Policy Mirror Descent with Reversed KL Projection

Ruitong Huang

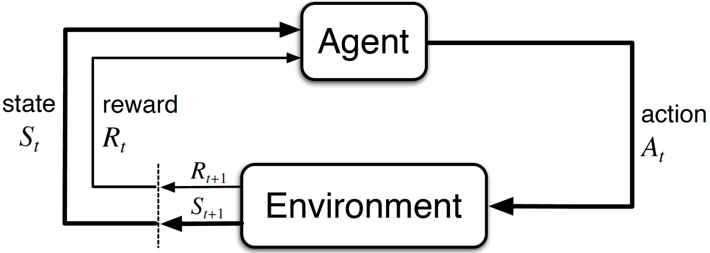
Joint work with: Jincheng Mei, Chenjun Xiao, and Dale Schuurmans

Borealis AI

Overview

1. Background
   - Reinforcement learning
   - Policy gradient methods
     - REINFORCE
     - UREX
2. Our Algorithms
   - PMD
   - REPMD
3. Experimental Results
   - Algorithmic tasks in OpenAI gym
   - Ablation Study

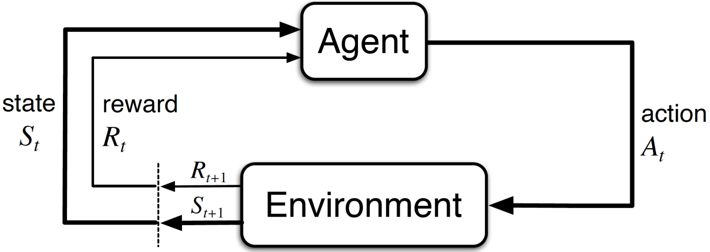
Reinforcement Learning



- MDP: finite actions $A_t \in \mathcal{A}$ and finite states $S_t \in \mathcal{S}$;
- Agent: policy $\pi_\theta(a_t|s_t)$, parametrized by θ; **Non-convex!**
- Environment: reward $\mathbb{P}\left(R_{t+1}|A_t, S_t\right)$, and transition $\mathbb{P}\left(S_{t+1}|A_t, S_t\right)$;

[Image source: Sutton and Barto, Reinforcement Learning book]

Reinforcement Learning



- WLOG, assume the state transition function is deterministic.
- State-action sequence: $\rho = (s_1, a_1, \ldots, a_{T-1}, s_T)$;

Goal: Maximizing the expected reward

$$\max_{\theta} \mathbb{E}\left[R(\rho) \,|\, \pi_\theta\right] .$$

[Image source: Sutton and Barto, Reinforcement Learning book]

Policy Gradient

$$\textbf{Goal:} \;\; \max_\theta \mathbb{E}\left[R(\rho) \mid \pi_\theta\right] = \max_\theta \sum_\rho \mathbb{P}\left(\rho|\pi_\theta\right) R(\rho);$$

Compute the gradient: $\nabla f = f \nabla \log(f)$,

$$\begin{aligned}\nabla_\theta \mathbb{E}\left[R(\rho) \mid \pi_\theta\right] = \sum_\rho \nabla_\theta \mathbb{P}\left(\rho|\pi_\theta\right) R(\rho) &= \sum_\rho \mathbb{P}\left(\rho|\pi_\theta\right) \nabla_\theta \log \mathbb{P}\left(\rho|\pi_\theta\right) R(\rho) \\ &= \mathbb{E}_\rho\left[\nabla_\theta \log \mathbb{P}\left(\rho|\pi_\theta\right) R(\rho)\right]\end{aligned}$$

To compute $\nabla_\theta \log \mathbb{P}\left(\rho|\pi_\theta\right)$**:**

$$\nabla_\theta \log \mathbb{P}\left(\rho|\pi_\theta\right) = \nabla_\theta \sum_{t=1}^{T-1} \log \pi_\theta(a_t|s_t); \;\; \text{(Dynamics Independent!)}$$

REINFORCE

REINFORCE

Compute the gradient w.r.t. θ:

- Sample $\rho_1, \ldots, \rho_M$ from the current π_θ;
- Estimate the gradient $\mathbb{E}_\rho \left[\nabla_\theta \log \mathbb{P}\left(\rho|\pi_\theta\right) R(\rho)\right]$ by

$$\frac{1}{M} \sum_{\rho_i} R(\rho_i) \sum_{t=1}^{T-1} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t});$$

- Update θ.

