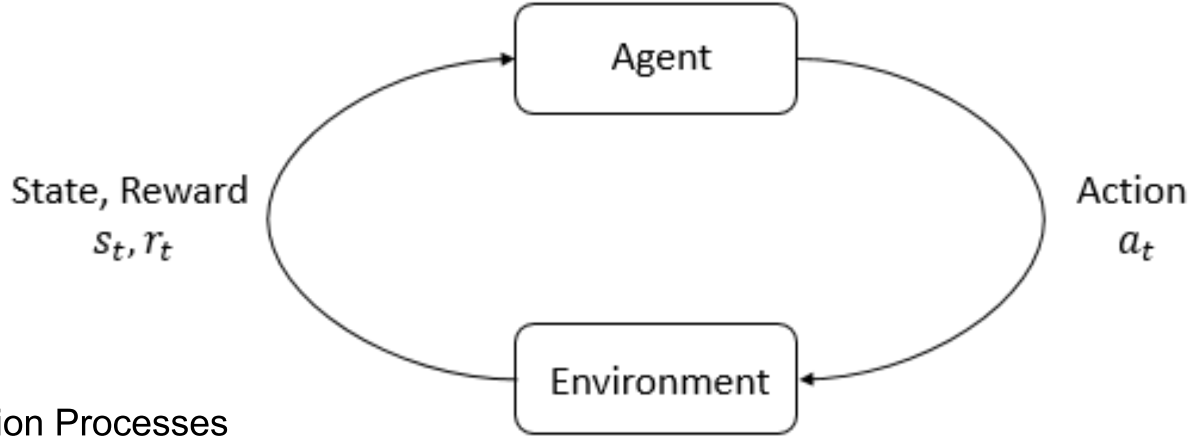
Trust Region Policy Optimization

Presenter: Puyuan (Jason) Peng

Oct 12

Reinforcement Learning



Markov Decision Processes

$$p_\theta(\tau) = p_\theta(s_1, a_1, \cdots, s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$

The goal is to maximize $J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \sum_t \gamma^t r(s_t, a_t)$

 Source: https://spinningup.openai.com/en/latest/spinningup/rl_intro1.html

Modern RL Algorithms

The Goal: maximize

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \sum_t \gamma^t r(s_t, a_t)$$

$$p_\theta(\tau) = p_\theta(s_1, a_1, \cdots, s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$

RL Algorithms

Source: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

SOTA in Policy Optimization (**As of 2014/2015**)

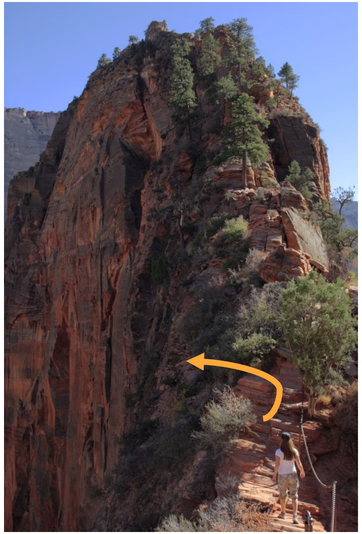
- Policy iteration methods --- #parameters linear in #states, impractical for large scale problems
- Policy gradient methods --- enjoys nice sample complexity guarantees, it's supervised learning counterparts have been successful in CV, NLP
- Derivative-free optimization methods --- very simple to understand, work (embarrassingly) the best in some classic benchmark problems

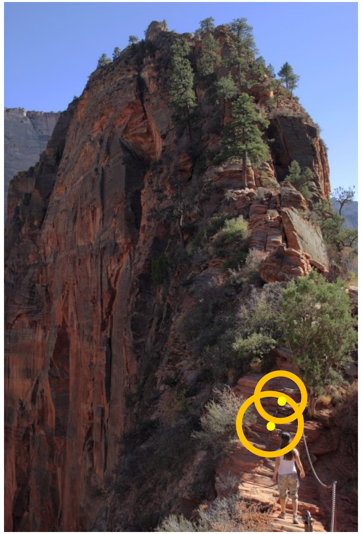
Preliminary: Policy Gradient

Objective: $J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \sum_t \gamma^t r(s_t, a_t)$

Vanilla policy gradient algorithm:

1. initialize policy π_θ
2. run policy π_θ to generate sample trajectories $\{\tau^i\}_{i=1}^N$
3. update policy: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$. Go to 2.





