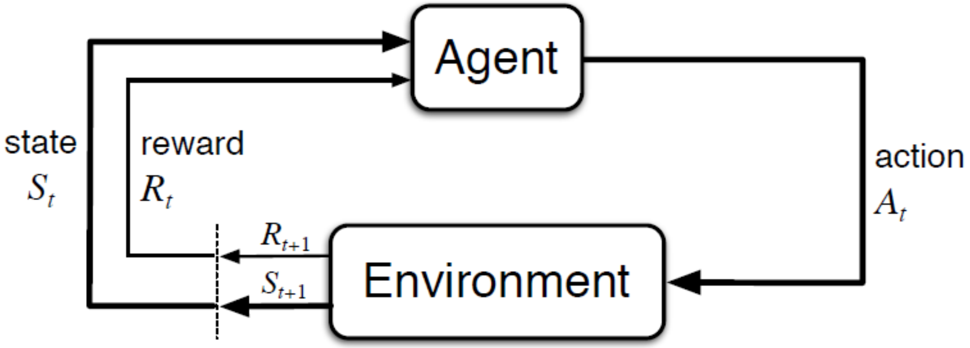
Proximal Policy Optimization

YIKANG, GANG

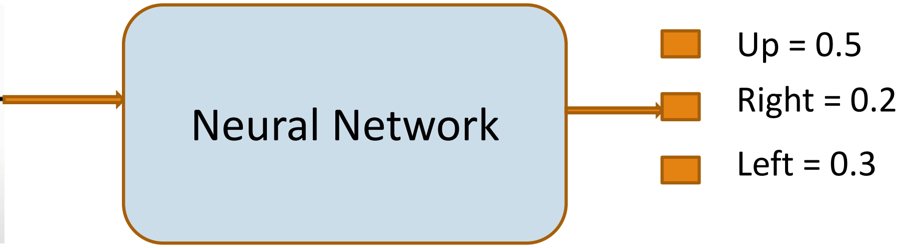
Outline

1. Background (Gang)
2. Theoretical Part (YiKang)
3. Algorithm and Experiment (Gang)

Review of Policy Gradient



- $\nabla J(\theta) = E_{\pi(a|S;\,\theta)}[R\,\nabla \log \pi(a|S;\,\theta)]$
- Reinforce: R = G_t
- Actor-Critic: R = $Q(s,a;w)$ or $\Delta V(s)$
- A2C: R = $A(s,a;w)$



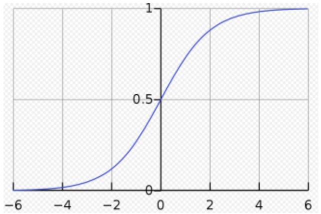
$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta)$$

Limitations of Policy Gradient

Data Inefficient

- Sample once, update once
- On-policy learning can be extremely inefficient.

Not Robust



Consider a family of policies with parametrization:

$$\pi_\theta(a) = \begin{cases} \sigma(\theta) & a = 1 \\ 1 - \sigma(\theta) & a = 2 \end{cases}$$

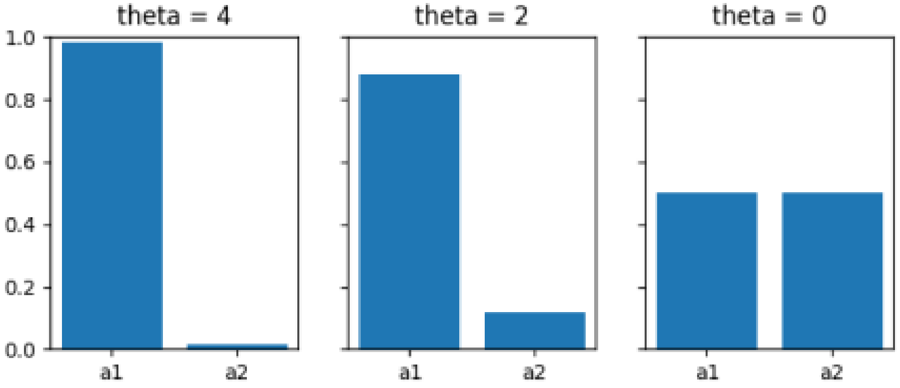
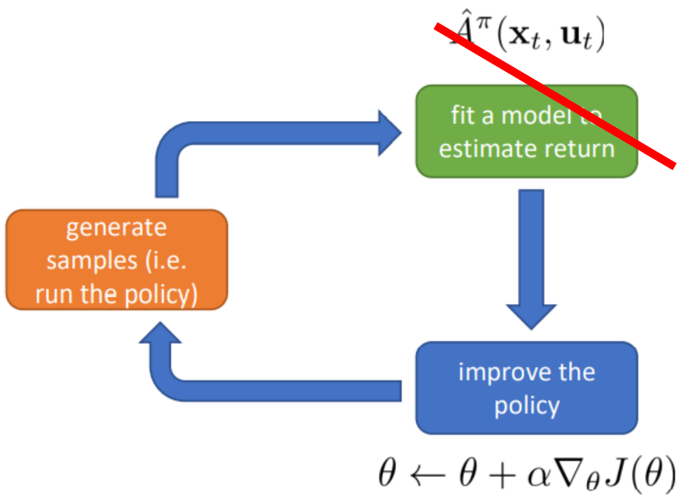


