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

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2026

Refresh Your Knowledge L6N1

- 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 no
  5. Not sure

π_θ

- 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

(why 2 is false)

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

Class Structure

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

Today

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

Class Structure

- Last time: time: DQN and REINFORCE
- This time: Policy Gradient and PPO
- Next time: Policy Search Cont.

Policy Gradient Algorithms and Reducing Variance

Table of Contents

recall π parameterized by θ

$\arg\max_\theta V^{\pi_\theta}$

- Leverages likelihood ratio / score function and temporal structure

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$$

REINFORCE:

Initialize policy parameters θ arbitrarily
for each episode $\{s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
  for $t = 1$ to $T - 1$ **do**
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$
  endfor
endfor
return θ

Likelihood Ratio / Score Function Policy Gradient

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

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

Policy Gradient: Introduce Baseline

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

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

$b(s_t)$

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

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

- 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

to prove that ↓ $= 0$

$\tau = s_{0:t}, a_{0:t-1}, a_{t:T-1}, s_{t+1:T}$

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

$$= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[\mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}} [\nabla_\theta \log \pi(a_t|s_t;\theta) b(s_t)] \right]$$

break up expectation

$= E_{s_{0:t}, a_{0:t-1}} [b(s_t) E_{s_{t+1:T}, a_{t:T-1}} \nabla_\theta \log \pi(a_t|s_t,\theta)]$ pull b out

$= E$ " $[b(s_t) E_{a_t} \nabla_\theta \log \pi(a_t|s_t,\theta)]$ ~~$E_{s_{t+1 \ldots T}, a_{t+1:T}}$~~ $= 1$

$=$ " $[b(s_t) \sum_a \pi(a_t|s_t,\theta) \frac{\nabla_\theta \pi(a_t|s_t,\theta)}{\pi(a_t|s_t,\theta)}]$

$=$ " $[b(s_t) \sum_a \nabla_\theta \pi(a_t|s_t,\theta)]$

$=$ " $[b(s_t) \nabla_\theta \sum_a \pi(a_t|s_t,\theta)]$

$= 1$

$= 0$

$= 0$

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

$$
\begin{aligned}
&\mathbb{E}_\tau[\nabla_\theta \log \pi(a_t|s_t;\theta) b(s_t)] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[\mathbb{E}_{s_{(t+1):T},a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t;\theta) b(s_t)]\right] \text{ (break up expectation)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\mathbb{E}_{s_{(t+1):T},a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t;\theta)]\right] \text{ (pull baseline term out)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}[b(s_t)\mathbb{E}_{a_t}[\nabla_\theta \log \pi(a_t|s_t;\theta)]] \text{ (remove irrelevant variables)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\sum_a \pi_\theta(a_t|s_t)\frac{\nabla_\theta \pi(a_t|s_t;\theta)}{\pi_\theta(a_t|s_t)}\right] \text{ (likelihood ratio)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\sum_a \nabla_\theta \pi(a_t|s_t;\theta)\right] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\nabla_\theta \sum_a \pi(a_t|s_t;\theta)\right] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}[b(s_t)\nabla_\theta 1] \\
&= \mathbb{E}_{s_0:t,a_0:(t-1)}[b(s_t)\cdot 0] = 0
\end{aligned}
$$

Argument for Why Baseline $b(s)$ Can Reduce Variance

$G(s_t) = \sum_t^T r_t$

- Motivation was for introducing baseline $b(s)$ was to reduce variance

$$Var[\nabla_\theta \mathbb{E}_\tau[R]] \quad = Var\left[\mathbb{E}_\tau\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta)\,(R_t(s_t) - b(s_t))\right]\right] \; (*) \qquad (1)$$

$Var(\nabla_\theta \log \pi(a_t;\theta)(G(s_t) - b(s_t)))$

$Var(X) = E[X^2] - E[X]^2$ baseline does not change $E[X]$

$\Rightarrow \arg\min_b$ ~~Var~~ $* = \arg\min_b E[E_\tau[\ldots]^2]$

$\arg\min_b E[(\nabla_\theta \log \pi(a|s))^2 (G_t(s_t) - b(s_t))^2]$

$\downarrow$

$\approx V^\pi(s_t)$

Argument for Why Baseline $b(s)$ Can Reduce Variance

- Motivation was for introducing baseline $b(s)$ was to reduce variance

$$Var[\nabla_\theta \mathbb{E}_\tau[R]] = Var\left[\mathbb{E}_\tau\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta)\,(R_t(s_t) - b(s_t))\right]\right] \quad (2)$$

$$\approx \sum_{t=0}^{T-1} \mathbb{E}_\tau\, Var\left[\left[\nabla_\theta \log \pi(a_t|s_t;\theta)\,(R_t(s_t) - b(s_t))\right]\right] \quad (3)$$

