

Lecture 6: Policy Gradient II. Advanced policy gradient section slides from Joshua Achiam (OpenAI)'s slides, with minor modifications

Emma Brunskill

CS234 Reinforcement Learning.

Spring 2024

- 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 T
- 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:

- 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

- Last time: Policy Search
- This time: Policy search continued.

- Likelihood ratio / score function policy gradient
 - Baseline
 - Alternative targets
- Advanced policy gradient methods
 - Proximal policy optimization (PPO) (will implement in homework)

↙ policy parameter

$$\nabla_{\theta} V(\theta) \approx \underbrace{(1/m) \sum_{i=1}^m R(\tau^{(i)})}_{\text{Monte Carlo returns}} \underbrace{\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})}_{\text{score function}}$$

- Unbiased but very noisy
- Fixes that can make it practical
 - Temporal structure
 - **Baseline**
 - Alternatives to using Monte Carlo returns $R(\tau^{(i)})$ as targets

- Goal: Converge as quickly as possible to a local optima
 - To obtain data that use to learn, have to make actual decisions which may be suboptimal
 - Aim: minimize number of iterations / time steps until reach a good policy

Policy Gradient Algorithms and Reducing Variance

- 1 Policy Gradient Algorithms and Reducing Variance
 - Baseline
 - Alternatives to MC Returns

- Reduce variance by introducing a *baseline* $b(s)$

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

not w.r.t θ
 a func of a

- For any choice of b , gradient estimator is unbiased.
- Near optimal choice is the expected return,

$$b(s_t) \approx \mathbb{E}[r_t + r_{t+1} + \dots + r_{T-1}]$$

- Interpretation: increase logprob of action a_t proportionally to how much returns $\sum_{t'=t}^{T-1} r_{t'}$ are better than expected

Baseline $b(s)$ Does Not Introduce Bias-Derivation

✓ goal is to show = 0

$\Upsilon: s_0:T \quad a_0:T-1$

$$\mathbb{E}_{\Upsilon} [\nabla_{\theta} \log \pi(a_t | s_t; \theta) b(s_t)]$$

$$= \mathbb{E}_{s_0:t, a_0:(t-1)} [\mathbb{E}_{s_{t+1}:T, a_{t:(T-1)}} [\nabla_{\theta} \log \pi(a_t | s_t; \theta) b(s_t)]]$$

$$= \mathbb{E}_{s_0:t, a_0:t-1} [b(s_t) \mathbb{E}_{s_{t+1}:T, a_{t:T-1}} \nabla_{\theta} \log \pi(a_t | s_t, \theta)]$$

$$= \mathbb{E}_{s_0:t, a_0:t-1} [b(s_t) \mathbb{E}_{a_t} \nabla_{\theta} \log \pi(a_t | s_t, \theta)]$$

$$= \mathbb{E}_{s_0:t, a_0:t-1} [b(s_t) \sum_a \pi(a_t | s_t, \theta) \nabla_{\theta} \log \pi(a_t | s_t, \theta)]$$

$$= \mathbb{E}_{s_0:t, a_0:t-1} [b(s_t) \sum_a \pi(a_t | s_t, \theta) \frac{\nabla_{\theta} \pi(a_t | s_t, \theta)}{\pi(a_t | s_t, \theta)}]$$

$$= \mathbb{E}_{s_0:t, a_0:t-1} [b(s_t) \sum_a \nabla_{\theta} \pi(a_t | s_t, \theta)]$$

$$= \mathbb{E}_{s_0:t, a_0:t-1} [b(s_t) \nabla_{\theta} \underbrace{\sum_a \pi(a_t | s_t, \theta)}_{=1}]$$

$$= \mathbb{E}_{s_0:t, a_0:t-1} [b(s_t) \nabla_{\theta} 1]$$

$$= 0$$

Baseline $b(s)$ Does Not Introduce Bias–Derivation

$$\begin{aligned} & \mathbb{E}_\tau [\nabla_\theta \log \pi(a_t | s_t; \theta) b(s_t)] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[\mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}} [\nabla_\theta \log \pi(a_t | s_t; \theta) b(s_t)] \right] \text{ (break up expectation)} \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}} [\nabla_\theta \log \pi(a_t | s_t; \theta)] \right] \text{ (pull baseline term out)} \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [b(s_t) \mathbb{E}_{a_t} [\nabla_\theta \log \pi(a_t | s_t; \theta)]] \text{ (remove irrelevant variables)} \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \sum_a \pi_\theta(a_t | s_t) \frac{\nabla_\theta \pi(a_t | s_t; \theta)}{\pi_\theta(a_t | s_t)} \right] \text{ (likelihood ratio)} \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \sum_a \nabla_\theta \pi(a_t | s_t; \theta) \right] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \nabla_\theta \sum_a \pi(a_t | s_t; \theta) \right] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [b(s_t) \nabla_\theta 1] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [b(s_t) \cdot 0] = 0 \end{aligned}$$

