Proximal Policy Optimization (cont.)

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, ...$ **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:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$
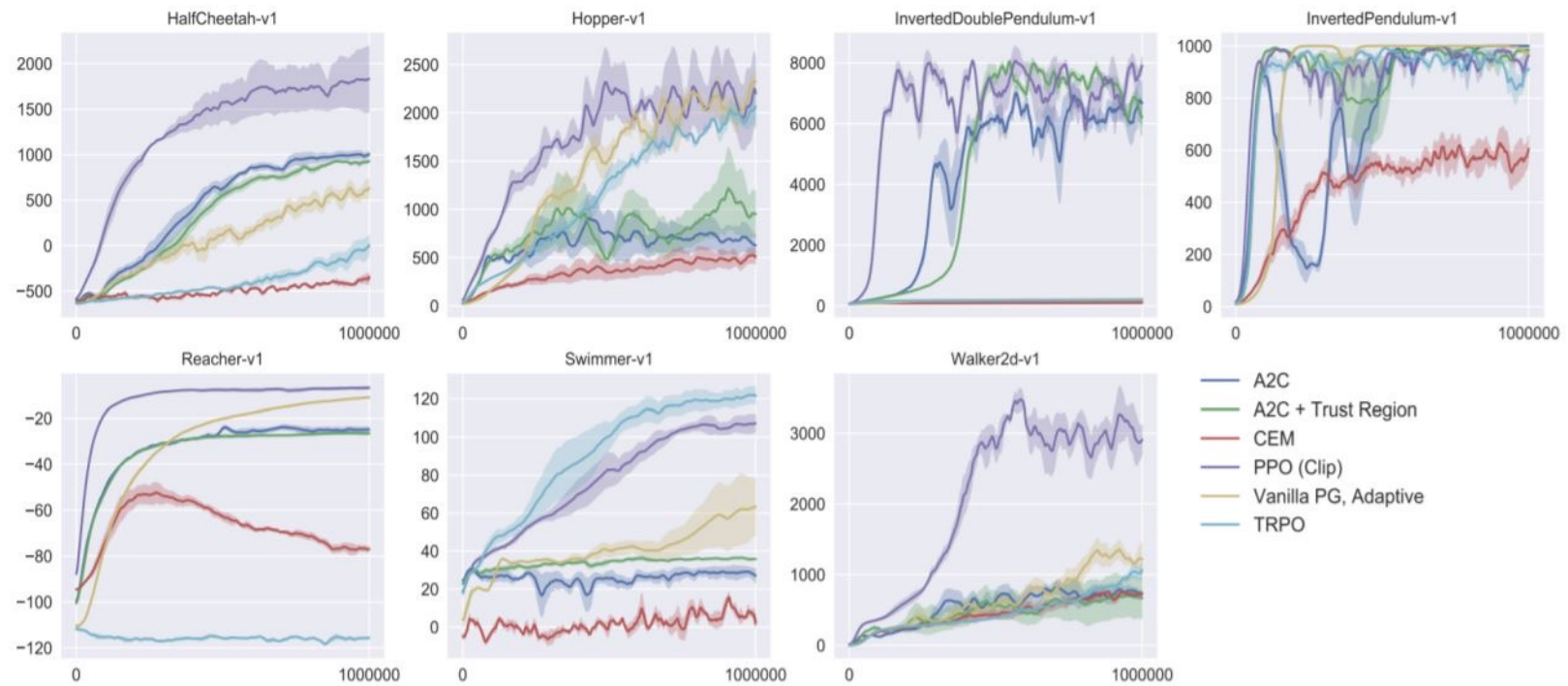
    typically via stochastic gradient ascent with Adam.
7:     Fit value function by regression on mean-squared error:

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

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

Review

- Policy gradient = policy iteration

- Optimize advantage under new policy state distribution

- Using old policy state distribution optimizes a bound, if the policies are close enough

- Results in constrained optimization problem

- Gradient ascent: first order approximation to objective

- Regular gradient ascent has the wrong constraint (in parameter space): use natural gradient with D_{KL} constraint

- PPO simply uses the importance sampling ratio for regularization