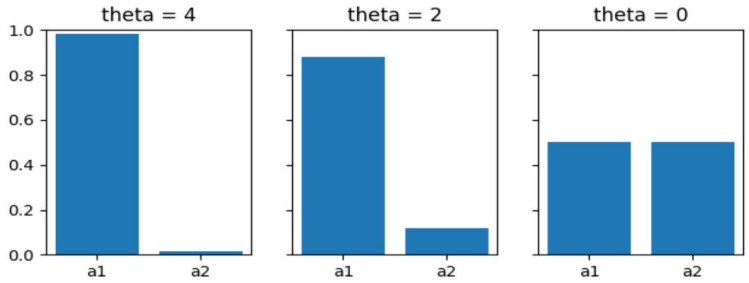
Constrained Policy Optimization

B. Ravindran

- In this lecture we will be mostly dealing with On-Policy algorithms
- Some issues with such algorithms
  - Sample inefficient: we cannot reuse data collected by old policies
  - Difficult to choose step size
    - Step too big ==> Bad policy ==> Data collected under bad policy ==> Hard to recover
    - Step too small ==> Inefficient use of experience

Consider a family of policies with parametrization:

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



$$\sigma(\theta) = \frac{1}{1 + e^{-\theta}}$$

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

Constrained Policy Optimization

- Take the largest possible step to improve policy performance, while satisfying a constraint on how far the new and old policies are allowed to be.
- The distance between the new and old policies is expressed in terms of **KL-Divergence**, a measure of distance between probability distributions.

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\,\log\left(\frac{P(i)}{Q(i)}\right)$$

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x)\,\log\left(\frac{p(x)}{q(x)}\right)\,dx$$

Constrained Policy Optimization

- Natural Policy Gradient, Kakade et al
- Natural actor-critic, Peters et al
- Trust Region Policy Optimization (TRPO), Schulman et al

Second-order methods

- Proximal Policy Optimization (PPO), Schulman et al

First-order method

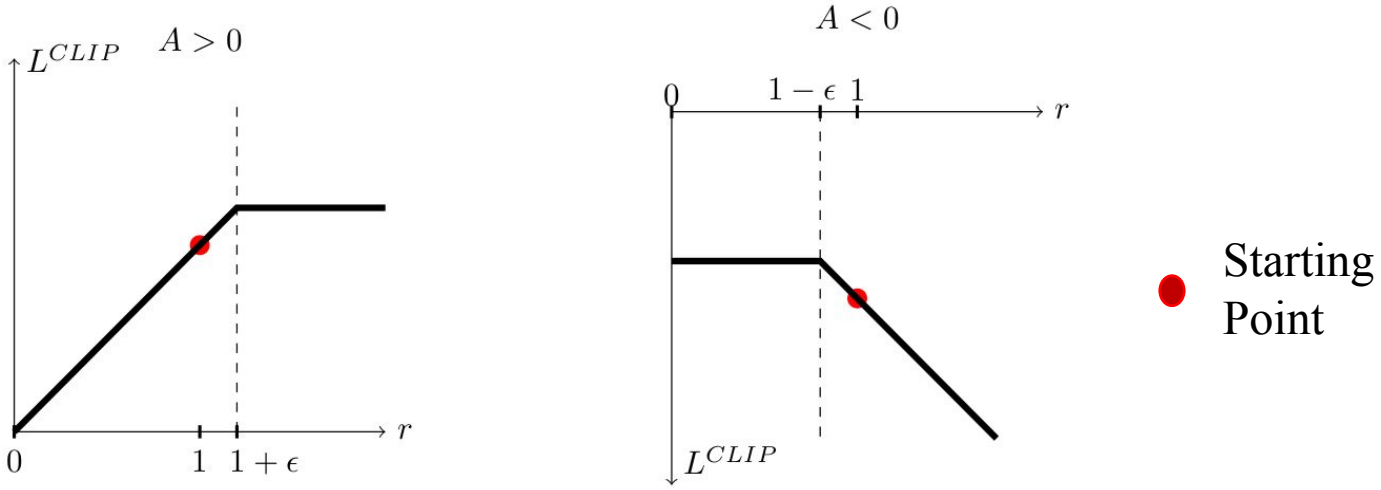
Proximal Policy Optimization (PPO)

- Consider the probability ratio between the old policy and the new policy

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

- The Clipped Objective is defined as

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



- The new policy does not benefit by going far away from the old policy.