REINFORCE with entropy regularization

Entropy regularization to encourage exploration:

$$\max_\theta \mathbb{E}\left[R(\rho) \,|\, \pi_\theta\right] + \tau \mathcal{H}(\pi_\theta) = \max_\theta \mathbb{E}\left[R(\rho) \,|\, \pi_\theta\right] - \tau \pi_\theta \log \pi_\theta \,.$$

UREX

UREX

Reward-guided exploration:

$$\max_{\theta} \mathbb{E}\left[R(\rho)\,|\,\pi_\theta\right] + \tau \pi_\tau \log \pi_\theta \,,$$

where $\pi_\tau(\rho) = \frac{1}{Z}\exp(\frac{1}{\tau}R(\rho))$.

- Encourages exploration, but emphasizes more on larger rewarded ρ;
- May converge to poor local optima: estimate of π_τ is bad at the beginning of the training, thus it provides poor guidance;
- Difficult to interpret its fixed point (for a fixed τ), even in the simplex set;

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

- No theoretical justification in its performance.

[*] α is the normalizer.

This work

- Typical analysis only focus on policy space.
- We focus on analysis in the parameter space.
  - One-layer neural network: $\pi_\theta(\cdot|s) = \mathrm{softmax}(X_s^\top \theta)$;
  - Fixed points analysis;
  - Theoretical guarantee in performances.

Our Algorithms

Policy Mirror Descent (PMD)

PMD

Mirror descent type regularization (trust region):

$$\max_{\theta} \mathbb{E}\left[R(\rho) \,|\, \pi_\theta\right] - \tau D_{\mathrm{KL}}(\pi_\theta || \pi_{\theta_{\mathrm{old}}}) \,.$$

- Essentially the same as TRPO;
- Our analysis is in the parameter space.

Projection view of PMD

$$\arg\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}\left(\pi_\theta || \pi_\tau^*\right) \qquad \text{(Projection)}$$

$$\text{where } \pi_\tau^* = \arg\max_{\pi \in \Delta} \mathbb{E}\left[R(\rho) \,|\, \pi_\theta\right] - \tau D_{\mathrm{KL}}(\pi_\theta || \pi_{\theta_{\mathrm{old}}}) \qquad \text{(Improvement)}$$

Properties of PMD

Properties of PMD

If the projection step can be solved optimally,

- Property 1: Monotonically improves the expected reward;

$$\mathbb{E}\left[R(\rho) \,\middle|\, \pi_{\theta_{t+1}}\right] - \mathbb{E}\left[R(\rho) \,|\, \pi_{\theta_t}\right] \geq 0,$$

- Property 2: Its fixed points include the optimal policy in the parameter space.

- TRPO also has the monotonical improvement property, but only in the policy space;
- Poor local optima still exist: can be observed in simulation and real experiments;
- In practice, the projection step cannot be solved optimally, because it is non-convex in θ, even for 1-layer neural network.

Reversed Entropy Policy Mirror Descent (REPMD)

REPMD

$$\arg\min_{\pi_\theta \in \Pi} D_{\mathrm{KL}}\left(\pi^*_{\tau,\tau'} || \pi_\theta\right)$$

$$\text{where } \pi^*_{\tau,\tau'} = \arg\max_{\pi \in \Delta} \mathbb{E}\left[R(\rho) \mid \pi_\theta\right] - \tau D_{\mathrm{KL}}(\pi_\theta || \pi_{\theta_{\mathrm{old}}}) + \tau' \mathcal{H}(\pi_\theta),$$

with $\tau \to \infty$ and $\tau' \to 0$.

- **Entropy regularization:** encourages exploration to mitigate the local optima problem;
- **Reversed entropy:** convexifies the projection step in θ for the 1-layer neural network case, thus the projection step is solvable even in the parameter space.

Properties of REPMD

Properties of REPMD

When using 1-layer neural network: $\pi_\theta(\cdot|s) = \text{softmax}(X_s^\top \theta)$, REPMD satisfies

- Property 1: Fixed points include the optimal policy.
- Property 2: Monotonically improves the surrogate reward;

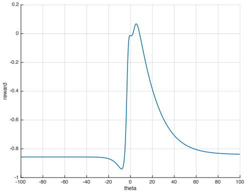
$$SR(\pi_{\theta_{t+1}}) - SR(\pi_{\theta_t}) \geq 0,$$

where $SR(\pi) = (\tau + \tau') \log \sum_\rho \exp\left(\frac{R(\rho)+\tau \log \pi(\rho)}{\tau+\tau'}\right)$.

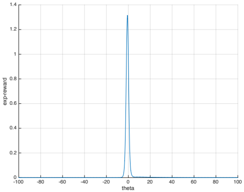
- Local optima still exist.

