained by directly applying Lemma 4 of Nachum et al. (2017b), i.e.

$$\bar{\pi}^*_{\tau,\tau'}(\rho) \propto \exp\left\{\frac{r(\rho)+\tau\log\bar{\pi}(\rho)}{\tau+\tau'}\right\} = \bar{\pi}(\rho)\exp\left\{\frac{r(\rho)-\tau'\log\bar{\pi}(\rho)}{\tau+\tau'}\right\}. \tag{21}$$

□

The next lemma provides the derivation of the gradient estimation of REPMD (Lemma 1).

Proof. Note that

$$D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_\theta) = \mathop{\mathbb{E}}_{\rho\sim\bar{\pi}^*_{\tau,\tau'}}\left[\log\bar{\pi}^*_{\tau,\tau'}(\rho) - \log\pi_\theta(\rho)\right] = \mathop{\mathbb{E}}_{\rho\sim\bar{\pi}}\left[\frac{\bar{\pi}^*_{\tau,\tau'}(\rho)}{\bar{\pi}(\rho)}\left(\log\bar{\pi}^*_{\tau,\tau'}(\rho) - \log\pi_\theta(\rho)\right)\right].$$

Therefore, taking gradient on both sides,

$$\begin{aligned}
\nabla_\theta D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_\theta) &\approx -\frac{1}{K}\sum_{k=1}^{K}\frac{\bar{\pi}^*_{\tau,\tau'}(\rho_k)}{\bar{\pi}(\rho_k)}\nabla_\theta\log\pi_\theta(\rho_k)\\
&= -\frac{1}{K}\sum_{k=1}^{K}\frac{\bar{\pi}(\rho_k)\exp\left\{\frac{r(\rho_k)-\tau'\log\bar{\pi}(\rho_k)}{\tau+\tau'}\right\}}{\bar{\pi}(\rho_k)\sum_{\rho'}\bar{\pi}(\rho')\exp\left\{\frac{r(\rho')-\tau'\log\bar{\pi}(\rho')}{\tau+\tau'}\right\}}\nabla_\theta\log\pi_\theta(\rho_k) \qquad \text{by 19}\\
&\approx -\frac{1}{K}\sum_{k=1}^{K}\frac{\exp\{\omega_k\}}{\frac{1}{K}\sum_{j=1}^{K}\exp\{\omega_j\}}\nabla_\theta\log\pi_\theta(\rho_k)\\
&= -\sum_{k=1}^{K}\frac{\exp\{\omega_k\}}{\sum_{j=1}^{K}\exp\{\omega_j\}}\nabla_\theta\log\pi_\theta(\rho_k). \qquad \square
\end{aligned}$$

Recall that in Algorithm 1 the project step is performed by SGD. In our implementation, the end condition of SGD is controlled by two parameters: $\epsilon > 0$ and F_STEP $\in \{0, 1\}$. First, SGD halts if the change of the KL divergence is below or equal to ϵ. Second, F_STEP decides the maximum number of SGD steps. If F_STEP $= 1$, the maximum number is $\sqrt{t}$ at iteration t; while if F_STEP $= 0$, there is no restriction on the maximum number of gradient steps, and stopping condition only depends on ϵ.

C Details of PMAC Learning

To increase the stability of the training, we include a target state value network $V_{\bar{\phi}}$, where $\bar{\phi}$ is an exponentially moving average of the value network weights ϕ. Different from Eq. (10), the soft Q-function parameters ψ is then trained to minimize the soft Bellman error using the target state value network,

$$L(\psi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\psi(s,a) - \left[r(s,a) + \gamma V_{\bar{\phi}}(s')\right]\right)^2\right] \tag{22}$$

Our approach also use two soft-Q functions in order to mitigate the overestimation problem caused by value function approximation (Haarnoja et al., 2018; Fujimoto et al., 2018). Specifically, we apply two soft-Q function approximations, $Q_{\psi_1}(s,a)$ and $Q_{\psi_2}(s,a)$, and train them independently. The minimum of the two Q-functions will be used whenever the soft-Q value is needed.

