Actor Critic Methods

CS 224R
Reminders

Since Wednesday: Homework 1 is out
Next Monday: Project survey due
4/19: Homework 1 due, Homework 2 out

Reminder:

• Provide your AWS account ID if you haven’t yet! (see Ed)
The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

Actor-critic

Case studies: robotics & RLHF

Key learning goals:

• Practical policy gradient implementation tricks & case studies

• Understanding a generic actor-critic method
The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

Actor-critic

Case studies: robotics & RLHF
Evaluating the objective

\[ \theta^* = \arg \max_\theta E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]

sum over samples from \( \pi_\theta \)

Slide adapted from Sergey Levine
Policy gradients

\[
\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right)
\]

\[
\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\theta} \log \pi_{\theta}(\tau_{i}) r(\tau_{i})
\]

\[
\sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t})
\]

good stuff is made more likely
bad stuff is made less likely
simply formalizes the notion of “trial and error”!

REINFORCE algorithm:

1. sample \(\{\tau^i\}\) from \(\pi_{\theta}(a_t|s_t)\) (run it on the robot)
2. \(\nabla_{\theta} J(\theta) \approx \sum_{i} \left( \sum_{t} \nabla_{\theta} \log \pi_{\theta}(a^i_t|s^i_t) \right) \left( \sum_{t} r^i_t, a^i_t \right)\)
3. \(\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)\)
The anatomy of a reinforcement learning algorithm

- Generate samples (i.e., run the policy)
- Fit a model to estimate return
  - Compute \( \hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \) (MC policy gradient)
  - Fit \( Q_\phi(s, a) \) (actor-critic, Q-learning)
  - Estimate \( p(s'|s, a) \) (model-based)
- Improve the policy
  - Update \( \theta \) with \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \) (policy gradient)
  - Set \( \pi(s) = \text{arg max } Q_\phi(s, a) \) (Q-learning)
  - Optimize \( \pi_\theta(a|s) \) (model-based)

\[
\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right)
\]
Variance of the gradient estimator

policy gradient: \[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i) r(\tau_i) \]
\[ \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|s_{i,t}) \]
Small way to reduce variance

policy gradient: \[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]
Small way to reduce variance

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]
Small way to reduce variance

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]
Small way to reduce variance

policy gradient: \( \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \)

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right)
\]
Small way to reduce variance

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left( \sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left( \sum_{t' = 1}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'}) \right) \]
Small way to reduce variance

policy gradient: \[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]
Small way to reduce variance

policy gradient: \[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{t'=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]
Small way to reduce variance

Policy gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left( \sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \left( \sum_{t'=1}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'}) \right)$$
Small way to reduce variance

policy gradient:  \( \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \)

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t=1}^{T} r(a_{i,t'}, s_{i,t'}) \right)
\]
Small way to reduce variance

Policy gradient: \[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]
Small way to reduce variance

Policy gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right)$$

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right)$$

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right)$$

Sum together these rewards.

Not these!
Small way to reduce variance

policy gradient: \( \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \)

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right)
\]

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right)
\]
Small way to reduce variance

policy gradient: \[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \left( \sum_{t'=1}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

\[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”
The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

Actor-critic

Case studies: robotics & RLHF
Improving the policy gradient

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”
Improving the policy gradient

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”

\[ \hat{Q}_{i,t} \]
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \left( \sum_{t' = t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”

\[ \hat{Q}_{i,t} \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]

Slide adapted from Sergey Levine
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”

\[ \hat{Q}_{i,t} \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go” \( \hat{Q}_{i,t} \)

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \)

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left( \sum_{t' = t}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'}) \right) \]

Reward “to go”

\[ \hat{Q}_{i,t} \]

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( \mathbf{a}_{i,t} \) in state \( \mathbf{s}_{i,t} \)

can we get a better estimate?

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \]
Improving the policy gradient

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left( \sum_{t' = t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go” \( \hat{Q}_{i,t} \)

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \)

can we get a better estimate?

\[ J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t} r(s_t, a_t) \right] \approx \frac{1}{N} \sum_{i} \sum_{t} r(s_{i,t}, a_{i,t}) \]
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go” \( \hat{Q}_{i,t} \)

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \)

can we get a better estimate?

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_{t} r(s_{t}, a_{t}) \right] \approx \frac{1}{N} \sum_{i} \sum_{t} r(s_{i,t}, a_{i,t}) \]
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”

\[ \hat{Q}_{i,t} \]

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \)

can we get a better estimate?

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_{t} r(s_{t}, a_{t}) \right] \approx \frac{1}{N} \sum_{i} \sum_{t} r(s_{i,t}, a_{i,t}) \]
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta (a_{i,t} | s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”

\[ \hat{Q}_{i,t} \]

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \)

can we get a better estimate?