"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

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

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

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

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

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

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

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

endfor

Initialize policy parameter θ , baseline b

for iteration=1, 2, ... **do**

Collect a set of trajectories by executing the current policy

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

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

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

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

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

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

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

endfor

- Recall Q-function / state-action-value function:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[r_0 + \gamma r_1 + \gamma^2 r_2 \cdots \mid s_0 = s, a_0 = a \right]$$

- State-value function can serve as a great baseline

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi \left[r_0 + \gamma r_1 + \gamma^2 r_2 \cdots \mid s_0 = s \right] \\ &= \mathbb{E}_{a \sim \pi} [Q^\pi(s, a)] \end{aligned}$$

(for new ~~assum~~
episodic so
you want
 $\gamma=1$)

Policy Gradient Algorithms and Reducing Variance

- Baseline

2 Policy Gradient Algorithms and Reducing Variance

- Alternatives to MC Returns

- Policy gradient:

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

- 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 we saw for TD vs MC, and value function approximation

- 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

θ V/Q
actor critic

- Recall:

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

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

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

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

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

$$r + \gamma V(s')$$

$$\nabla_{\theta} V(\theta) \approx (1/m) \sum_{i=1}^m \sum_{t=0}^{T-1} R_t^i \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})$$

$\tau_{\theta}(c)$

- Note that critic can select any blend between TD and MC estimators for the target to substitute for the true state-action value function.

$$\nabla_{\theta} V(\theta) \approx (1/m) \sum_{i=1}^m \sum_{t=0}^{T-1} R_t^i \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})$$

- Note that critic can select any blend between TD and MC estimators for the target to substitute for the true state-action value function.

$$\hat{R}_t^{(1)} = r_t + \gamma V(s_{t+1})$$


$$\hat{R}_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \quad \dots$$

$$\hat{R}_t^{(\text{inf})} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

- If subtract baselines from the above, get advantage estimators

$$\hat{A}_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$\hat{A}_t^{(\text{inf})} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots - V(s_t)$$

MSE 

$$\nabla_{\theta} V(\theta) \approx (1/m) \sum_{i=1}^m \sum_{t=0}^{T-1} R_t^i \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})$$

- If subtract baselines from the above, get advantage estimators

$$\hat{A}_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t) \quad \text{estimator} \quad \text{low var, high bias}$$

$$\hat{A}_t^{(\infty)} = \underline{r}_t + \gamma \underline{r}_{t+1} + \gamma^2 \underline{r}_{t+2} + \dots - V(s_t) \quad \text{high var, low or 0 bias}$$

- Select all that are true
- $\hat{A}_t^{(1)}$ has low variance & low bias.
- $\hat{A}_t^{(1)}$ has high variance & low bias.
- $\hat{A}_t^{(\infty)}$ low variance and high bias.
- $\hat{A}_t^{(\infty)}$ high variance and low bias.
- Not sure



$$\nabla_{\theta} V(\theta) \approx (1/m) \sum_{i=1}^m \sum_{t=0}^{T-1} R_t^i \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})$$

- If subtract baselines from the above, get advantage estimators

$$\hat{A}_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$\hat{A}_t^{(\text{inf})} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \dots - V(s_t)$$

Solution: $\hat{A}_t^{(1)}$ has low variance & high bias. $\hat{A}_t^{(\infty)}$ high variance but low bias.

- 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

- 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:

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

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

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

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

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

Advanced Policy Gradients

Theory:

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

Algorithms:

- 1 Proximal Policy Optimization

The Problems with Policy Gradients

Policy gradient algorithms try to solve the optimization problem

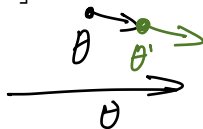
$$\max_{\theta} J(\pi_{\theta}) \doteq \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

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



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

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

- 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
 - Collect sample trajectories from policy, then form sample estimate. (More stable)
 - Use trajectories from other policies (Less stable)
- Opportunity: use old data to take **multiple gradient steps** before using the resulting new policy to gather more data
- Challenge: even if this is possible to use old data to estimate multiple gradients, how many steps should be taken?

