

Lecture 8: Policy Gradient II. Advanced policy gradient section slides from Joshua Achiam (OpenAI)'s slides, with minor modifications

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2023

- Select all that are true about policy gradients:

- 1 $\nabla_{\theta} V(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) Q^{\pi_{\theta}}(s, a)]$ T
- 2 θ is always increased in the direction of $\nabla_{\theta} \ln(\pi(S_t, A_t, \theta))$. F
- 3 State-action pairs with higher estimated Q values will increase in probability on average T
- 4 Are guaranteed to converge to the global optima of the policy class F
- 5 Not sure

- Select all that are true about policy gradients:

Sutton & Barto pg 387

- 1 $\nabla_{\theta} V(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) Q^{\pi_{\theta}}(s, a)]$
- 2 θ is always increased in the direction of $\nabla_{\theta} \ln(\pi(S_t, A_t, \theta))$.
- 3 State-action pairs with higher estimated Q values will increase in probability on average
- 4 Are guaranteed to converge to the global optima of the policy class
- 5 Not sure

1 and 3 are true. The direction of θ also depends on the Q -values / returns. We are only guaranteed to reach a local optima

Vanilla Policy Gradient

"Vanilla" Policy Gradient Algorithm

Initialize policy parameter θ , baseline b

for iteration= $1, 2, \dots$ **do**

Collect a set of trajectories by executing the current policy π_θ

At each timestep t in each trajectory τ^i , compute

Return $\underline{G}_t^i = \sum_{t'=t}^{T-1} r_{t'}^i$, and

Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t)$.

Re-fit the baseline, by minimizing $\sum_i \sum_t \|b(s_t) - G_t^i\|^2$,

Update the policy, using a policy gradient estimate \hat{g} ,

Which is a sum of terms $\nabla_\theta \log \pi(a_t | s_t, \theta) \hat{A}_t^i$.

(Plug \hat{g} into SGD or ADAM)

endfor

Q^π

- Policy gradient:

$$\nabla_{\theta} \mathbb{E}[R] \approx (1/m) \sum_{i=1}^m \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t, s_t) (\underbrace{G_t^{(i)}} - b(s_t))$$

- Fixes that improve simplest estimator
 - Temporal structure (shown in above equation)
 - Baseline (shown in above equation)
 - **Alternatives to using Monte Carlo returns G_t^i as estimate of expected discounted sum of returns for the policy parameterized by θ ?**

- G_t^i is an estimation of the value function at s_t from a single roll out
- Unbiased but high variance
- Reduce variance by introducing bias using bootstrapping and function approximation
 - Just like in we saw for TD vs MC, and value function approximation

policy θ	value/ Q w
actor	critic

- Estimate of V/Q is done by a **critic**
- **Actor-critic** methods maintain an explicit representation of policy and the value function, and update both
- A3C (Mnih et al. ICML 2016) is a very popular actor-critic method

- Recall:

$$\nabla_{\theta} \mathbb{E}_{\tau}[R] = \mathbb{E}_{\tau} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t | s_t; \theta) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right]$$

$$\nabla_{\theta} \mathbb{E}_{\tau}[R] \approx \mathbb{E}_{\tau} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t | s_t; \theta) \left(\underbrace{Q(s_t, a_t; \mathbf{w})}_{\text{value function}} - b(s_t) \right) \right]$$

- Letting the baseline be an estimate of the value V , we can represent the gradient in terms of the state-action advantage function

$$\nabla_{\theta} \mathbb{E}_{\tau}[R] \approx \mathbb{E}_{\tau} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t | s_t; \theta) \underbrace{\hat{A}^{\pi}(s_t, a_t)}_{\text{advantage function}} \right]$$

- where the advantage function $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$

$$Q^{\pi}(s, a; \mathbf{w}) - V^{\pi}(s, \mathbf{w})$$

Advanced Policy Gradients

Theory:

- 1 Problems with Policy Gradient Methods
- 2 Policy Performance Bounds
- 3 Monotonic Improvement Theory