The next lemma shows that the gradient of Eq. (11) can be computed by importance sampling using the reference policy,

Lemma 3. *The gradient of Eq.* (11) *is,*

$$\nabla_\theta L(\theta) = \nabla_\theta\mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\bar{\pi}}\left[\exp\left\{\frac{Q_\psi(s,a) - \tau'\log\bar{\pi}(a|s) - V_\phi(s)}{\tau+\tau'}\right\}\log\pi_\theta(a|s)\right]\right]. \tag{23}$$

Algorithm 2 The PMAC algorithm

Input: temperature parameters τ and τ', lag on target value network α, number of training steps M

Initialize $\pi_\theta, V_\phi, V_{\bar{\phi}}, Q_{\psi_1}, Q_{\psi_2}$ and replay buffer $\mathcal{D}$
For $t = 1, 2, \ldots$ **do**
  For each environment step **do**
    $a \sim \pi_\theta(\cdot|s)$
    Observe s' and r from environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$
  Set $\bar{\pi} = \pi_\theta$
  For $i = 1, \ldots, M$ **do**
    Sample a mini-batch of data $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^B$ from $\mathcal{D}$
    Compute gradient $\nabla_\theta L(\theta), \nabla_\phi L(\phi), \nabla_{\psi_1} L(\psi_1), \nabla_{\psi_2} L(\psi_2)$according to Eqs. (9) to (11)
    Update parameters $\theta, \phi, \psi_1, \psi_2$ by gradient descent
    Update $\bar{\phi}$ by $\alpha\phi + (1-\alpha)\bar{\phi}$

Proof. Let $\pi(a|s) = \exp\left\{\frac{Q_\psi(s,a)+\tau\log\bar{\pi}(a|s)-V_\phi(s)}{\tau+\tau'}\right\}$, then we have,

$$
\begin{aligned}
\nabla_\theta L(\theta) =& \nabla_\theta \mathbb{E}_{s\sim\mathcal{D}}\left[\sum_a \pi(a|s)\log\pi(a|s) - \pi(a|s)\log\pi_\theta(a|s)\right] \\
=& \nabla_\theta \mathbb{E}_{s\sim\mathcal{D}}\left[-\sum_a \exp\left\{\frac{Q_\psi(s,a)+\tau\log\bar{\pi}(a|s)-V_\phi(s)}{\tau+\tau'}\right\}\log\pi_\theta(a|s)\right] \\
=& \nabla_\theta \mathbb{E}_{s\sim\mathcal{D}}\left[-\sum_a \bar{\pi}(a|s)\exp\left\{\frac{Q_\psi(s,a)-\tau'\log\bar{\pi}(a|s)-V_\phi(s)}{\tau+\tau'}\right\}\log\pi_\theta(a|s)\right] \\
=& \nabla_\theta \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\bar{\pi}}\left[-\exp\left\{\frac{Q_\psi(s,a)-\tau'\log\bar{\pi}(a|s)-V_\phi(s)}{\tau+\tau'}\right\}\log\pi_\theta(a|s)\right]\right]
\end{aligned}
$$

□

Pseudocode for PMAC is presented in Algorithm 2. The major difference between PMAC and REPMD in the lift step is that instead of sampling K actions as described in Algorithm 1, PMAC only samples one action to construct $\bar{\pi}^*_{\tau,\tau'}$, due to the fact both the soft-Q and state value function approximations are adopted. The value function approximations also make PMAC capable of using off-policy data from a reply buffer. Furthermore, in the project step of PMAC, we use a fixed number of iteration for SGD, which is given by an input parameter of the algorithm.

D Stochastic Transition Setting

In Section 1.1, we assume that the state transition function is deterministic for simplicity. For completeness, we consider the general stochastic transition setting here.

D.1 Notations and Settings

Recall in Section 1.1, the policy probability of trajectory $\rho = (s_1, a_1, \ldots, a_{T-1}, s_T)$ is denoted as $\pi(\rho) = \prod_{t=1}^{T-1}\pi(a_t|s_t)$. We define transition probability of ρ as $f(\rho) \triangleq \prod_{t=1}^{T-1} f(s_{s+1}|s_t, a_t)$. The total probability of ρ under policy π and transition f is then $p_{\pi,f}(\rho) \triangleq \pi(\rho)f(\rho) = \prod_{t=1}^{T-1}\pi(a_t|s_t)f(s_{s+1}|s_t, a_t)$. We use $\Delta_f \triangleq \{\pi|\sum_\rho p_{\pi,f}(\rho) = \sum_\rho \pi(\rho)f(\rho) = 1, \pi(\rho) \geq 0, f(\rho) > 0, \forall\rho\}$ to refer to the probabilistic simplex over all possible trajectories. It is obvious that $p_{\pi,f}(\rho) = \pi(\rho)$ and $\Delta_f = \Delta$ under deterministic transition setting, i.e., $f(\rho) = 1, \forall\rho$.

