Policy Mirror Descent with Reversed KL Projection

Anonymous Author(s)
Affiliation
Address
email

Abstract

Policy optimization is a basic problem in reinforcement learning. This paper provides Reversed Entropy Policy Mirror Descent (REPMD), achieving two properties that enhance on-line exploration: preventing early convergence to sub-optimal policies, and monotonically increasing a performance measure. REPMD adopts maximum entropy exploration within the classic mirror descent framework, and updates policy by a reversed KL projection. This approach overcomes undesirable mode seeking behaviour, while still enjoying the policy improvement guarantee. Experiments on bandit and algorithmic tasks demonstrate that the proposed method achieves better exploration than both undirected maximum entropy exploration and directed exploration with standard entropy projection.

1 Introduction

Model-free deep reinforcement learning (RL) has recently been demonstrated to successfully solve a wide range of difficult sequential decision making problems [18, 8, 19]. Policy based DRL approaches represent the learner's policy with a neural network, and directly optimizes the policy by gradient update with respect to the parameters of the neural network. While the great expressive power of neural networks helps in modelling complicated policies, it also significantly introduces additional complications in understanding the behavior of the model.

In practice, policy based DRL is different than a traditional optimization problem, in the sense that the argument to be optimized, i.e. the policy, is also used to collect training data from the environment. This interaction may lead to lack of exploration, since the learner's policy may get stuck in a local optimum and fail to collect high reward trajectories, preventing any learning from useful signals. An on-line learning algorithm should have the ability to explore the policy space properly and efficiently, to avoid getting trapped in a locally optima, while discovering the globally optimal policy quickly.

In this paper, we propose a method with a better exploration strategy, such that (a) it retains the exploration efficiency of existing methods; (b) it monotonically increases its chance of exploring trajectories generated by the optimal policies, or evolving closer to the optimal policies. Our proposed method Reversed Entropy Policy Mirror Descent (REPMD), takes both the entropy and relative entropy regularizers. Unlike common policy gradient based methods, REPMD is in a two-stage manner. REPMD first updates the policy in the entire policy-simplex, ignoring the constraint induced by its parametrization, then a projection step is performed to update the policy in the parametrized policy space. Such a two-stage update guarantees REPMD to increase performance monotonically. The proposed REPMD method is then justified from both theoretical and empirical perspectives.

The rest of the paper is organized as follows. After introducing some basic knowledge of mirror descent, we propose the REPMD method in Section 2. We provide the analysis for the monotonically increasing performance property of REPMD in Section 3, and conduct experiments to validate our algorithm in Section 4. Some related work is discussed in Section 5, and the conclusion and directions for future work are presented in Section 6.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

1.1 Notations and Settings

We consider finite horizon reinforcement learning settings with finite state and action spaces. The behavior of an agent is modelled by a policy $\pi(a|s)$, which estimates 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 and it is 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 $r(\rho) = \sum_{t=1}^{T-1} r(s_t, a_t)$. We use $\Delta \triangleq \{\pi | \sum_\rho \pi(\rho) = 1, \pi(\rho) \geq 0, \forall \rho\}$ to refer to the probabilistic simplex over all possible trajectories. For simplicity we also assume that the state transition function is deterministic, while our results can be easily extended to the general setting with random transition functions.

2 Exploration in Policy Optimization

Policy optimization has been widely used across reinforcement learning (RL) settings. Given a set of parametrized policy functions $\pi_\theta \in \Pi$, policy optimization aims to search the optimal policy π_θ^* that achieves the highest expected reward,

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

However, directly optimizing the expected reward can lead to a lack of exploration, and such learning algorithms usually converge to a sub-optimal policy. Often, entropy regularization is proposed to mitigate the lack of exploration, leading to the following objective function,

$$\max_{\pi_\theta \in \Pi} \mathop{\mathbb{E}}_{\rho \sim \pi_\theta} r(\rho) + \tau \mathcal{H}(\pi_\theta) = \max_{\pi_\theta \in \Pi} \mathop{\mathbb{E}}_{\rho \sim \pi_\theta} [r(\rho) - \tau \log \pi_\theta(\rho)]. \tag{2}$$

where τ is a non-negative temperature parameter to control the degree of exploration. Using sampled trajectories to solve the above maximum entropy exploration (MENT) problem Eq. (2) is a classic policy gradient method, well known as REINFORCE [22, 23].