Algorithms:

- 1 Proximal Policy Optimization

The Problems with Policy Gradients

Policy gradient algorithms try to solve the optimization problem

$$\max_{\theta} J(\pi_{\theta}) \doteq \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

$$\tau = (s_0, a_0, s_1, \dots)$$

by taking stochastic gradient ascent on the policy parameters θ , using the *policy gradient*

$$g = \nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) A^{\pi_{\theta}}(s_t, a_t) \right].$$

Limitations of policy gradients:

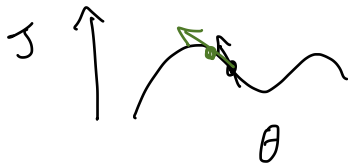
- Sample efficiency is poor
- Distance in parameter space \neq distance in policy space!
 - What is policy space? For tabular case, set of matrices

$$\Pi = \left\{ \pi : \pi \in \mathbb{R}^{|S| \times |A|}, \sum_a \pi_{sa} = 1, \pi_{sa} \geq 0 \right\}$$

$$p(a_t | s_t)$$

- Policy gradients take steps in parameter space
- Step size is hard to get right as a result

Sample Efficiency in Policy Gradients



- Sample efficiency for vanilla policy gradient methods is poor
- Discard each batch of data immediately after **just one gradient step**
- Why? PG is an **on-policy expectation**.
- Two main approaches to obtaining an unbiased estimate of the policy gradient
 - Run policy in environment and collect sample trajectories, then form sample estimate. (More stable)
 - Use trajectories from other policies with **importance sampling**. (Less stable)

Importance Sampling

Importance sampling is a technique for estimating expectations using samples drawn from a different distribution.

$$\begin{aligned} \mathbb{E}_{x \sim p} [f(x)] &= \int_{\mathcal{X}} p(x) f(x) dx \\ &= \int_{\mathcal{X}} p(x) \frac{q(x)}{q(x)} f(x) dx \quad \leftarrow \text{if } p(x) > 0 \text{ then } q(x) > 0 \\ &= \int_{\mathcal{X}} q(x) \left[\frac{p(x)}{q(x)} f(x) \right] dx \\ &= \mathbb{E}_{x \sim q} \left[\frac{p(x)}{q(x)} f(x) \right] \end{aligned}$$

Importance sampling is a technique for estimating expectations using samples drawn from a different distribution.

$$\mathbb{E}_{x \sim P}[f(x)] = \mathbb{E}_{x \sim Q} \left[\frac{P(x)}{Q(x)} f(x) \right] \approx \frac{1}{|D|} \sum_{x \in D} \frac{P(x)}{Q(x)} f(x), \quad \text{dataset } D \sim Q$$

The ratio $P(x)/Q(x)$ is the **importance sampling weight** for x .

$$\begin{aligned} \text{var}(\hat{f}) &= \frac{1}{N} \text{var} \left(\frac{P(x)}{Q(x)} f(x) \right) \\ &= \frac{1}{N} \left(\mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)}^2 f(x)^2 \right) - \left(\mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)} f(x) \right) \right)^2 \right) \end{aligned}$$

Importance Sampling

Importance sampling is a technique for estimating expectations using samples drawn from a different distribution.

$$\mathbb{E}_{x \sim P}[f(x)] = \mathbb{E}_{x \sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx \frac{1}{|D|} \sum_{x \in D} \frac{P(x)}{Q(x)}f(x), \quad D \sim Q$$

The ratio $P(x)/Q(x)$ is the **importance sampling weight** for x .

What is the variance of an importance sampling estimator?

$$\begin{aligned} \text{var}(\hat{\mu}_Q) &= \frac{1}{N} \text{var}\left(\frac{P(x)}{Q(x)}f(x)\right) \\ &= \frac{1}{N} \left(\mathbb{E}_{x \sim Q}\left[\left(\frac{P(x)}{Q(x)}f(x)\right)^2\right] - \mathbb{E}_{x \sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right]^2 \right) \\ &= \frac{1}{N} \left(\mathbb{E}_{x \sim P}\left[\frac{P(x)}{Q(x)}f(x)^2\right] - \mathbb{E}_{x \sim P}[f(x)]^2 \right) \end{aligned}$$

The term in red is problematic—if $P(x)/Q(x)$ is large in the wrong places, the variance of the estimator explodes.