D.2 REPMD OPTIMIZATION PROBLEM

The proposed REPMD algorithm solves Eq. (4) in the deterministic transition setting. In the stochastic setting, the corresponding problem is,

$$\begin{aligned} &\textbf{(Project Step)} \quad \arg\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f} \| p_{\pi_\theta,f}), \\ &\textbf{(Lift Step)} \quad \text{where } \bar{\pi}^*_{\tau,\tau'} = \arg\max_{\pi \in \Delta_f} \mathop{\mathbb{E}}_{\rho \sim p_{\pi,f}} [r(\rho) - \tau' \log \pi(\rho)] - \tau D_{\mathrm{KL}}(p_{\pi,f} \| p_{\pi_{\theta_t},f}). \end{aligned} \tag{24}$$

which also recovers Eq. (4) as a special case when $f(\rho) = 1, \forall \rho$.

Like Eq. (4), $\bar{\pi}^*_{\tau,\tau'}$ in Eq. (24) also has a closed form expression,

Lemma 4. *The unconstrained optimal policy of Eq.* (24) *has the following closed form expression:*

$$\bar{\pi}^*_{\tau,\tau'}(\rho) \triangleq \frac{\bar{\pi}(\rho) \exp\left\{\frac{r(\rho) - \tau' \log \bar{\pi}(\rho)}{\tau + \tau'}\right\}}{\sum_{\rho'} \bar{\pi}(\rho') f(\rho') \exp\left\{\frac{r(\rho') - \tau' \log \bar{\pi}(\rho')}{\tau + \tau'}\right\}}.$$

Proof. Rewrite the maximization problem in Eq. (24) as (take π_{θ_t} as the reference policy $\bar{\pi}$),

$$\begin{aligned} &\underset{\pi}{\text{maximize}} \sum_\rho \pi(\rho) f(\rho) \left[r(\rho) - (\tau + \tau') \log \pi(\rho) + \tau \log \bar{\pi}(\rho)\right] \\ &\text{subject to} \sum_\rho \pi(\rho) f(\rho) = 1. \end{aligned}$$

The KKT condition of the above problem is,

$$\begin{aligned} f(\rho) \left[r(\rho) - (\tau + \tau') \log \pi(\rho) + \tau \log \bar{\pi}(\rho) + \lambda - (\tau + \tau')\right] = 0, \ \forall \rho \\ \sum_\rho \pi(\rho) f(\rho) = 1. \end{aligned}$$

Using $f(\rho) > 0, \forall \rho$ and solving the KKT condition, we obtain the expression of $\bar{\pi}^*_{\tau,\tau'}$. □

Lemma 4 recovers Lemma 2 as a special case when $f(\rho) = 1, \forall \rho$.

D.3 THEORETICAL ANALYSIS

In stochastic transition setting, we define the follow softmax approximated expected reward of π_θ

$$\mathrm{SR}_f(\pi_\theta) \triangleq (\tau + \tau') \log \sum_\rho f(\rho) \exp\left\{\frac{r(\rho) + \tau \log \pi_\theta(\rho)}{\tau + \tau'}\right\},$$

which recovers $\mathrm{SR}(\pi_\theta)$ when $f(\rho) = 1, \forall \rho$. The monotonic improvement property is for $\mathrm{SR}_f(\pi_\theta)$.

Theorem 2. *Assume that π_{θ_t} is the update sequence of the REPMD algorithm in Eq.* (24)*, then*

$$\mathit{SR}_f(\pi_{\theta_{t+1}}) - \mathit{SR}_f(\pi_{\theta_t}) \geq 0.$$

Proof. Using $D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f}\|p_{\pi_{\theta_{t+1}},f}) = \min_{\pi_\theta\in\Pi} D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f}\|p_{\pi_\theta,f}) \le D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f}\|p_{\pi_{\theta_t},f})$ and Jensen's inequality,