Although MENT assigns non-zero probability to every action, hence allowing it to be explored by π_θ in principle, such exploration is inherently *undirected*, i.e., "exploration is ensured only by randomness" [20]. Recently, Nachum et al. [10] proposed the under-appreciated reward exploration method (UREX) that provides extra guidance on the exploration probability, as follows:

$$\max_{\pi_\theta \in \Pi} \mathop{\mathbb{E}}_{\rho \sim \pi_\theta} r(\rho) - \tau D_{\mathrm{KL}}(\pi_\tau^* \| \pi_\theta), \tag{3}$$

where $\pi_\tau^*(\rho) \triangleq \frac{\exp\{r(\rho)/\tau\}}{\sum_{\rho'} \exp\{r(\rho')/\tau\}}$ is the optimal policy of Eq. (2) without the policy constraint Π. With $D_{\mathrm{KL}}(\pi_\tau^* \| \pi_\theta)$ as its regularizer, UREX tries to give larger probability values over high reward sampled trajectories to prevent π_θ from being too far away with π_τ^*. Despite its efficiency in exploration, UREX suffers from two fundamental problems. On the one hand, it is not clear what the fixed point of UREX actually represents, as shown in Proposition 1. On the other hand, there is no theoretical justification that UREX can monotonically increase the performance, which is favorable for any policy update methods.

Proposition 1. *The fixed point of UREX in Eq.* (3) *has the form*

$$\pi_\theta(\rho) = \frac{\tau \pi_\tau^*(\rho)}{\alpha - r(\rho)},$$

where α *is a constant such that* $\sum_\rho \pi_\theta(\rho) = 1$ *and* $\max_\rho r(\rho) + \tau \geq \alpha \geq \max\{\min_\rho r(\rho) + \tau, \max_\rho r(\rho)\}$.

Proof. See the appendix of [10]. □

3 Reversed Entropy Policy Mirror Descent

In this section we firstly discuss relative entropy regularization for policy based reinforcement learning and its relationship with mirror descent, paving the path for presenting our proposed Reversed Entropy Policy Mirror Descent (REPMD) algorithm.

3.1 Policy Mirror Descent

We first present a basic policy optimization algorithm based on the idea of Mirror Descent (MD). MD has been widely used in the literature of online learning and constrained optimization problems [13, 2]. In particular, given a *reference policy* $\bar{\pi}$, let

$$\mathrm{ENT}(\pi_\theta; \tau, \bar{\pi}) = \mathop{\mathbb{E}}_{\rho\sim\pi_\theta} r(\rho) - \tau D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi}).$$

The MD update for policy optimization problem is

$$\pi_{\theta_{t+1}} = \arg\max_{\pi_\theta \in \Pi} \mathrm{ENT}(\pi_\theta; \tau, \pi_{\theta_t}), \tag{4}$$

which combines the expected reward Eq. (1) with a relative entropy regularizer. Similar ideas have also been explored in [17, 21, 16, 18, 9, 12, 5, 1]. The following Proposition 2 suggests an efficient implementation of the mirror descent algorithm.

Proposition 2. *Given a* reference policy $\bar{\pi}$*, define* $\bar{\pi}^*_\tau(\rho) = \frac{\bar{\pi}(\rho)\exp\{r(\rho)/\tau\}}{\sum_{\rho'}\bar{\pi}(\rho')\exp\{r(\rho')/\tau\}}$*, then*

$$\arg\max_{\pi_\theta\in\Pi} \mathit{ENT}(\pi_\theta; \tau, \bar{\pi}) = \arg\min_{\pi_\theta\in\Pi} D_{\mathit{KL}}(\pi_\theta \| \bar{\pi}^*_\tau),$$