PPO algorithm

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k} \left[\sum_{t=0}^{T} \left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k}) \right] \right]$$

end for

Results

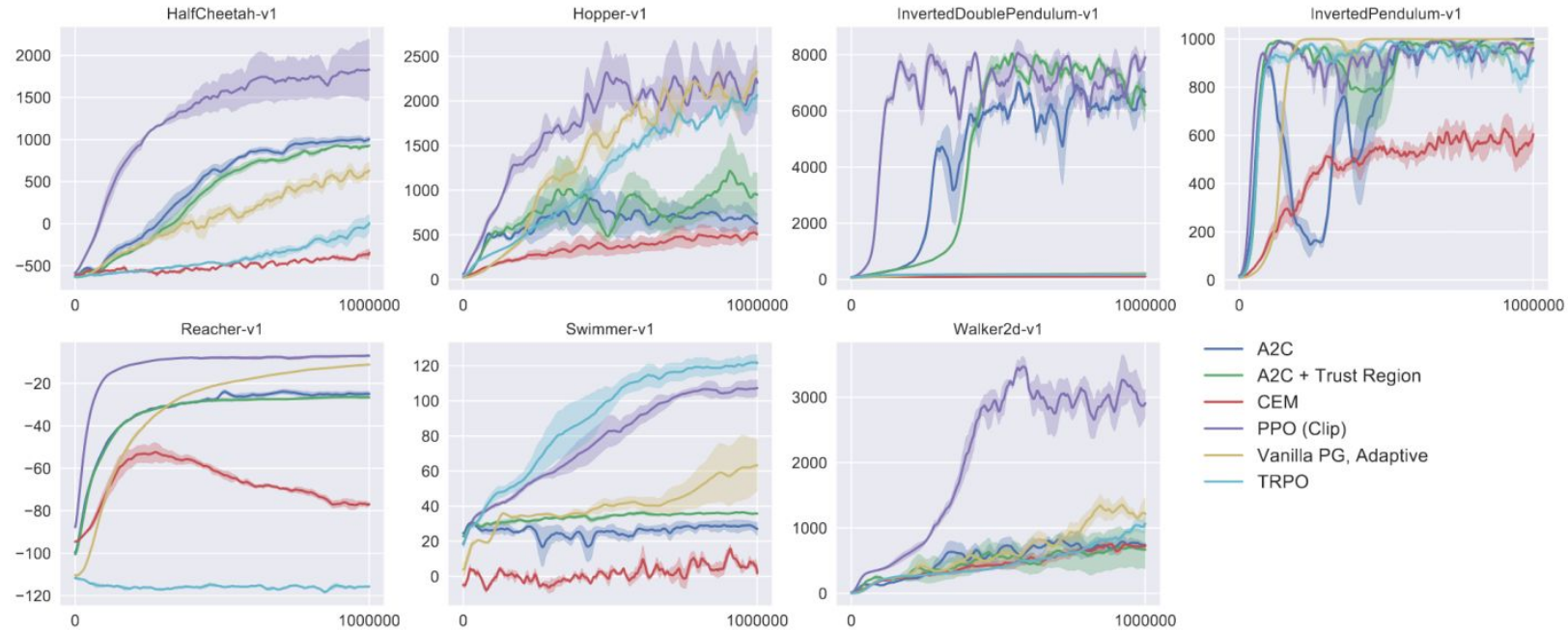


Figure: Performance comparison between PPO with clipped objective and various other deep RL methods on a slate of MuJoCo tasks. [10]

Implementation details

- Typically, clipping threshold ϵ = 0.2.
- Advantage is estimated using generalized advantage estimation

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$
$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

- Observations are normalized and then clipped to a range (usually between -10 and 10). This is achieved by maintaining a running mean and variance of observations coming in from the environment.
- Early stopping - calculate approximate KL divergence between the current policy and the old policy, and stop the updates of the current epoch if the approximate KL divergence exceeds some preset threshold.

Implementation details

- Separate actor and critic networks.
- tanh activation functions.
- Orthogonal initialization of actor, critic networks with appropriate scaling.
- Entropy term not included in actor loss function.
- Gradient clipping - ensure that the norm of the concatenated gradients of all network parameters does not exceed 0.5.
- Generalized advantage estimates are normalized at the batch level.

Implementation details

- Reward scaling - rewards are divided by the standard deviation of a rolling discounted sum of the rewards, followed by clipping to a range (usually between -10 and 10). Not used frequently.
- Value function clipping - the value function loss is clipped in a manner that is similar to the clipped objective. Not used frequently.

Thank you !!