Importance Sampling for Policy Gradients

Here, we compress the notation π_θ down to θ in some places for compactness.

$$\begin{aligned}
 \underline{g} &= \underline{\nabla_\theta J(\theta)} = \underline{\mathbb{E}_{\tau \sim \theta} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^\theta(s_t, a_t) \right]} \\
 &= \sum_{\tau} \sum_{t=0}^{\infty} \gamma^t \underline{P(\tau_t | \theta)} \nabla_\theta \log \pi_\theta(a_t | s_t) \underline{A^\theta(s_t, a_t)} \\
 &= \underline{\mathbb{E}_{\tau \sim \theta'}} \left[\sum_{t=0}^{\infty} \underline{\frac{P(\tau_t | \theta)}{P(\tau_t | \theta')}} \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^\theta(s_t, a_t) \right]
 \end{aligned}$$

$$\begin{aligned}
 \frac{P(\tau_t | \theta)}{P(\tau_t | \theta')} &= \frac{\cancel{\mu(s_0)} \cancel{\prod_{t'=1}^t \pi_\theta(a_{t'} | s_{t'})} \pi_\theta(a_t | s_t) P(s_{t+1} | s_t, a_t)}{\cancel{\mu(s_0)} \cancel{\prod_{t'=1}^t \pi_{\theta'}(a_{t'} | s_{t'})} \pi_{\theta'}(a_t | s_t) P(s_{t+1} | s_t, a_t)} \\
 &= \prod_{t'=1}^t \frac{\pi_\theta(a_{t'} | s_{t'})}{\pi_{\theta'}(a_{t'} | s_{t'})}
 \end{aligned}$$

Importance Sampling for Policy Gradients


Here, we compress the notation π_θ down to θ in some places for compactness.

$$\begin{aligned} g &= \nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \theta} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^\theta(s_t, a_t) \right] \\ &= \sum_{\tau} \sum_{t=0}^{\infty} \gamma^t P(\tau_t | \theta) \nabla_\theta \log \pi_\theta(a_t | s_t) A^\theta(s_t, a_t) \\ &= \mathbb{E}_{\tau \sim \theta'} \left[\sum_{t=0}^{\infty} \frac{P(\tau_t | \theta)}{P(\tau_t | \theta')} \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^\theta(s_t, a_t) \right] \end{aligned}$$

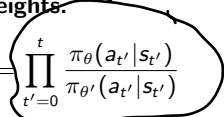
$$\frac{P(\tau_t | \theta)}{P(\tau_t | \theta')} =$$

Importance Sampling for Policy Gradients

Here, we compress the notation π_θ down to θ in some places for compactness.

$$\begin{aligned} g &= \nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \theta} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^\theta(s_t, a_t) \right] \\ &= \sum_{\tau} \sum_{t=0}^{\infty} \gamma^t P(\tau_t | \theta) \nabla_\theta \log \pi_\theta(a_t | s_t) A^\theta(s_t, a_t) \\ &= \mathbb{E}_{\tau \sim \theta'} \left[\sum_{t=0}^{\infty} \frac{P(\tau_t | \theta)}{P(\tau_t | \theta')} \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^\theta(s_t, a_t) \right] \end{aligned}$$


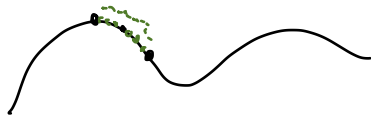
Challenge? **Exploding or vanishing importance sampling weights.**

$$\frac{P(\tau_t | \theta)}{P(\tau_t | \theta')} = \frac{\mu(s_0) \prod_{t'=0}^t P(s_{t'+1} | s_{t'}, a_{t'}) \pi_\theta(a_{t'} | s_{t'})}{\mu(s_0) \prod_{t'=0}^t P(s_{t'+1} | s_{t'}, a_{t'}) \pi_{\theta'}(a_{t'} | s_{t'})} = \prod_{t'=0}^t \frac{\pi_\theta(a_{t'} | s_{t'})}{\pi_{\theta'}(a_{t'} | s_{t'})}$$


Even for policies only slightly different from each other, **many small differences multiply to become a big difference.**

Big question: how can we make efficient use of the data we already have from the old policy, while avoiding the challenges posed by importance sampling?