Proof. $-\tau D_{\mathrm{KL}}(\pi_\theta\|\bar{\pi}^*_\tau) = -\tau\sum_\rho \pi_\theta(\rho)\log\pi_\theta(\rho) + \tau\sum_\rho \pi_\theta(\rho)(\log\bar{\pi}(\rho) + r(\rho)/\tau) - Z_{\bar{\pi}} = \mathbb{E}_{\rho\sim\pi_\theta} r(\rho) - \tau D_{\mathrm{KL}}(\pi_\theta\|\bar{\pi}) - Z_{\bar{\pi}}$. 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 $\bar{\pi}$. □

Based on Proposition 2, policy mirror descent (PMD) updates the policy $\pi_{\theta_{t+1}}$ by

$$\begin{aligned} &\arg\min_{\pi_\theta\in\Pi} D_{\mathrm{KL}}(\pi_\theta\|\bar{\pi}^*_\tau), \\ \text{where } \bar{\pi}^*_\tau &= \arg\max_{\pi\in\Delta} \mathop{\mathbb{E}}_{\rho\sim\pi} r(\rho) - \tau D_{\mathrm{KL}}(\pi\|\pi_{\theta_t}). \end{aligned} \tag{5}$$

One can show that Mirror descent asymptotically converges to the optimal policy when Π is a convex set [13, 2]. However, in practice, the policy π_θ is often parameterized by a complex non-linear function, such as a neural network, which does not satisfy the convex constraint set assumption. The next proposition shows that despite of the non-convexity of Π, the MD update can still improve the expected reward monotonically.

Proposition 3. *Assume that* π_{θ_t} *is the update sequence of the policy mirror descent algorithm, then*

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

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

Although PMD enjoys this beneficial property, it is also well known that minimizing the KL divergence $D_{\mathrm{KL}}(\pi_\theta\|\bar{\pi}^*_\tau)$ is *mode seeking* [7], which can cause mode collapse during the learning process. Once the policy $\bar{\pi}^*_\tau$ drops some of the modes, learning could be trapped into sub-optimal policies. Moreover, PMD is also prone to converging to local optima: Note that the learning in Eq. (5) may converge to the policy π_{θ_t} that is a local maximizer of $\mathbb{E}_{\rho\sim\pi} r(\rho)$. Since $D_{\mathrm{KL}}(\pi\|\pi_{\theta_t}) = 0$ for such policy, the learning algorithm has no incentive for further exploration, and thus will become trapped at a local optimum. To overcome these drawbacks, we propose two modifications for the PMD algorithm, leading to our new algorithm.

3.2 Reversed Entropy Policy Mirror Descent

The new algorithm, called Reversed Entropy Policy Mirror Descent (REPMD), instead solves the following optimization problem to update the policy $\pi_{\theta_{t+1}}$:

$$\begin{aligned} &\arg\min_{\pi_\theta\in\Pi} D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_\theta), \\ \text{where } \bar{\pi}^*_{\tau,\tau'} &= \arg\max_{\pi\in\Delta} \mathop{\mathbb{E}}_{\rho\sim\pi} r(\rho) - \tau D_{\mathrm{KL}}(\pi\|\pi_{\theta_t}) + \tau'\mathcal{H}(\pi). \end{aligned} \tag{6}$$

First, an additional entropy regularizer, controlled by a separate parameter $\tau' \geq 0$, is added to the objective, which encourages additional exploration as in MENT. Second, we employ the reversed *mean seeking* direction of KL divergence $D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'} \| \pi_\theta)$ to update the policy. The idea of optimizing this direction of KL divergence has proven to be effective for structured prediction and reinforcement learning in previous work, such as reward augmented maximum likelihood [14] and UREX [10].

A similar monotonic improvement result to Proposition 3 can also be proved for REPMD but only on the $\mathrm{SR}(\pi_\theta)$, as shown in Theorem 1, where we let

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

denote the softmax approximated expected reward of π_θ.

Theorem 1. *Assume that π_{θ_t} is the update sequence of the reversed entropy policy mirror descent algorithm, then*

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

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

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

□

Theorem 1 implies that the fixed points of REPMD have a correspondence with the stationary point of $\mathrm{SR}(\pi_\theta)$. Although $\mathrm{SR}(\pi_\theta)$ is different than the expected reward $\mathbb{E}_{\rho \sim \pi_\theta} r(\rho)$, we argue that $\mathrm{SR}(\pi_\theta)$ is still a reasonable performance measure. In fact, by properly adjusting the two temperature parameters τ and τ', $\mathrm{SR}(\pi_\theta)$ recovers several existing performance measures, as shown in Proposition 4.