Problem: data used for estimating gradient come from previous policy

bad policy bad data

Core idea: Improving expected reward in each policy update

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \sum_t \gamma^t r(s_t, a_t)$$

New policy Old policy

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t r(s_t, a_t) \right] - \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t \gamma^t r(s_t, a_t) \right] \rightarrow \mathbb{E}_{s_0 \sim p(s_0)} \left[V^{\pi_\theta}(s_0) \right]$$

$$\downarrow$$

$$= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t r(s_t, a_t) \right] - \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[V^{\pi_\theta}(s_0) + \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(s_t) - \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(s_t) \right] \leftarrow \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[V^{\pi_\theta}(s_0) \right]$$

$$= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t r(s_t, a_t) \right] + \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t (\gamma V(s_{t+1}) - V^{\pi_\theta}(s_t)) \right]$$

$$= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t (r(s_t, a_t) + \gamma V(s_{t+1}) - V^{\pi_\theta}(s_t)) \right] = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

Improving expected reward in each policy update

What we have:

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)\right]$$

Problem: Can't sample trajectories from the new policy!

$$= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)}\left[\mathbb{E}_{a_t \sim \pi_{\theta'}} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]$$

$$\mathbb{E}_{s_t \sim p_\theta(s_t)}$$

Approximate using old state dist.

$$\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)}$$

Importance sampling

Importance sampling + using previous state dist.

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)\right]$$

$$= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)}\left[\mathbb{E}_{a_t \sim \pi_{\theta'}} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]$$

Importance sampling

$$= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)}\left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]$$

Approximation using old state dist.

$$\approx \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}\left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]$$

How good is the approximation?

where $\epsilon = \max_{s,a} |A_\pi(s,a)|$

$$\underbrace{J(\theta') - J(\theta) \geq \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)} \left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t) \right] - \frac{4\epsilon\gamma}{(1-\gamma)} D_{\mathrm{KL}}^{\max}(\theta, \theta')}_{\text{Maximize it?}}$$

Too slow

Reformulate it as a constraint optimization problem:

$$\max_{\theta'} \sum_t \mathbb{E}_{s \sim p_\theta(s_t)} \left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

$$\text{s.t.} \quad \max_{s \sim p_\theta(s)} \left[D_{\mathrm{KL}}(\pi_\theta(\cdot|s) \| \pi_{\theta'}(\cdot|s)) \right] \leq \delta$$

Trust Region

The optimization problem

$$\max_{\theta'} \sum_t \mathbb{E}_{s \sim p_\theta(s_t)} \left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

$$\text{s.t. } \max_{s \sim p_\theta(s)} [D_{\mathrm{KL}}(\pi_\theta(\cdot|s) \| \pi_{\theta'}(\cdot|s))] \leq \delta$$

Finally, the optimization objective:

$$\max_{\theta'} \sum_t \mathbb{E}_{s \sim p_\theta(s_t)} \left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

$$\text{s.t. } \mathbb{E}_{s \sim p_\theta(s)} [D_{\mathrm{KL}}(\pi_\theta(\cdot|s) \| \pi_{\theta'}(\cdot|s))] \leq \delta$$

The practical algorithm

$$\bar{A}(\theta') := \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)} \left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

1st order approx.

$$\begin{aligned} &\bar{A}(\theta') \\ &\approx \bar{A}(\theta) + \nabla_{\theta'} \bar{A}(\theta)^T (\theta' - \theta) \\ &\propto \nabla_{\theta'} \bar{A}(\theta)^T (\theta' - \theta) \end{aligned}$$

Solution as direction

$$\max_{\theta'} \sum_t \mathbb{E}_{s \sim p_\theta(s_t)} \left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

$$\text{s.t. } \mathbb{E}_{s \sim p_\theta(s)} \left[D_{\mathrm{KL}}(\pi_\theta(\cdot|s) \| \pi_{\theta'}(\cdot|s)) \right] \leq \delta$$