- Focus on the variance of one term.

$$Var\left[\left[\nabla_\theta \log \pi(a_t|s_t;\theta)\,(R_t(s_t) - b(s_t))\right]\right] = E\left[\left[\nabla_\theta \log \pi(a_t|s_t;\theta)\,(R_t(s_t) - b(s_t))\right]^2\right] - \left[E\left[\nabla_\theta \log \pi(a_t|s_t;\theta)\,(R_t(s_t) - b(s_t))\right]\right]^2$$

- Choosing a baseline to minimize variance
- Recall the baseline $b(s)$ does not impact the expectation. Therefore sufficient to consider

$$\arg\max_b Var\left[\left[\nabla_\theta \log \pi(a_t|s_t;\theta)\,(G_t(s_t) - b(s_t))\right]\right] = \arg\min_b E\left[\left[(\nabla_\theta \log \pi(a_t|s_t;\theta))^2\,(G_t(s_t) - b(s_t))^2\right]\right] \quad (4)$$

$$= \arg\min_b E_{s\sim d^\pi}\left[E_{a\sim\pi(\cdot|s),G|s,a}\left[(\nabla_\theta \log \pi(a_t|s;\theta))^2\,(G_t(s) - b(s))^2\right]\right]$$

- This is a weighted least squares problem. Taking the derivative and setting to zero yields

$$b(s) = = \frac{E_{a\sim\pi(\cdot|s),G|s,a}\left[(\nabla_\theta \log \pi(a_t|s;\theta))^2 G_t(s)\right]}{E_{a\sim\pi(\cdot|s),G|s,a}(\nabla_\theta \log \pi(a_t|s;\theta))^2} \approx E_{a\sim\pi(\cdot|s),G|s,a}[G_t(s)] \quad (5)$$

$= V^\pi(s_t)$

"Vanilla" Policy Gradient Algorithm

Advantga $A^{\pi}(s) = Q^{\pi}(s,a) - V^{\pi}(s)$

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **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

Other Choices for Baseline?

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **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

Choosing the Baseline: Value Functions

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

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

- State-value function can serve as a great baseline

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

Table of Contents

Likelihood Ratio / Score Function Policy Gradient

- 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?**

Choosing the Target

- 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

Actor-critic Methods

- 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

Policy Gradient Formulas with Value Functions

- Recall:

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

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

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

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

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

Advanced Policy Gradients

Outline

Theory:

1. Problems with Policy Gradient Methods
2. Policy Performance Bounds
3. Monotonic Improvement Theory (next time)

Algorithms:

1. Proximal Policy Optimization

The Problems with Policy Gradients

Policy Gradients Review

Policy gradient algorithms try to solve the optimization problem

$J(\pi_\theta) = V^{\pi_\theta}$

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

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

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

if $|\theta - \theta'|_\infty < \epsilon$

then $|V^\theta - V^{\theta'}|_\infty < \epsilon$?

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 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
  - Collect sample trajectories from policy, then form sample estimate. (More stable)
  - Use trajectories from other policies (Less stable) → distrib mismatch
- 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?

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?)

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?

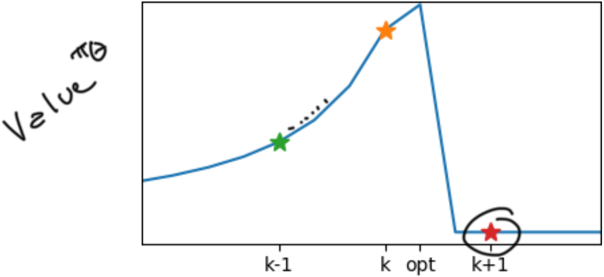


Figure: Policy parameters on x-axis and performance on y-axis. A bad step can lead to performance collapse, which may be hard to recover from.