Proposition 4. *$\mathit{SR}(\pi_\theta)$ satisfies the following properties:*

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

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

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

$$\begin{aligned}
&\lim_{\tau \to \infty} (\tau + \tau') \log \sum_\rho \exp\{\frac{r(\rho) + \tau \log \pi_\theta(\rho)}{\tau + \tau'}\} = \lim_{\tau \to \infty} \frac{\sum_\rho \pi_\theta(\rho) \exp\{\frac{r(\rho) - \tau' \log \pi_\theta(\rho)}{\tau + \tau'}\} \big(r(\rho) - \tau' \log \pi_\theta(\rho)\big)}{\sum_\rho \pi_\theta(\rho) \exp\{\frac{r(\rho) - \tau' \log \pi_\theta(\rho)}{\tau + \tau'}\}} \\
&= \sum_\rho \pi_\theta(\rho) \left(r(\rho) - \tau' \log \pi_\theta(\rho)\right) = \mathop{\mathbb{E}}_{\rho \sim \pi_\theta} r(\rho) + \tau' \mathcal{H}(\pi_\theta)
\end{aligned}$$

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

Note that $\text{SR}(\pi_\theta)$ also resembles "softmax value function" that appeared in value based RL [11, 5, 4]. The standard soft value can be recovered by $\text{SR}(\pi_\theta)$ 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 has been collected during the learning process. Now we can make different choices for τ, depending on the policy constraint set Π.

Remark 1. *Using very small τ value, i.e., $\tau \to 0$, given $\tau' \to 0$ and the reward landscape has been sufficiently explored, the constructed unconstrained policy $\bar{\pi}^*_{\tau,\tau'} \to \pi^*$, where π^* is the global deterministic optimal policy. Solving the projection $\pi_\theta \leftarrow \arg\min_{\pi_\theta \in \Pi} D_{KL}(\bar{\pi}^*_{\tau,\tau'} \| \pi_\theta) \approx \arg\min_{\pi_\theta \in \Pi} D_{KL}(\pi^* \| \pi_\theta)$, we actually obtain π_θ policy by directly projecting π^* into Π. When the policy constraint Π has additional nice properties, such as convexity, that support good behavior of KL projection, this should be a good choice.*

Algorithm 1 The REPMD algorithm

Input: τ, τ', K
Output: Policy π_θ
1: Random initialized π_{θ_1};
2: **For** $t = 1, 2, \ldots, T$ **do**
3: Set $\bar{\pi} = \pi_{\theta_t}$;
4: **Repeat**
5: Sample a mini-batch of K trajectories from $\bar{\pi}$;
6: Compute the gradient according to Eq. (10);
7: Update $\pi_{\theta_{t+1}}$ by the gradient;
8: **Until** converged or reach max_iter;
9: Return π_{θ_T}.

However, in practice, Π is typically nonconvex, even when π_θ is parameterized by a linear function approximator, let alone more complex function approximations like a neural network. In such cases, the small τ value suggested in Remark 1 might not work very well, since directly projecting π^* into Π might not always lead to a π_θ with large expected reward.

Remark 2. *Given $\tau' \to 0$ and a sufficiently explored reward landscape, as $\tau \to \infty$, the stationary point set of SR(π_θ) will approach the stationary point set of $\sum_\rho \pi_\theta(\rho) r(\rho)$.*

Remark 2 indicates that during learning, π_θ may get stuck around some local maximum of $\text{SR}(\pi_\theta)$, and increasing the τ value will lead to a different contour of $\text{SR}(\pi_\theta)$, which might help π_θ escape to alternative local maxima. There exists an ideal sequence of τ values and $\tau \to \infty$ that make π_θ finally converge to $\pi^*_\theta \in \arg\max_{\pi_\theta} \sum_\rho \pi_\theta(\rho) r(\rho)$, i.e., the optimal policy in Π with highest expected reward, recovering the target of policy optimization Eq. (1). A principled way to find such an ideal sequence of τ is under investigation.