Behavior of $SR(\pi)$

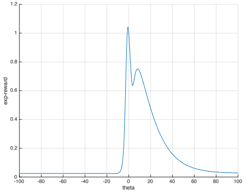
- $SR(\pi) = \tau \log \sum_\rho \pi(\rho) \exp\left(\frac{R(\rho)}{\tau}\right)$, as $\tau' \to 0$;
- $SR(\pi) \to \mathbb{E}\left[R(\rho)|\pi\right]$, as $\tau \to \infty$ and $\tau' \to 0$.
- Landscape of $SR(\pi)$: X-axis is θ, Y-axis is reward.



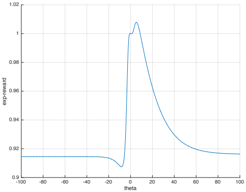
true reward landscape



$\tau = 0.2$



$\tau = 0.5$

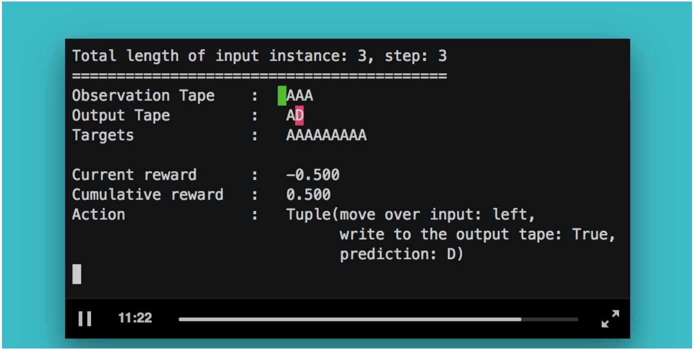


$\tau = 10$

Experiments

- Tasks:
  - Bandit simulation
  - OpenAI gym algorithmic tasks
    - **Copy:** *ababaa* $\rightarrow$ *ababaa*.
    - **DuplicatedInput:** *aba* $\rightarrow$ *aabbaa*.
    - **RepeatCopy:** *abc* $\rightarrow$ *abccbaabc*.
    - **Reverse:** *aabc* $\rightarrow$ *cbaa*.
    - **ReversedAddition:** Observe two numbers in base 3 in little-endian order on a $2 \times n$ grid tape: $120; 221 \rightarrow 022$
- Reward:
  - Correct(+1); incorrect(-0.5 & terminate); idle(0); overtime(-1);
  - Only accumulative reward is available
- Observation: letter on the current position of the read head
- Actions: move the read head; write (some symbol or nothing);

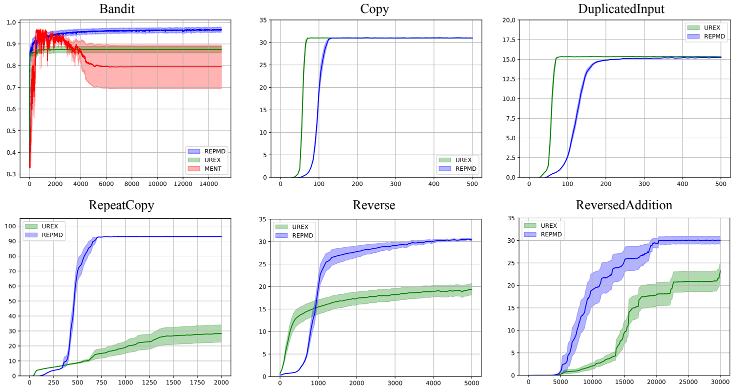
Experiments



- Reward:
  - Correct(+1); incorrect(-0.5 & terminate); idle(0); overtime(-1);
  - Only accumulative reward is available
- Observation: letter on the current position of the read head
- Actions: move the read head; write (some symbol or nothing);

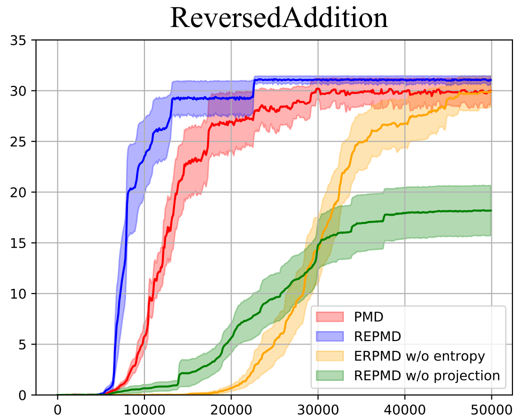
[Image source: OpenAI Gym]

Experiments



- X-axis is number of sampled trajectories; Y-axis is reward;
- REINFORCE (red), UREX (green), and REPMD (blue);
- Bandit results averaged over 5 repetitions. Algorithmic task results averaged over 25 runs (5 repetitions $\times$ 5 random seeds).

Ablation Study



- REPMD (blue), REPMD w/o entropy (yellow), PMD with entropy (red), REPMD w/o projection (green).

Thank you !

Questions?