Line search

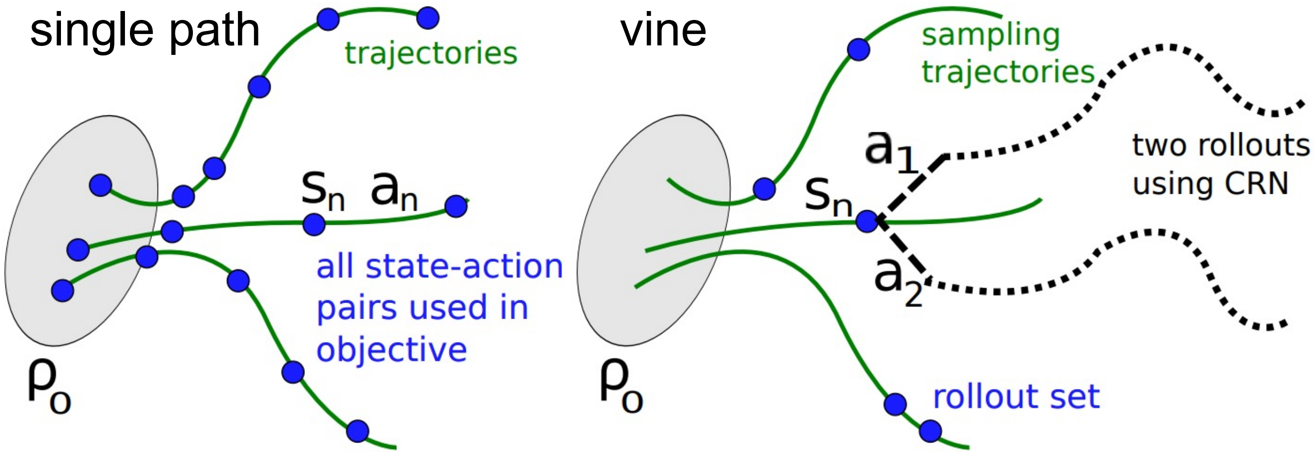
2nd order approx.

$$\frac{1}{2} (\theta' - \theta)^T \left[\frac{1}{T} \sum_{t=1}^{T} \frac{\partial^2}{\partial \theta_i \partial \theta_j} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid s_t) \| \pi_{\theta'}(\cdot \mid s_t)) \right] (\theta' - \theta) < \delta$$

The practical algorithm

How do we approximate objective using samples

$$\bar{A}(\theta') := \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)} \left[\mathbb{E}_{a_t \sim q} \frac{\pi_{\theta'}(a_t \mid s_t)}{q(a_t \mid s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$



Experimental Setup

Domains

- ❖ Simulated Robotic Locomotion: swimmer, hopper and walker on MuJoCo
- ❖ Atari gams: raw image as input

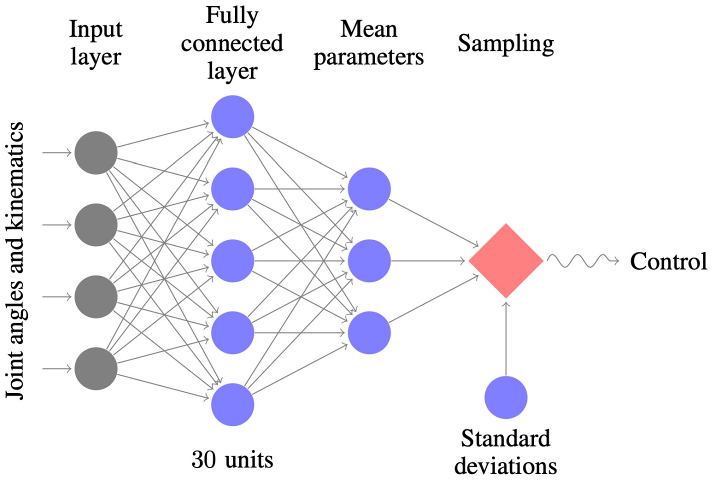
Questions to study?

- ❖ What are the performance characteristics of the single path and vine sampling procedures?
- ❖ How does TRPO compare to related prior works (e.g. natural policy gradient)?
- ❖ How does TRPO compare with other methods when applied to large-scale problems, with regard to final performance, computation time, and sample complexity?