3.3 Learning

We discuss some implementation details of REPMD. The full algorithm is presented in Algorithm 1. First note that the *unconstrained optimal policy* $\bar{\pi}^*_{\tau,\tau'}$ in (6) has a closed form expression.

Lemma 1. *The unconstrained optimal policy of Eq.* (6) *has the following closed form expression:*

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

Proof. Rewrite the objective function defined in Eq. (6),

$$\mathop{\mathbb{E}}_{\rho \sim \pi} r(\rho) - \tau D_{\text{KL}}(\pi \| \bar{\pi}) + \tau' \mathcal{H}(\pi) = \mathop{\mathbb{E}}_{\rho \sim \pi} [r(\rho) + \tau \log \bar{\pi}(\rho)] + (\tau + \tau') \mathcal{H}(\pi), \tag{7}$$

which is an entropy regularized reshaped reward objective. The optimal policy of this objective can be obtained by directly applying Lemma 4 of [11], i.e.

$$\bar{\pi}^*_{\tau,\tau'}(\rho) \propto \exp\left\{\frac{r(\rho) + \tau \log \bar{\pi}(\rho)}{\tau + \tau'}\right\} = \bar{\pi}(\rho) \exp\left\{\frac{r(\rho) - \tau' \log \bar{\pi}(\rho)}{\tau + \tau'}\right\}. \tag{8}$$

□

The REPMD objective in Eq. (6) can be easily optimized via stochastic gradient descent, given one can sample trajectories from $\bar{\pi}^*_{\tau,\tau'}$. Following the idea of UREX [10], we approximate the KL

divergence using self-normalized importance sampling [15],

$$D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_\theta) = \mathop{\mathbb{E}}_{\rho\sim\bar{\pi}^*_{\tau,\tau'}}\left[\log\bar{\pi}^*_{\tau,\tau'}(\rho) - \log\pi_\theta(\rho)\right] = \mathop{\mathbb{E}}_{\rho\sim\bar{\pi}}\frac{\bar{\pi}^*_{\tau,\tau'}(\rho)}{\bar{\pi}(\rho)}\left[\log\bar{\pi}^*_{\tau,\tau'}(\rho) - \log\pi_\theta(\rho)\right]. \tag{9}$$

We can draw K *i.i.d.* samples $\{\rho_1,\ldots,\rho_K\}$ from the *reference policy* $\bar{\pi}$, and then approximate the gradient of $D_{\mathrm{KL}}(\bar{\pi}^*_{\tau,\tau'}\|\pi_\theta)$ by averaging these K samples according to Eq. (9).

Theorem 2. *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,*

$$\nabla_\theta \mathrm{KL}(\bar{\pi}^*_{\tau,\tau'}\|\pi_\theta) \approx -\sum_{k=1}^{K}\frac{\exp\{\omega_k\}}{\sum_{j=1}^{K}\exp\{\omega_j\}}\nabla_\theta\log\pi_\theta(\rho_k), \tag{10}$$

Proof. According to Eq. (9) we have,

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

□

4 Experiments

We present our experimental results in this section. REPMD is compared with standard policy based reinforcement learning algorithms on several different test domains.

4.1 Tasks

We test the performance of REPMD on a synthetic bandit problem and the algorithmic tasks from OpenAI gym library [3].

4.1.1 Synthetic Bandit

We develop a simple synthetic multi-armed bandit problem as an initial testbed. The 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 $[0, 1)$. We represent each action i with a feature vector $\phi_i \in \mathbb{R}^{20}$, randomly sampled from a standard normal distribution. Let $\Phi = [\phi_1,\ldots,\phi_{10,000}]$ denote the feature matrix. The policy is parameterized by a weight vector $\theta \in \mathbb{R}^{20}$ and defined by $\mathrm{softmax}(\Phi^\top\theta)$. Note that we only learn the θ parameter, the features Φ are fixed during training.