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]
Improving the policy gradient

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right)$$

Reward “to go”
$$\hat{Q}_{i,t}$$

$\hat{Q}_{i,t}$: estimate of expected reward if we take action $a_{i,t}$ in state $s_{i,t}$
can we get a better estimate?

$$J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t} r(s_{t}, a_{t}) \right] \approx \frac{1}{N} \sum_{i} \sum_{t} r(s_{i,t}, a_{i,t})$$

Slide adapted from Sergey Levine
Improving the policy gradient

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left( \sum_{t' = t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”

\[ \hat{Q}_{i,t} \]

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \)

can we get a better estimate?

\[ J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t} r(s_{t}, a_{t}) \right] \approx \frac{1}{N} \sum_{i} \sum_{t} r(s_{i,t}, a_{i,t}) \]
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \left( \sum_{t' = t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”
\[ \hat{Q}_{i,t} \]

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \)
can we get a better estimate?

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]
Improving the policy gradient

\[
\n\n\n\n\n\n\text{Reward “to go”}
\]

\[\hat{Q}_{i,t}\]

\[\hat{Q}_{i,t}: \text{estimate of expected reward if we take action } a_{i,t} \text{ in state } s_{i,t}\]

can we get a better estimate?

Slide adapted from Sergey Levine
Improving the policy gradient

\[ \nabla \theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla \theta \log \pi_\theta (a_{i,t} | s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go”

\[ \hat{Q}_{i,t} \]

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \).

can we get a better estimate?

\[ Q(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta} [r(s_{t'}, a_{t'}) | s_t, a_t] \]: true expected reward-to-go

Slide adapted from Sergey Levine
Improving the policy gradient

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right) \]

Reward “to go” \( \hat{Q}_{i,t} \)

\( \hat{Q}_{i,t} \): estimate of expected reward if we take action \( a_{i,t} \) in state \( s_{i,t} \) can we get a better estimate?

\[ Q(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] \]: true expected reward-to-go

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) Q(s_{i,t}, a_{i,t}) \]
State & state-action value functions

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] \]: total reward from taking \( a_t \) in \( s_t \)

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)] \]: total reward from \( s_t \)

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]: how much better \( a_t \) is
Value-Based RL

Value function: $V^\pi(s_t) = \ ?$

Q function: $Q^\pi(s_t, a_t) = \ ?$

Advantage function: $A^\pi(s_t, a_t) = \ ?$

Reward = 1 if I can play it in a month, 0 otherwise

Current $\pi(a_1|s) = 1$
State & state-action value functions

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] \]: total reward from taking \( a_t \) in \( s_t \)

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)] \]: total reward from \( s_t \)

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]: how much better \( a_t \) is

Slide adapted from Sergey Levine
State & state-action value functions

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] \]: total reward from taking \( a_t \) in \( s_t \)

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)] \]: total reward from \( s_t \)

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]: how much better \( a_t \) is

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) Q^\pi(s_{i,t}, a_{i,t}) \]
State & state-action value functions

\[ Q^{\pi}(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] : \text{total reward from taking } a_t \text{ in } s_t \]

\[ V^{\pi}(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^{\pi}(s_t, a_t)] : \text{total reward from } s_t \]

\[ A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) : \text{how much better } a_t \text{ is} \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) Q^{\pi}(s_{i,t}, a_{i,t}) \]
State & state-action value functions

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] : \text{total reward from taking } a_t \text{ in } s_t \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)] : \text{total reward from } s_t \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) : \text{how much better } a_t \text{ is} \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]
State & state-action value functions

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t]: \] total reward from taking \(a_t\) in \(s_t\)

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]: \] total reward from \(s_t\)

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t): \] how much better \(a_t\) is

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_{\theta}(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

Slide adapted from Sergey Levine
State & state-action value functions

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t] : \text{total reward from taking } a_t \text{ in } s_t \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t | s_t)}[Q^\pi(s_t, a_t)] : \text{total reward from } s_t \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) : \text{how much better } a_t \text{ is} \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

fit \( Q^\pi, V^\pi, \) or \( A^\pi \)

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

Slide adapted from Sergey Levine
State & state-action value functions

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{s_{t'}, a_{t'}}[r(s_{t'}, a_{t'})|s_t, a_t]: \text{total reward from taking } a_t \text{ in } s_t \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]: \text{total reward from } s_t \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t): \text{how much better } a_t \text{ is} \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

fit \( Q^\pi, V^\pi, \) or \( A^\pi \)

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

\[ \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \]

Slide adapted from Sergey Levine
Value function fitting

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_{\theta}} [r(s_{t'}, a_{t'})|s_t, a_t] \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_{\theta}(a_t|s_t)}[Q^\pi(s_t, a_t)] \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