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?

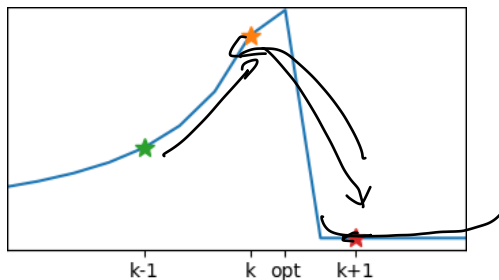


Figure: Policy parameters on x-axis and performance on y-axis. A bad step can lead to performance collapse, which may be hard to recover from.

The Problem is More Than Step Size

Consider a family of policies with parametrization:

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

logistic function $\frac{1}{1+e^{-\theta}}$

$\theta \rightarrow \pi(a|\theta)$

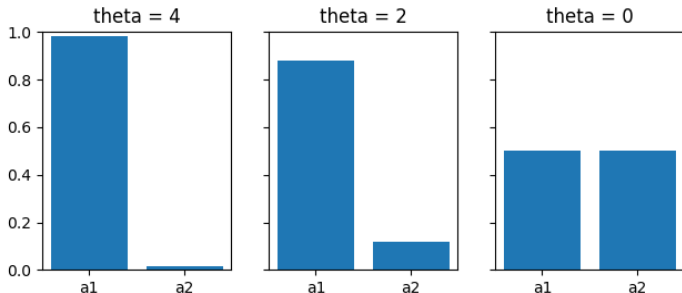


Figure: Small changes in the policy parameters can unexpectedly lead to **big** changes in the policy.

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 we ask you to prove that for any policies π, π'

$$\begin{aligned}
 J(\pi') - J(\pi) &= \mathbb{E}_{\pi \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi'}(s_t, a_t) \right] && \text{value of policy } \pi' \\
 &= \frac{1}{1-\gamma} \mathbb{E}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'}} [A^{\pi'}(s, a)] && \text{if fruit counter } \gamma = 1 \\
 & && Q^{\pi'}(s_t, a_t) - V^{\pi'}(s_t) \quad (1) \\
 & && Q^{\pi'}(s_t, \pi'_t(a_t)) \quad (2) \\
 & && \text{state action distri. } \zeta
 \end{aligned}$$

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?

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

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' ...

Goal: estimate $J(\pi')$ only from data from π

Looking at it from another angle...

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

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

we can rewrite the relative policy performance identity:

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

from (traj) not from

$$\begin{aligned} &= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi'}, a \sim \pi'} A^\pi(s, a) \\ &= \frac{1}{1-\gamma} \mathbb{E}_s \sum_a \pi'(a|s) A^\pi(s, a) \\ &= \frac{1}{1-\gamma} \mathbb{E}_s \sum_a \frac{\pi'(a|s)}{\pi(a|s)} \pi(a|s) A^\pi(s, a) \\ &= \frac{1}{1-\gamma} \mathbb{E}_s \sum_a \mathbb{E}_{a \sim \pi} \left(A^\pi(s, a) \frac{\pi'(a|s)}{\pi(a|s)} \right) \end{aligned}$$

Note: Instance of Importance Sampling

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

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

we can rewrite the relative policy performance identity:

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

Last step is an instance of **importance sampling** (more on this next time)

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

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

we can rewrite the relative policy performance identity:

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

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

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

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

$$\doteq \mathcal{L}_{\pi}(\pi')$$



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

constant

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

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,

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

$\pi'(a|s)$ $\pi(a|s)$

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.

We will talk about next week

- “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>

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

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k) \quad (4)$$

$$\bar{D}_{KL}(\theta || \theta_k) = E_{s \sim d^{\pi_k}} D_{KL}(\theta_k(\cdot | s), \pi_{\theta}(\cdot | s)) \quad (5)$$

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

Algorithm 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

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

L6N3 Check Your Understanding: Proximal Policy Optimization

- Clipped Objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

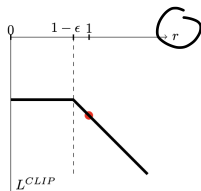
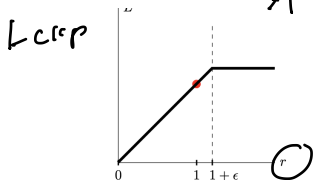
$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathbb{E}_{\tau \sim \pi_k} \left[\sum_{t=0}^T \left[\min(r_t(\theta) \hat{A}_t^{\pi_k}, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t^{\pi_k}) \right] \right]$$

- where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)
- Policy update is $\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$.

consider for $A > 0$
 $A = advantage$

Consider the figure⁵. Select all that are true. $\epsilon \in (0, 1)$.