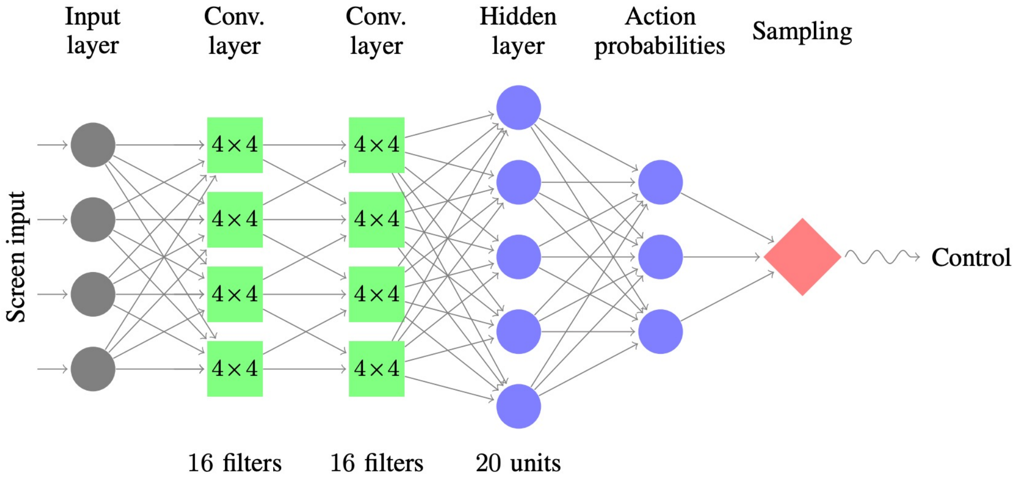
Experimental Setup

Policy network

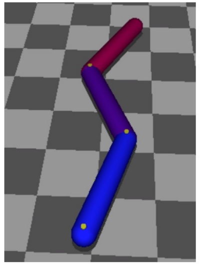
Locomotion



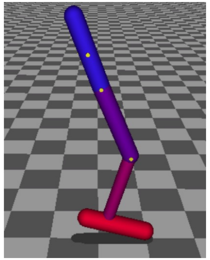
Atari games



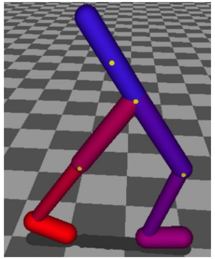
Simulated Robotic Locomotion



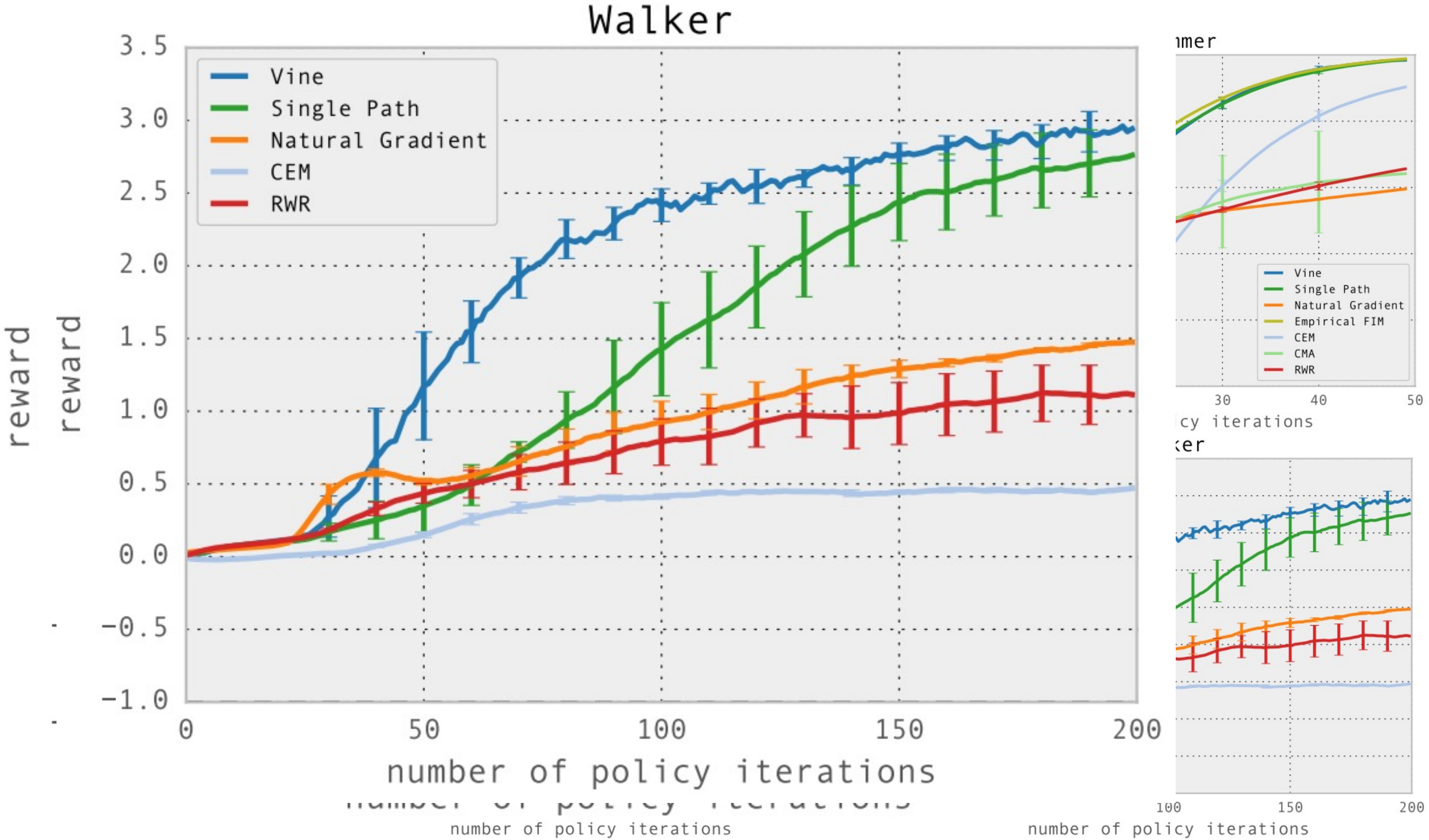
Swir
State
Con



Hop
State
Con

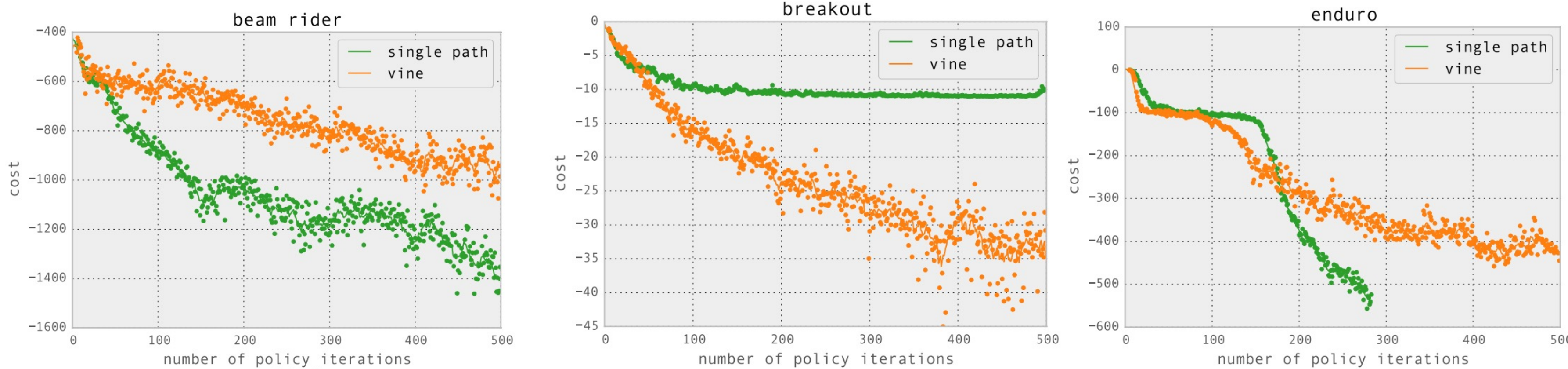


Walk
State
Con



Atari games

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Human (Mnih et al., 2013)	7456	31.0	368	−3.0	18900	28010	3690
Deep Q Learning (Mnih et al., 2013)	4092	168.0	470	20.0	1952	1705	581
UCC-I (Guo et al., 2014)	5702	380	741	21	20025	2995	692
TRPO - single path	1425.2	10.8	534.6	20.9	1973.5	1908.6	568.4
TRPO - vine	859.5	34.2	430.8	20.9	7732.5	788.4	450.2



Summary of the experiments

- Locomotion: Single path and vine TRPO solved all four locomotion problems, while natural gradient method, derivative-free methods only work on two of the easier problem (i.e. Cartpole and swimmer).

- Atari games: TRPO consistently achieves reasonable result across different games and outperforms prior methods in some of them.

Discussion

Pros

- Theory side: provably monotonic policy improvement
- Empirical side:

    1. First work that learns controllers from scratch for all four locomotion tasks; the algorithm is very general and doesn’t require on any hand-architected policy classes.

    2. the same algorithm also works well on image-based Atari game playing, which again shows the generality of TRPO.

Cons

- Second order optimization, therefore it is relatively complicated and not very compatible with modern automatic differentiation packages
- Not very compatible with modern distributed training paradigm.

Subsequent work

- J. Schulman, et al. "Proximal policy optimization algorithms.", 2017.

  Proximal policy optimization, or PPO, improves upon TRPO. PPO attains the data efficiency and reliable performance of TRPO, but much simpler and only use first order gradient

Other related work

- R. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning.", 1992.
- S. Kakade. "A Natural Policy Gradient.", 2001.
- S. Kakade and J. Langford. "Approximately optimal approximate reinforcement learning", 2002.
- J. Schulman, et al. "High-dimensional continuous control using generalized advantage estimation.", 2015
- T.P. Lillicrap, et al. "Continuous control with deep reinforcement learning." , 2015.
- T. Haarnoja, et al. "Soft actor-critic algorithms and applications.", 2018.