Figure: Small changes in the policy parameters can unexpectedly lead to **big** changes in the policy.

On-Policy -> Off-Policy



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\cancel{\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)}$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

$$\nabla J(\theta) = E_{\pi(a|S;\,\theta)}\left[A^\theta(s,a)\,\nabla \log \pi(a|S;\,\theta)\right]$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ.

θ' is fixed, so we can collect a batch of sample data and then training.

Two Important Classes

- Line search methods
  - Find a direction of improvement
  - Select a step length

- Trust region methods
  - Select a trust region (analog to max step length)
  - Find a point of improvement in the region

Trust Region Methods

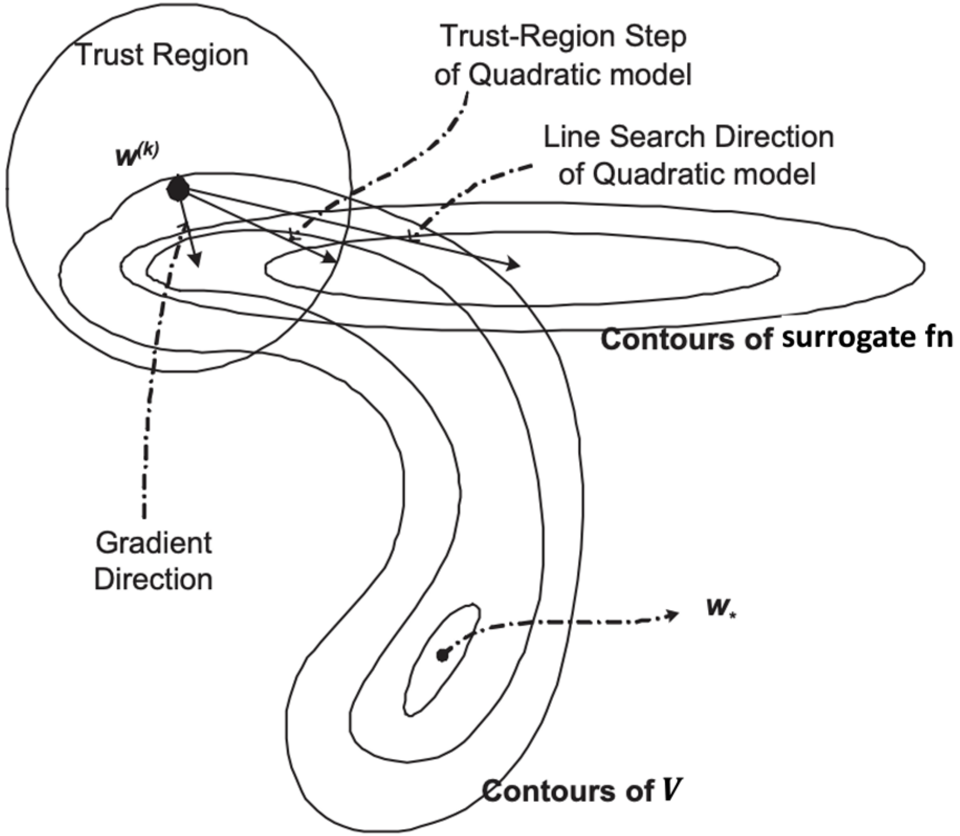
- Idea:
  - Approximate objective f with a simpler objective $\tilde{f}$
  - Solve $\tilde{x}^* = argmin_x \tilde{f}(x)$

- **Problem:** The optimum $\widetilde{x}^*$ might be in a region where $\tilde{f}$ poorly approximates f and therefore $\tilde{x}^*$ might be far from optimal
- **Solution:** restrict the search to a region where we trust $\tilde{f}$ to approximate f well.
  - Solve $\tilde{x}^* = argmin_{x \in trustRegion} \tilde{f}(x)$