4.1.2 Algorithmic Tasks

We consider five algorithmic tasks, in rough order of difficulty: Copy, DuplicatedInput, RepeatCopy, Reverse, and ReversedAddition [3]. In each problem, 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 for 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.

For all of these algorithmic tasks, the policy is parameterized by a recurrent neural network with LSTM cells of hidden dimension 128 [6].

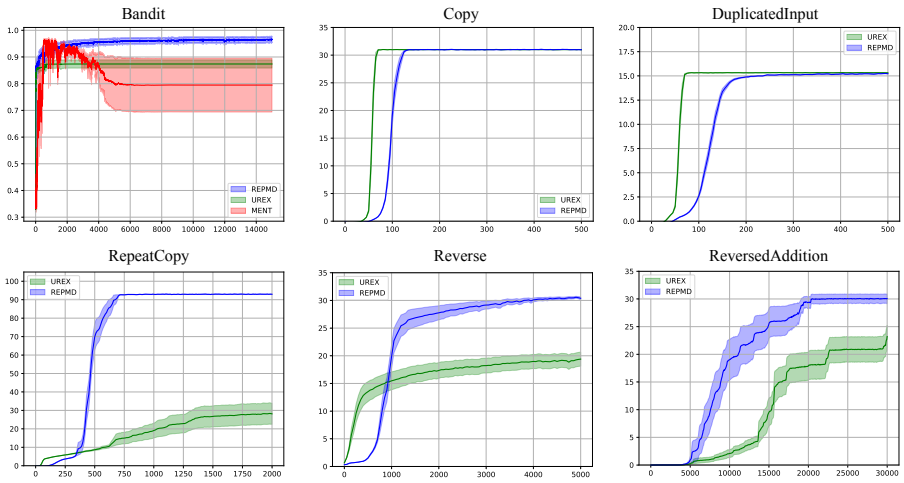

Figure 1: Results using the best hyper-parameters for each method: MENT (red), UREX (green), and REPMD (blue). Plots show average reward with standard error during training. Synthetic bandit results averaged over 5 runs. Algorithmic task results averaged over 25 random training runs (5 runs $\times$ 5 random seeds for neural network initialization). X-axis is number of sampled trajectories.

4.2 Implementation Details

As shown in Algorithm 1, the policy is updated by performing KL divergence projection using stochastic gradient descent (SGD). In our experiments, 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 of SGD only depends on ϵ.

For the synthetic bandit problem, we explore the following main hyper-parameters: learning rate $\eta \in \{0.1, 0.01, 0.001\}$; entropy regularizer of UREX and MENT $\tau \in \{1.0, 0.5, 0.1, 0.05\}$; relative entropy regularizer of REPMD $\tau \in \{1.0, 0.5, 0.1, 0.05\}$; $\epsilon \in \{0.01, 0.005, 0.001\}$ and F_STEP $\in \{0, 1\}$ for the stop condition of SGD in REPMD. The entropy regularizer τ' of REPMD is set to 0.

For the algorithmic tasks, 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 (10), which gives the batch size $N \times K$ in total. We apply the same batch training setting as in UREX [10], where $N = 40$ and $K = 10$. The following main hyper-parameters are explored: learning rate $\eta \in \{0.1, 0.01, 0.001\}$; relative entropy regularizer of REPMD $\tau \in \{1.0, 0.5, 0.1, 0.05\}$; entropy regularizer of REPMD $\tau' \in \{0, 0.01, 0.005, 0.001\}$; gradient clipped norm for training LSTM $c \in \{1, 10, 40, 100\}$; $\epsilon \in \{0.01, 0.005, 0.001\}$ and F_STEP $\in \{0, 1\}$ for the stopping condition of SGD in REPMD. Parameters of UREX are set according to the ones reported in [10]. Implementations of all algorithm are based on the open source code by the author of UREX [1].

4.3 Comparative Evaluation

For the synthetic bandit problem, we compare REPMD against REINFORCE with entropy regularization (MENT) [22] and under-appreciated reward exploration (UREX) [10]. For the algorithmic tasks, we compare REPMD only against UREX, since UREX has been shown to outperform MENT in these cases [10]. The results are reported in Figure (1). 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

[1] `https://github.com/tensorflow/models/tree/master/research/pcl_rl`

with the ones reported in the paper[2]. Note that the performance of REPMD is sill significantly better than UREX even compared with the results reported in [10].