- The left graph shows the L^{CLIP} objective when the advantage function $A > 0$ and the right graph shows when $A < 0$
- The right graph shows the L^{CLIP} objective when the advantage function $A > 0$ and the left graph shows when $A < 0$
- It depends on the value of ϵ ✗
- Not sure



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

L6N3: Check Your Understanding L6N2 Proximal Policy Optimization Solutions

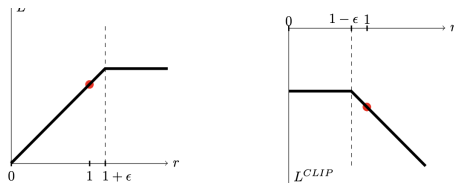
- Clipped Objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

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

- where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)
- Policy update is $\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$.

Consider the figure⁶. Select all that are true. $\epsilon \in (0, 1)$.

The left graph shows the L^{CLIP} objective when the advantage function $A > 0$ and the right graph shows when $A < 0$



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

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

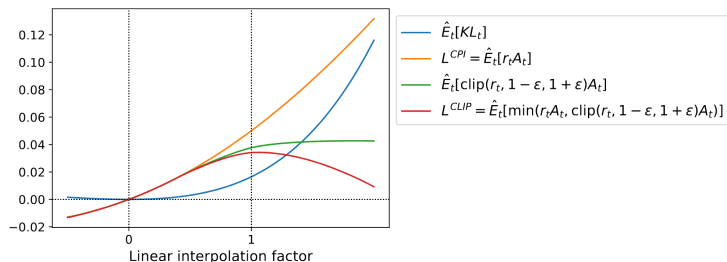


Figure: Various objectives as a function of interpolation factor α between θ_{k+1} and θ_k after one update of PPO-Clip ⁷

⁷Schulman, Wolski, Dhariwal, Radford, Klimov, 2017

Algorithm 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

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

next time will discuss

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

Empirical Performance of PPO

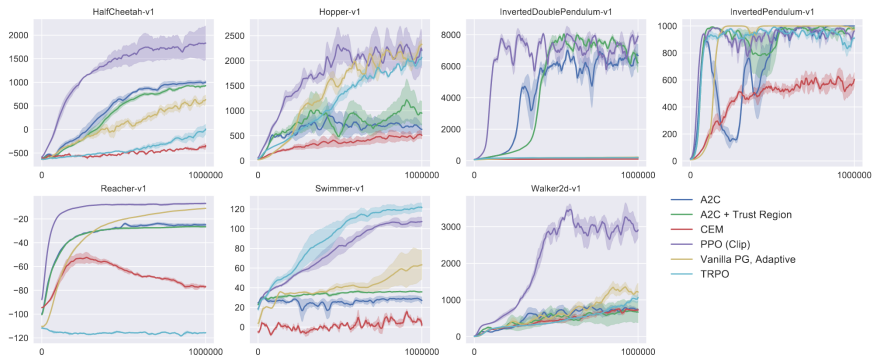


Figure: Performance comparison between PPO with clipped objective and various other deep RL methods on a slate of MuJoCo tasks. ⁸

- Wildly popular, and key component of ChatGPT

⁸Schulman, Wolski, Dhariwal, Radford, Klimov, 2017

PPO

- “Proximal Policy Optimization Algorithms,” Schulman et al. 2017 ⁹
- OpenAI blog post on PPO, 2017 ¹⁰

⁹<https://arxiv.org/pdf/1707.06347.pdf>

¹⁰<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

- Likelihood ratio / score function policy gradient
 - Baseline
 - Alternative targets
- Advanced policy gradient methods
 - Proximal policy optimization (PPO) algorithm (will implement in homework)

- Last time: Policy Search
- This time: Policy search continued.
- Next time: Proximal Policy Optimization (PPO) cont (theory and additional discussion)