Trust Region Methods

- We often optimize a surrogate objective (approximation of V)
- Surrogate objective may be trustable (close to V) only in a small region

- Limit search to small trust region



Choi, Choi (2005)

Trust Region for Policies

- Let θ be the parameters for policy $\pi_\theta(s|a)$
- We can define a region
  around θ: $\{\theta' |\, D(\theta, \theta') < \delta\}$
  or around π_θ: $\{\theta' |\, D(\pi_\theta, \pi_{\theta'}) < \delta\}$
  where D is a distance measure
- V often varies more smoothly with π_θ than θ
  small change in π_θ $\xrightarrow{\text{usually}}$ small change in V
  small change in θ $\xrightarrow{\text{more often}}$ large change in V
- Hence, define policy trust regions

Why Does Policy Gradient Work?

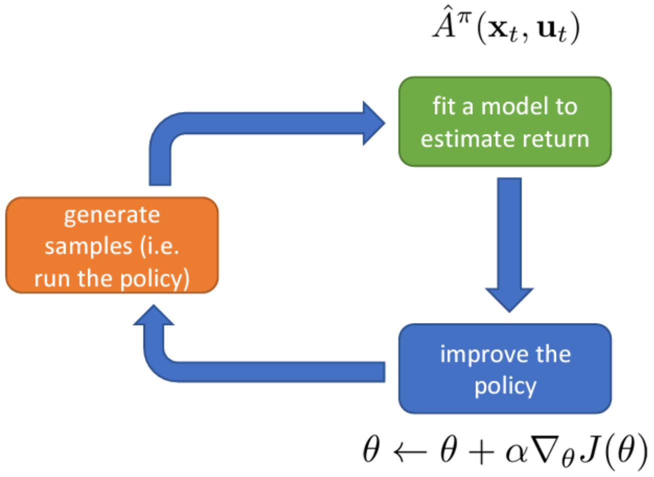
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{A}^\pi_{i,t}$$

1. Estimate $\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t)$ for current policy π
2. Use $\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t)$ to get *improved* policy π'

look familiar?

policy iteration algorithm:

1. evaluate $A^\pi(\mathbf{s}, \mathbf{a})$
2. set $\pi \leftarrow \pi'$



Policy Gradient as Policy Iteration

$$J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$\begin{aligned}
J(\theta') - J(\theta) &= J(\theta') - E_{\mathbf{s}_0 \sim p(\mathbf{s}_0)}\left[V^{\pi_\theta}(\mathbf{s}_0)\right] \\
&= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)}\left[V^{\pi_\theta}(\mathbf{s}_0)\right] \\
&= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t V^{\pi_\theta}(\mathbf{s}_t) - \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(\mathbf{s}_t)\right] \\
&= J(\theta') + E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t))\right] \\
&= E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right] + E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t))\right] \\
&= E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t))\right] \\
&= E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]
\end{aligned}$$

Policy Gradient as Policy Iteration

$$J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]$$

expectation under $\pi_{\theta'}$

advantage under π_θ

importance sampling

$$\begin{aligned} E_{x \sim p(x)}[f(x)] &= \int p(x) f(x) dx \\ &= \int \frac{q(x)}{q(x)} p(x) f(x) dx \\ &= \int q(x) \frac{p(x)}{q(x)} f(x) dx \\ &= E_{x \sim q(x)}\left[\frac{p(x)}{q(x)} f(x)\right] \end{aligned}$$

$$\begin{aligned} E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right] &= \sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)}\left[E_{\mathbf{a}_t \sim \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}\left[\gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]\right] \\ &= \sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)}\left[\boxed{E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\right.}\gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]\right] \end{aligned}$$

is it OK to use $p_\theta(\mathbf{s}_t)$ instead?