Choosing a Step Size for Policy Gradients

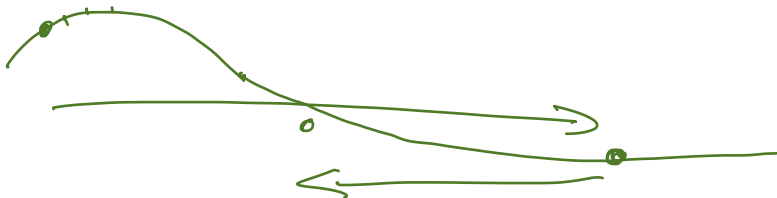


Policy gradient algorithms are stochastic gradient ascent:

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$$

with step $\Delta_k = \alpha_k \hat{g}_k$.

- If the step is too large, **performance collapse** is possible (Why?)



Choosing a Step Size for Policy Gradients

Policy gradient algorithms are stochastic gradient ascent:

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$$

with step $\Delta_k = \alpha_k \hat{g}_k$.

- If the step is too large, **performance collapse** is possible (Why?)
- If the step is too small, progress is unacceptably slow
- “Right” step size changes based on θ

Automatic learning rate adjustment like advantage normalization, or Adam-style optimizers, can help. But does this solve the problem?

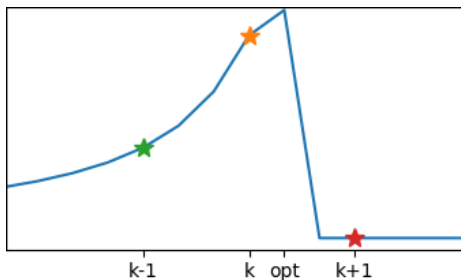


Figure: Policy parameters on x-axis and performance on y-axis. A bad step can lead to performance collapse, which may be hard to recover from.

The Problem is More Than Step Size

Consider a family of policies with parametrization:

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

softmax
 $p(a) = \frac{e^{\theta x_a}}{\sum_a e^{\theta x_a}}$

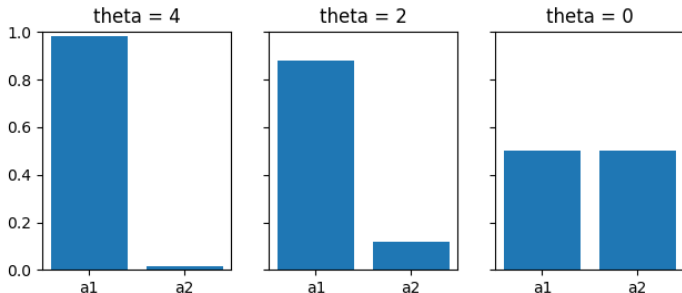


Figure: Small changes in the policy parameters can unexpectedly lead to **big** changes in the policy.

Big question: how do we come up with an update rule that doesn't ever change the policy more than we meant to?

Policy Performance Bounds

Relative Performance of Two Policies

In a policy optimization algorithm, we want an update step that

- uses rollouts collected from the most recent policy as efficiently as possible,
- and takes steps that respect **distance in policy space** as opposed to distance in parameter space.

To figure out the right update rule, we need to exploit relationships between the performance of two policies.

Performance difference lemma: In CS234 HW2 you proved that for any policies π, π'

$$J(\pi') - J(\pi) = \mathbb{E}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right] \quad (1)$$

$$= \frac{1}{1-\gamma} \mathbb{E}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'}} [A^{\pi}(s, a)] \quad (2)$$

where

$$d^{\pi}(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi)$$

What is it good for?