4.4 Ablation Study

Importance of entropy regularizer. The main difference between our objective Eq. (5) with the original MD is to add another entropy regularizer. We demonstrate the importance of this choice by presenting the results of REPMD with $\tau' = 0$.

Importance of KL divergence projection. The main difference between REPMD and the UREX and MENT training methods is to use a projection step to optimize policy rather than performing a single gradient step. To show the importance of the projection step, we reimplement REPMD without projection, which only performs one step of gradient update at each iteration of training.

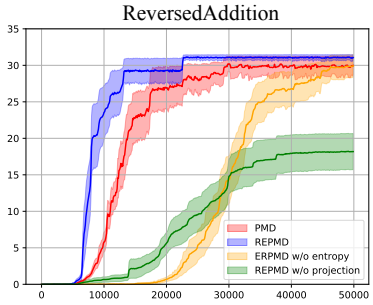

Figure 2: Ablation Study.

Importance of direction of KL divergence. We implemented Policy Mirror Descent (PMD) as another baseline to prove the effectiveness of using the *mean seeking* direction of KL divergence for policy optimization. Like in REPMD, we add a separate tempreture parameter $\tau' \geq 0$ to the original objective function (4) of PMD to encourage further exploration of the policy, which gives $\arg\max_{\pi_\theta \in \Pi} \mathbb{E}_{\rho \sim \pi_\theta} r(\rho) - \tau \mathrm{KL}(\pi_\theta \| \bar{\pi}) + \tau' \mathcal{H}(\pi_\theta)$.

Results on ReversedAddition are reported in Figure (2). It clearly shows that optimizing policy by performing the *mean seeking* KL divergence projection is very important as suggested in REPMD.

5 Related Work

As mentioned in Section 3.1, several existing methods are similar with the policy mirror descent (PMD) framework, by either considering the relative entropy regularizer in policy search/optimization, or using KL divergence as the loss function to optimize the policy. These related works include reward-weighted regression [17, 21], relative entropy policy search [16], trust region policy optimization [18], approximate mirror descent policy search [9], maximum a posterior policy optimization [1], and soft actor-critic [5]. The proposed reversed entropy policy mirror descent (REPMD) method employs a modified version of PMD framework. First, the objective function of PMD is combined with entropy regularizer to enhance further exploration. Second, the projection step applies the *mean seeking* direction of KL divergence instead of the *mean seeking* one used PMD. As shown in both theoretical and practical justification, these two novelties play a very important role in REPMD.

The Trust PCL method adopts the same objective defined in Eq. (6), which includes both entropy and relative entropy regularizer [12]. However, the policy update strategy is different: while REPMD uses KL projection, Trust PCL inherits the same idea from PCL that minimizes path inconsistency error between value and policy for any trajectories [11]. Although policy optimization by minimizing path inconsistency error can efficiently utilize off-policy data, it is not justified that this method can guarantee policy improvement during learning. On the other hand, the proposed REPMD method can monotonically increase the softmax approximated expected reward, as shown in Section 3.2.

6 Conclusion and Future Work

In this paper, we have proposed the reversed entropy policy mirror descent (REPMD) method for policy based reinforcement learning, which guarantees monotonic improvement in a well motivated objective. We show that the resulting method achieves better exploration than both a directed exploration method (UREX) and undirected maximum entropy exploration (MENT). It would be interesting to further extend the REPMD method within the actor-critic framework, by developing proper value function learning approach.

[2]The results reported in [10] 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).

References

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

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

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[4] Nan Ding and Radu Soricut. Cold-start reinforcement learning with softmax policy gradient. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2817–2826. 2017.

[5] 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.

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

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

[8] 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.

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

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

[11] 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*, pages 2772–2782, 2017.

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

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

[14] 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*, pages 1723–1731, 2016.

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

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

[17] 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*, pages 745–750. ACM, 2007.

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

[19] 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.

[20] Sebastian B Thrun. Efficient exploration in reinforcement learning. *Technical report*, 1992.

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

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

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