Bounding The Distribution Change

Claim: $p_\theta(\mathbf{s}_t)$ is *close* to $p_{\theta'}(\mathbf{s}_t)$ when π_θ is *close* to $\pi_{\theta'}$

Simple case: assume π_θ is a *deterministic* policy $\mathbf{a}_t = \pi_\theta(\mathbf{s}_t)$

$\pi_{\theta'}$ is *close* to π_θ if $\pi_{\theta'}(\mathbf{a}_t \neq \pi_\theta(\mathbf{s}_t)|\mathbf{s}_t) \leq \epsilon$

$$p_{\theta'}(\mathbf{s}_t) = \underbrace{(1-\epsilon)^t}_{\text{probability we made no mistakes}} p_\theta(\mathbf{s}_t) + (1-(1-\epsilon)^t)\underbrace{p_{\text{mistake}}(\mathbf{s}_t)}_{\text{some } \textit{other} \text{ distribution}}$$

$$|p_{\theta'}(\mathbf{s}_t) - p_\theta(\mathbf{s}_t)| = (1-(1-\epsilon)^t)|p_{\text{mistake}}(\mathbf{s}_t) - p_\theta(\mathbf{s}_t)| \leq 2(1-(1-\epsilon)^t)$$

useful identity: $(1-\epsilon)^t \geq 1 - \epsilon t$ for $\epsilon \in [0, 1]$

$$\leq 2\epsilon t$$

What We Get So Far?

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

such that $|\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) - \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)| \le \epsilon$

for small enough ϵ, this is guaranteed to improve $J(\theta') - J(\theta)$

- $\nabla J(\theta) = E_{\pi(a|S;\,\theta)}[A^\theta(s,a)\,\nabla \log \pi(a|S;\ \theta)]$
- $\nabla J(\theta) = E_{\pi'(a|S;\,\theta')}\left[\boxed{\frac{\pi(a|S;\theta)}{\pi'(a|S;\,\theta')}} A^{\theta'}(s,a) \nabla \log \pi(a|S;\theta)\right]$
- Sample the data from θ'.
- Use the data to train θ many times.
- $\hat{J}(\theta) = E_{\pi'(a|S;\,\theta')}\left[\frac{\pi(a|S;\theta)}{\pi'(a|S;\,\theta')} A^{\theta'}(s,a)\right]$

PPO Algorithm (Adaptive KL Penalty)

Trust Region Policy Optimization (TRPO)

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

- Computation is complicated.

Proximal Policy Optimization (PPO)

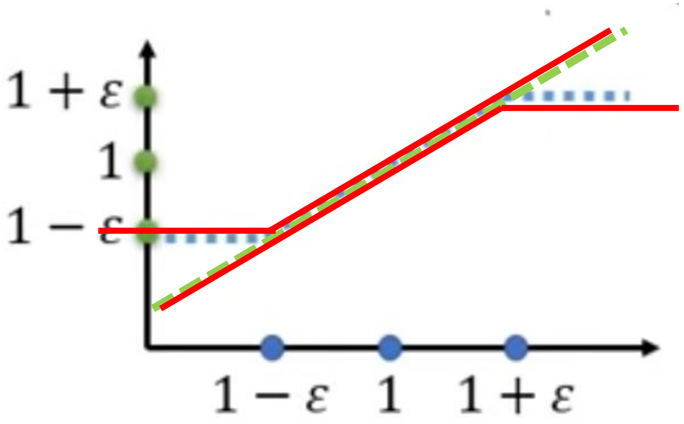
$$\underset{\theta}{\text{maximize}}\, \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta\, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right]$$

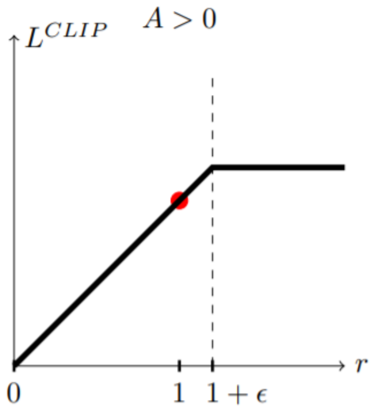
- Compute $d = \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$
  - If $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$
  - If $d > d_{\text{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$

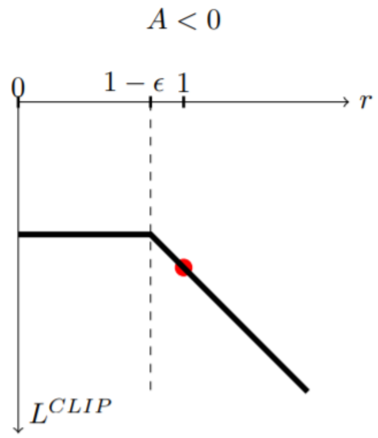
PPOv2 Algorithm (Clip)

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$







Comparison

No clipping or penalty:	$L_t(\theta) = r_t(\theta)\hat{A}_t$
Clipping:	$L_t(\theta) = \min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta)), 1-\epsilon, 1+\epsilon)\hat{A}_t$
KL penalty (fixed or adaptive)	$L_t(\theta) = r_t(\theta)\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\mathrm{old}}}, \pi_\theta]$