The Problem is More Than Step Size

Consider a family of policies with parametrization:

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

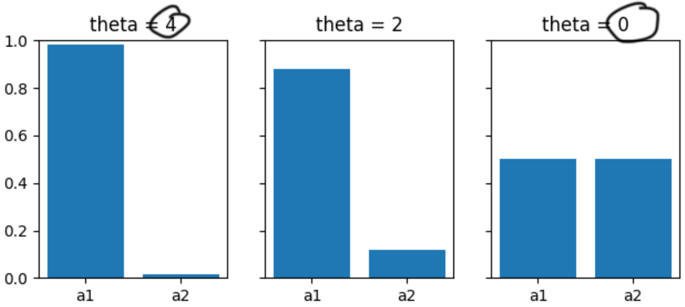


Figure: Small changes in the policy parameters can unexpectedly lead to **big** changes in the policy.

> 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

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. $p(a|\pi_\theta, s)$ θ

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'

$Q^\pi(s_t, a_t) - V^\pi(s_t)$

$$
\begin{aligned}
J(\pi') - J(\pi) &= \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^\pi(s_t, a_t) \right] && (6) \\
&= \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'}} \left[A^\pi(s, a) \right] && (7)
\end{aligned}
$$

where

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

distrib over states

What is it good for?

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

$J(\pi')$

$$\max_{\pi'} J(\pi') = \max_{\pi'} J(\pi') - J(\pi)$$

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

$J(\pi)$

$Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$

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'...

but we don't have π'! that is the new π we are trying to step to

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:

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

$$= \frac{1}{1-\gamma} E_{s\sim d^{\pi'},\ a\sim\pi'} A^\pi(s,a)$$

$$= \frac{1}{1-\gamma} E_{s\sim d^{\pi'}} \sum_a \pi'(a|s) A^\pi(s,a)$$

$$= \quad " \quad " \quad \sum_a \pi'(a|s) \cdot \frac{\pi(a|s)}{\pi(a|s)} A^\pi(s,a)$$

$$= \quad " \quad " \quad E_{a\sim\pi}\left[\frac{\pi'(a|s)}{\pi(a|s)} A^\pi(s,a)\right]$$

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:

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

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

Problem: State Distribution

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:

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

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

A Useful Approximation

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

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

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

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

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]

$$\left| J(\pi') - \left(J(\pi) + \mathcal{L}_{\pi}(\pi') \right) \right| \leq C \sqrt{\mathop{\mathrm{E}}_{s \sim d^{\pi}} \left[D_{KL}(\pi' || \pi)[s] \right]} \tag{8}$$

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

not θ

[1] Achiam, Held, Tamar, Abbeel, 2017

What is KL-divergence?

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

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

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?

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

A Useful Approximation

What did we gain from making that approximation?

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

$$\mathcal{L}_\pi(\pi') = \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s\sim d^\pi \\ a\sim\pi}} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^\pi(s,a) \right]$$

$$= \mathop{\mathrm{E}}_{\tau\sim\pi} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)} A^\pi(s_t, a_t) \right]$$

- 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.

Recommended Reading

- "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

striving for monotonic policy improvement

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

in policy space

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k) \quad (9)$$

$$\bar{D}_{KL}(\theta || \theta_k) = E_{s \sim d^{\pi_k}} D_{KL}(\theta_k(\cdot|s), \pi_\theta(\cdot|s)) \quad (10)$$

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

$$\mathcal{L}_{\pi}(\pi') = E_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)} A^{\pi}(s_t, a_t)\right]$$

$$\approx V^{\pi'} - V^{\pi}$$

Proximal Policy Optimization with Adaptive KL Penalty

Algorithm PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, ...$ **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

PPO with Adaptive KL Penalty: **Multiple Gradient Steps**

Algorithm PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, ...$ **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

importance ratio

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

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

$$\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}^{\pi_t}$$

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

L6 Check Your Understanding: Proximal Policy Optimization

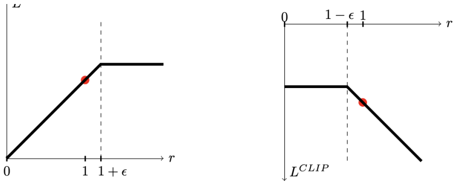
- Clipped Objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

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

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

Consider the figure[5]. Select all that are true. $\epsilon \in (0, 1)$.

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



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

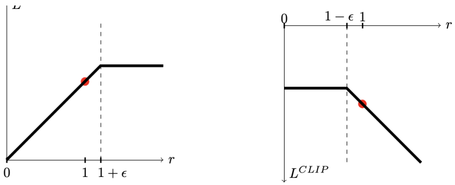
L6: Check Your Understanding Proximal Policy Optimization Solutions

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

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

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

Consider the figure[6]. 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 $A < 0$



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