Can we use this for policy improvement, where π' represents the new policy and π represents the old one?

$$\begin{aligned} \max_{\pi'} J(\pi') &= \max_{\pi'} J(\pi') - J(\pi) \\ &= \max_{\pi'} \mathbb{E}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right] \end{aligned}$$

↖ returns from π

This is suggestive, but not useful yet.

Nice feature of this optimization problem: defines the performance of π' in terms of the advantages from π !

But, problematic feature: still requires trajectories sampled from π' ...

Looking at it from another angle...

In terms of the **discounted future state distribution** d^π , defined by

$$d^\pi(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi),$$

samples from π reflect $d^\pi(s)$

we can rewrite the relative policy performance identity:

$$\begin{aligned} J(\pi') - J(\pi) &= \mathbb{E}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^\pi(s_t, a_t) \right] \\ &= \frac{1}{1 - \gamma} \mathbb{E}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'}} [A^\pi(s, a)] \\ &= \frac{1}{1 - \gamma} \mathbb{E}_{\substack{s \sim d^{\pi'} \\ a \sim \pi}} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^\pi(s, a) \right] \end{aligned}$$

$d^{\pi'}$
 $a^{\pi'}$

...almost there! Only problem is $s \sim d^{\pi'}$.

$s \sim d^{\pi'}$ unknown

$$d^{\pi'} \approx d^{\pi}$$

What if we just said $d^{\pi'} \approx d^{\pi}$ and didn't worry about it?

$$J(\pi') - J(\pi) \approx \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^{\pi}, a \sim \pi} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^{\pi}(s, a) \right]$$

$$\text{define } \rightarrow \doteq \underline{\mathcal{L}_{\pi}(\pi')}$$

Turns out: this approximation is pretty good when π' and π are close! But why, and how close do they have to be?

Relative policy performance bounds:¹

if approx
is close to
constraint

$$|J(\pi') - (J(\pi) + \mathcal{L}_{\pi}(\pi'))| \leq C \sqrt{\mathbb{E}_{s \sim d^{\pi}} [D_{KL}(\pi' || \pi)[s]]} \quad (3)$$

If policies are close in KL-divergence—the approximation is good!

¹Achiam, Held, Tamar, Abbeel, 2017

What is KL-divergence?

For probability distributions P and Q over a discrete random variable,

$$D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

(Handwritten notes: "def." above the sum, and an arrow pointing to the x in the denominator)

Properties:

- $D_{KL}(P||P) = 0$
- $D_{KL}(P||Q) \geq 0$
- $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ — Non-symmetric!

What is KL-divergence between policies?

$$D_{KL}(\pi' || \pi)[s] = \sum_{a \in \mathcal{A}} \pi'(a|s) \log \frac{\pi'(a|s)}{\pi(a|s)}$$

What did we gain from making that approximation?

$$J(\pi') - J(\pi) \approx \mathcal{L}_\pi(\pi')$$

$$\begin{aligned}\mathcal{L}_\pi(\pi') &= \frac{1}{1 - \gamma} \mathbb{E}_{\substack{s \sim d^\pi \\ a \sim \pi}} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^\pi(s, a) \right] \\ &= \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)} A^\pi(s_t, a_t) \right]\end{aligned}$$

- This is something we can optimize using trajectories sampled from the old policy π !
- Similar to using importance sampling, but because weights only depend on current timestep (and not preceding history), they don't vanish or explode.