$$
\begin{aligned}
&\mathrm{SR}_f(\pi_{\theta_{t+1}}) - \mathrm{SR}_f(\pi_{\theta_t}) = (\tau+\tau')\log\sum_\rho \frac{f(\rho)\exp\left\{\frac{r(\rho)+\tau\log\pi_{\theta_{t+1}}(\rho)}{\tau+\tau'}\right\}}{\sum_\rho f(\rho)\exp\left\{\frac{r(\rho)+\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\}} \\
=&(\tau+\tau')\log\sum_\rho \frac{f(\rho)\exp\left\{\frac{r(\rho)+\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\}}{\sum_\rho f(\rho)\exp\left\{\frac{r(\rho)+\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\}}\cdot\exp\left\{\frac{\tau\log\pi_{\theta_{t+1}}(\rho)-\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\} \\
=&(\tau+\tau')\log\sum_\rho \bar{\pi}^*_{\tau,\tau'}(\rho)f(\rho)\cdot\exp\left\{\frac{\tau\log\pi_{\theta_{t+1}}(\rho)-\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\} \\
\ge&(\tau+\tau')\sum_\rho \bar{\pi}^*_{\tau,\tau'}(\rho)f(\rho)\cdot\log\exp\left\{\frac{\tau\log\pi_{\theta_{t+1}}(\rho)-\tau\log\pi_{\theta_t}(\rho)}{\tau+\tau'}\right\} \\
=&\tau\sum_\rho \bar{\pi}^*_{\tau,\tau'}(\rho)f(\rho)\cdot\log\frac{\pi_{\theta_{t+1}}(\rho)}{\pi_{\theta_t}(\rho)} \\
=&\tau\sum_\rho \bar{\pi}^*_{\tau,\tau'}(\rho)f(\rho)\cdot\log\frac{\pi_{\theta_{t+1}}(\rho)f(\rho)}{\pi_{\theta_t}(\rho)f(\rho)} \\
=&\tau\left[D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f}\|p_{\pi_{\theta_t},f}) - D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f}\|p_{\pi_{\theta_{t+1}},f})\right]\ge 0.
\end{aligned}
$$

□

$\mathrm{SR}_f(\pi_\theta)$ also recovers corresponding performance measures in the stochastic transition setting.

Proposition 5. *$\mathrm{SR}_f(\pi_\theta)$ satisfies the following properties:*

(i) $\mathrm{SR}_f(\pi_\theta) \to \max_\rho r(\rho)$, *as* $\tau\to 0, \tau'\to 0$.

(ii) $\mathrm{SR}_f(\pi_\theta) \to \mathop{\mathbb{E}}_{\rho\sim p_{\pi_\theta,f}} r(\rho)$, *as* $\tau\to\infty, \tau'\to 0$.

Proof. To prove (i), note that as $\tau\to 0$, $\mathrm{SR}_f(\pi_\theta)\to\tau'\log\sum_\rho f(\rho)\exp\left\{\frac{r(\rho)}{\tau'}\right\}$. Taking limit on τ' gives the hardmax value $\max_\rho r(\rho)$ as $\tau'\to 0$.

To prove (ii), we have

$$
\begin{aligned}
&\lim_{\tau\to\infty}(\tau+\tau')\log\sum_\rho f(\rho)\exp\left\{\frac{r(\rho)+\tau\log\pi_\theta(\rho)}{\tau+\tau'}\right\} \\
=&\lim_{\tau\to\infty}\frac{\sum_\rho \pi_\theta(\rho)f(\rho)\exp\{\frac{r(\rho)-\tau'\log\pi_\theta(\rho)}{\tau+\tau'}\}\left(r(\rho)-\tau'\log\pi_\theta(\rho)\right)}{\sum_\rho \pi_\theta(\rho)f(\rho)\exp\{\frac{r(\rho)-\tau'\log\pi_\theta(\rho)}{\tau+\tau'}\}} \\
=&\sum_\rho \pi_\theta(\rho)f(\rho)\left[r(\rho)-\tau'\log\pi_\theta(\rho)\right] = \mathop{\mathbb{E}}_{\rho\sim p_{\pi_\theta,f}} r(\rho) - \tau'\cdot\mathop{\mathbb{E}}_{\rho\sim p_{\pi_\theta,f}}\log\pi_\theta(\rho)
\end{aligned}
$$

As $\tau'\to 0$, $\mathrm{SR}_f(\pi_\theta)\to\mathbb{E}_{\rho\sim p_{\pi_\theta,f}} r(\rho)$. □

D.4 Learning

The REPMD learning process is intact under the stochastic transition setting. Similar with Appendix B, we can estimate the KL divergence in the projection step of Eq. (24) by drawing K *i.i.d.* samples $\{\rho_1,\dots,\rho_K\}$ from $p_{\bar{\pi},f}$, i.e., the mixture of $\bar{\pi}$ and f, which is exactly the process of sampling from $\bar{\pi}$ and interacting with the environment,

$$
\begin{aligned}
D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f}\|p_{\pi_\theta,f}) &= \mathop{\mathbb{E}}_{\rho\sim p_{\bar{\pi}^*_{\tau,\tau'},f}}\left[\log\bar{\pi}^*_{\tau,\tau'}(\rho)-\log\pi_\theta(\rho)\right] \\
&= \mathop{\mathbb{E}}_{\rho\sim p_{\bar{\pi},f}}\frac{\bar{\pi}^*_{\tau,\tau'}(\rho)}{\bar{\pi}(\rho)}\left[\log\bar{\pi}^*_{\tau,\tau'}(\rho)-\log\pi_\theta(\rho)\right].
\end{aligned}
\tag{25}
$$

We can then approximate the gradient of $D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f}\|p_{\pi_\theta,f})$ by averaging these K samples according to Eq. (25).