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\mathrm{targ}} = 0.003$	0.68
Adaptive KL $d_{\mathrm{targ}} = 0.01$	0.74
Adaptive KL $d_{\mathrm{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

PPOv2 Pseudocode

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

PPO-PyTorch/PPO.py at master · nikhilbarhate99/PPO-PyTorch (github.com)

Experiments in Continuous Domain

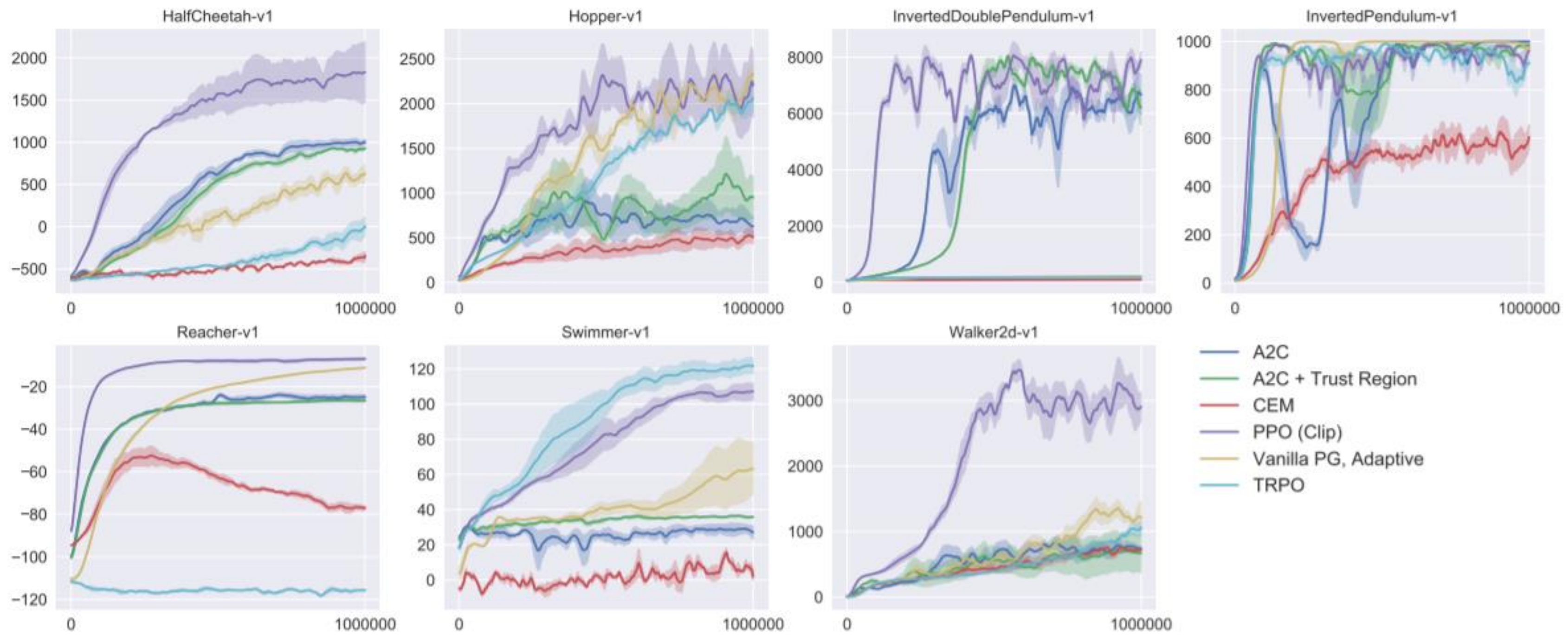


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

Experiments in Atari Domain

	A2C	ACER	PPO	Tie
(1) avg. episode reward over all of training	1	18	**30**	0
(2) avg. episode reward over last 100 episodes	1	**28**	19	1

Table 2: Number of games "won" by each algorithm, where the scoring metric is averaged across three trials.

Reference

1. Schulman, John et al. “Trust Region Policy Optimization.” *ArXiv* abs/1502.05477 (2015): n. pag.
2. Schulman, John et al. “Proximal Policy Optimization Algorithms.” *ArXiv* abs/1707.06347 (2017): n. pag.
3. Proximal Policy Optimization — Spinning Up documentation (openai.com)
4. PowerPoint Presentation (berkeley.edu)