- “Approximately Optimal Approximate Reinforcement Learning,” Kakade and Langford, 2002 ²
- “Trust Region Policy Optimization,” Schulman et al. 2015 ³
- “Constrained Policy Optimization,” Achiam et al. 2017 ⁴

²<https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/KakadeLangford-icml2002.pdf>

³<https://arxiv.org/pdf/1502.05477.pdf>

⁴<https://arxiv.org/pdf/1705.10528.pdf>

Monotonic Improvement Theory

From the bound on the previous slide, we get

$$J(\pi') - J(\pi) \geq \mathcal{L}_\pi(\pi') - C \sqrt{\mathbb{E}_{s \sim d^\pi} [D_{KL}(\pi' || \pi)[s]]}.$$

- If we maximize the RHS with respect to π' , we are **guaranteed to improve over π** .
 - This is a *majorize-maximize* algorithm w.r.t. the true objective, the LHS.
- **And** $\mathcal{L}_\pi(\pi')$ and the KL-divergence term *can both be estimated with samples from π* !

need samples only
from π

Monotonic Improvement Theory

Proof of improvement guarantee: Suppose π_{k+1} and π_k are related by

$$\pi_{k+1} = \arg \max_{\pi'} \underbrace{\mathcal{L}_{\pi_k}(\pi')}_{(1)} - C \underbrace{\sqrt{\mathbb{E}_{s \sim d^{\pi_k}} [D_{KL}(\pi' || \pi_k)[s]]}}_{(2)}.$$

π_k is feasible

$$\mathcal{L}_{\pi_k}(\pi_k) \propto \mathbb{E}_{s, a \sim d^{\pi_k}, \pi_k}$$

$$A^{\pi_k}(s, a) = 0$$

$$D_{KL}(\pi_k / \pi_k) = 0$$

$$(1) = 0$$

$$(2) = 0$$

$$(1) - (2) \geq 0$$

$$(1) - (2) > 0$$

$$J(\pi_{k+1}) - J(\pi_k) \geq \mathcal{L}_{\pi_k}(\pi_{k+1}) - C \sqrt{\mathbb{E} \dots} > 0$$

Proof of improvement guarantee: Suppose π_{k+1} and π_k are related by

$$\pi_{k+1} = \arg \max_{\pi'} \mathcal{L}_{\pi_k}(\pi') - C \sqrt{\mathbb{E}_{s \sim d^{\pi_k}} [D_{KL}(\pi' || \pi_k)[s]]}.$$

- π_k is a feasible point, and the objective at π_k is equal to 0.
 - $\mathcal{L}_{\pi_k}(\pi_k) \propto \mathbb{E}_{s, a \sim d^{\pi_k}, \pi_k} [A^{\pi_k}(s, a)] = 0$
 - $D_{KL}(\pi_k || \pi_k)[s] = 0$
- \implies optimal value ≥ 0
- \implies by the performance bound, $J(\pi_{k+1}) - J(\pi_k) \geq 0$

This proof works even if we restrict the domain of optimization to an arbitrary class of parametrized policies Π_θ , as long as $\pi_k \in \Pi_\theta$.

$$\pi_{k+1} = \arg \max_{\pi'} \mathcal{L}_{\pi_k}(\pi') - C \sqrt{\mathbb{E}_{s \sim d^{\pi_k}} [D_{KL}(\pi' || \pi_k)[s]]}. \quad (4)$$

approx error bound by using d^{π_k}

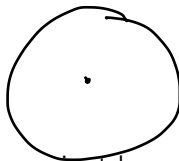
Problem:

- C provided by theory is quite high when γ is near 1
- \implies steps from (4) are too small.

Potential Solution:

- Tune the KL penalty
- Use KL constraint (called **trust region**).

Algorithms



Proximal Policy Optimization (PPO) is a family of methods that approximately penalize policies from changing too much between steps. Two variants:

- Adaptive KL Penalty
 - Policy update solves unconstrained optimization problem

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k)$$

- Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint

Algorithm 1 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0 , initial KL penalty β_0 , target KL-divergence δ

for $k = 0, 1, 2, \dots$ **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

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k)$$

by taking K steps of minibatch SGD (via Adam)

if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \geq 1.5\delta$ **then**

$\beta_{k+1} = 2\beta_k$

else if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \leq \delta/1.5$ **then**

$\beta_{k+1} = \beta_k/2$

end if

end for

- Initial KL penalty not that important—it adapts quickly
- Some iterations may violate KL constraint, but most don't