Theorem 3. *Let* $\omega_k = \frac{r(\rho_k)-\tau'\log\bar{\pi}(\rho_k)}{\tau+\tau'}$. *Given* K i.i.d. *samples* $\{\rho_1,\ldots,\rho_K\}$ *from the* reference policy $\bar{\pi}$*, we have the following unbiased gradient estimator,*

$$\nabla_\theta D_{\mathrm{KL}}(p_{\bar{\pi}^*_{\tau,\tau'},f}\|p_{\pi_\theta,f}) \approx -\sum_{k=1}^{K}\frac{\exp\{\omega_k\}}{\sum_{j=1}^{K}\exp\{\omega_j\}}\nabla_\theta\log\pi_\theta(\rho_k), \tag{26}$$

Proof. See the proof of Lemma 1. □

Similar argument could be applied for PMAC learning objectives.

E Experiments Details

We describe the algorithmic and mujoco tasks we experimented on as well as details of experimental setup in this section.

E.1 Algorithmic and Mujoco Tasks

In each algorithmic task, the agent operates on a tape of characters or digits. At each time step, the agent read one character or digit, and then decide to either move the read pointer one step in any direction of the tape, or write a character or digit to output. The total reward of each sampled trajectory is only observed at the end. The goal of each task is:

- **Copy:** Copy a sequence of characters to output.
- **DuplicatedInput:** Duplicate a sequence of characters.
- **RepeatCopy:** Copy a sequence of characters, reverse it, then forward the sequence again.
- **Reverse:** Reverse a sequence of characters.
- **ReversedAddition:** Observe two numbers in base 3 in little-endian order on a $2 \times n$ grid tape. The agent should add the two numbers together.

The Mujoco library contains various of continuous control tasks (Todorov et al., 2012). The specific action dimensions of each problem is summarized in Table 1.

Table 1: Action Dimensions of Mujoco Tasks

Task	Action Dimensions
Hopper	3
Walker2d	6
HalfCheetah	6
Ant	8
Humanoid	17

E.2 Implementation Details

For the synthetic bandit problem, we parameterize the policy by a weight vector $\theta \in \mathbb{R}^{20}$. Let $\Omega = (\omega_1,\ldots,\omega_{10,000})$ be the feature matrix. The policy is defined by softmax$(\Omega^\top\theta)$. The REPMD parameters used in Fig. 2 are summarized in Table 2.

For the algorithmic tasks, policy is parameterized by a recurrent neural network with LSTM cells of hidden dimension 256 (Hochreiter & Schmidhuber, 1997). In all algorithms, N distinct environments are used to generate samples. On each environment, K random trajectories are sampled using the agent's policy to estimate gradient according to (7), which gives the batch size $N \times K$ in total. We apply the same batch training setting as in UREX (Nachum et al., 2017a), where $N = 40$ and

Table 2: REPMD Hyperparameters in Synthetic Bandit

Parameter	Values
τ	0.1
τ'	0.0
learning rate	0.01
ϵ	$5 \cdot 10^{-4}$
F_STEP	0

Table 3: REPMD Hyperparameters in Algorithmic Tasks

	Copy	DuplicatedInput	RepeatCopy	Reverse	ReversedAddition
τ	0.5	0.5	2.0	0.2	0.5
τ'	0.01	0.01	0.01	0.02	0.01
learning rate	0.01	0.01	0.01	0.001	0.001
clip norm	20	20	20	20	20
ϵ	0.01	0.01	0.005	0.005	0.005

$K = 10$. F_STEP of REMPD is set to 1 in all tasks (See Appendix B). The REPMD parameters used in Fig. 2 are summarized in Table 3.

We use standard gaussian policy for all experimented algorithms in the mujoco tasks. Two layer fully-connected feed-forward neural networks with hidden dimension 300 and ReLU nonlinearity are applied to parameterize policy, soft state value, and soft-Q value. We batch size 256 for all algorithms on all tasks. The lag parameter α of PMAC for target value network update is 0.01, and the number of training steps is set as $M = 100$ in all tasks. The other domain-dependent PMAC parameters are summarized in Table 4.

Table 4: REPMD Hyperparameters in Mujoco Tasks

	Walker2d	Hopper	HalfCheetah	Ant	Humanoid
τ	1.5	0.5	0.5	1.0	2.0
τ'	0.2	0.05	0.2	0.1	0.05
ψ learning rate	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
ϕ learning rate	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
θ learning rate	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$