Proximal Policy Optimization (PPO) is a family of methods that approximately enforce KL constraint **without computing natural gradients**. Two variants:

- Adaptive KL Penalty
 - Policy update solves unconstrained optimization problem

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k)$$

- Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint
- Clipped Objective
 - New objective function: let $r_t(\theta) = \pi_{\theta}(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

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

where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)

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

Algorithm 2 PPO with Clipped Objective

Input: initial policy parameters θ_0 , clipping threshold ϵ

for $k = 0, 1, 2, \dots$ **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

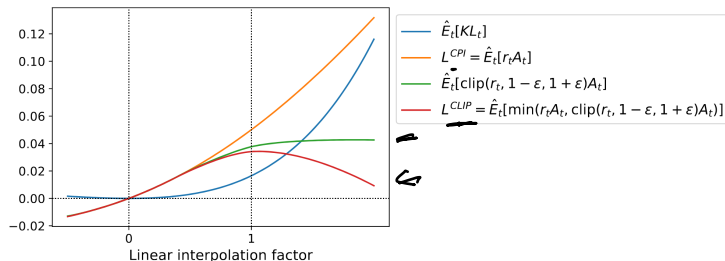
$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathbb{E}_{\tau \sim \pi_k} \left[\sum_{t=0}^T \left[\min(r_t(\theta) \hat{A}_t^{\pi_k}, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t^{\pi_k}) \right] \right]$$

end for

- Clipping prevents policy from having incentive to go far away from θ_{k+1}
- Clipping seems to work at least as well as PPO with KL penalty, but is simpler to implement

Proximal Policy Optimization with Clipped Objective

But *how* does clipping keep policy close? By making objective as pessimistic as possible about performance far away from θ_k :



⁵Schulman, Wolski, Dhariwal, Radford, Klimov, 2017

Empirical Performance of PPO

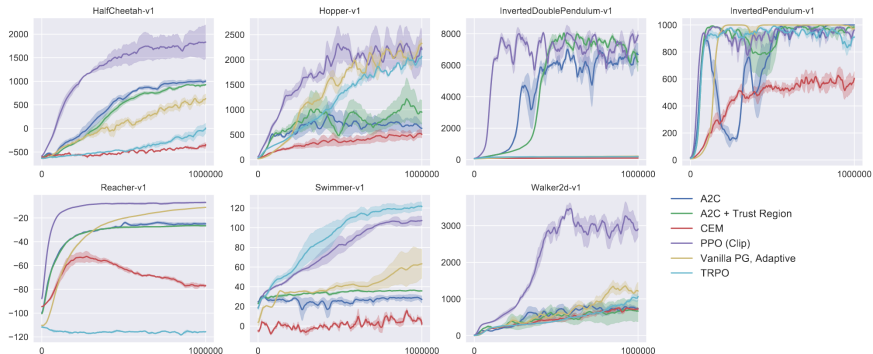


Figure: Performance comparison between PPO with clipped objective and various other deep RL methods on a slate of MuJoCo tasks. ⁶

- Wildly popular, and key component of ChatGPT

⁶Schulman, Wolski, Dhariwal, Radford, Klimov, 2017

PPO

- “Proximal Policy Optimization Algorithms,” Schulman et al. 2017 ⁷
- OpenAI blog post on PPO, 2017 ⁸

⁷<https://arxiv.org/pdf/1707.06347.pdf>

⁸<https://blog.openai.com/openai-baselines-ppo/>

- Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation Matters in Deep RL: A Case Study on PPO and TRPO. ICLR 2020
<https://openreview.net/forum?id=r1etN1rtPB>
- Reward scaling, learning rate annealing, etc. can make a significant difference

Theory:

- ① Problems with Policy Gradient Methods
- ② Policy Performance Bounds
- ③ Monotonic Improvement Theory

Proximal Policy Optimization:

- ① Approximately constraints policy steps
- ② Relatively simple to implement
- ③ Good empirical success and very widely used