Slide adapted from Sergey Levine
Value function fitting

\[
Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta} \left[ r(s_{t'}, a_{t'}) | s_t, a_t \right]
\]

\[
V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t | s_t)} [Q^\pi(s_t, a_t)].
\]

\[
A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)
\]

\[
\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) A^\pi(s_{i,t}, a_{i,t})
\]

fit what to what?
Value function fitting

\[
Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t]
\]

\[
V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)].
\]

\[
A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)
\]

\[
\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t})
\]

fit what to what?

\[Q^\pi, V^\pi, A^\pi?\]
**Value function fitting**

\[
Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta} \left[ r(s_{t'}, a_{t'}) | s_t, a_t \right]
\]

\[
V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)} \left[ Q^\pi(s_t, a_t) \right]
\]

\[
A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)
\]

\[
\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t})
\]

fit what to what?

\[
Q^\pi, V^\pi, A^\pi?
\]

\[
Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta} \left[ r(s_{t'}, a_{t'}) | s_t, a_t \right]
\]

fit \( Q^\pi, V^\pi, \text{ or } A^\pi \)

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

\[
\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)
\]

Slide adapted from Sergey Levine
Value function fitting

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t] \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t | s_t)}[Q^\pi(s_t, a_t)] \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

fit what to what?

\[ Q^\pi, V^\pi, A^\pi? \]

\[ Q^\pi(s_t, a_t) = r(s_t, a_t) + \sum_{t'=t+1}^T E_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t] \]

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

\[ \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \]
Value function fitting

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta} \left[ r(s_{t'}, a_{t'}) | s_t, a_t \right] \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t | s_t)} [Q^\pi(s_t, a_t)] \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

fit what to what?

\[ Q^\pi, V^\pi, A^\pi \]

\[ Q^\pi(s_t, a_t) = r(s_t, a_t) + \sum_{t'=t+1}^{T} E_{\pi_\theta} \left[ r(s_{t'}, a_{t'}) | s_t, a_t \right] \frac{V^\pi(s_{t+1})}{V^\pi(s_t)} \]

\[ \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \]

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

Slide adapted from Sergey Levine
Value function fitting

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)] \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

fit what to what?

\[ Q^\pi, V^\pi, A^\pi? \]

\[ Q^\pi(s_t, a_t) = r(s_t, a_t) + E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[V^\pi(s_{t+1})] \]

Slide adapted from Sergey Levine
**Value function fitting**

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t] \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(\cdot | s_t)}[Q^\pi(s_t, a_t)] \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

**fit what to what?**

\[ Q^\pi, V^\pi, A^\pi? \]

\[ Q^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) \]

---

Slide adapted from Sergey Levine
Value function fitting

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)] \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

fit what to what?

\[ Q^\pi, V^\pi, A^\pi? \]

\[ Q^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) \]

\[ A^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t) \]

fit \( Q^\pi, V^\pi, \) or \( A^\pi \)

generate samples (i.e. run the policy)

improve the policy

\[ \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \]

Slide adapted from Sergey Levine
Value function fitting

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t, a_t] \]
\[ V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)] \]
\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]
\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

fit what to what?
\[ Q^\pi, V^\pi, A^\pi? \]
\[ Q^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) \]
\[ A^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t) \]

let’s just fit \( V^\pi(s) \)!

Slide adapted from Sergey Levine
Value function fitting

\[ Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_{\theta}}[r(s_{t'}, a_{t'})|s_t, a_t] \]

\[ V^\pi(s_t) = E_{a_t \sim \pi_{\theta}(a_t|s_t)}[Q^\pi(s_t, a_t)] \]

\[ A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \]

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) A^\pi(s_{i,t}, a_{i,t}) \]

fit what to what?

\[ Q^\pi, V^\pi, A^\pi? \]

\[ Q^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) \]

\[ A^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t) \]

let’s just fit \( V^\pi(s) \)!
Multi-Step Prediction

$\hat{Q}_{i,t} \approx \left( \sum_{t'=t}^{T} r(a_{i,t'}, s_{i,t'}) \right)$

$Q_{i,t} = \sum_{t'=t}^{T} E_{\pi_0} [r(s_{t'}, a_{t'}) | s_t, a_t]$

- How do you update your predictions about winning the game?
- What happens if you don’t finish the game?
- Do you always wait till the end?

$\hat{Q}_{i,t} \approx r(a_{i,t}, s_{i,t}) + V^{\pi}(s_{t+1})$
How can we use all of this to fit a better estimator?

**Goal:** fit $V_\pi$

ideal target: $y_{i,t} = \sum_{t'=t}^T E_{\pi_\theta} [r(s_{t'}, a_{t'}) | s_{i,t}] \approx r(s_{i,t}, a_{i,t}) + V_\pi(s_{i,t+1}) \approx r(s_{i,t}, a_{i,t}) + \hat{V}_\phi(s_{i,t+1})$

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'})$

directly use previous fitted value function!

training data: $\left\{ \left(s_{i,t}, r(s_{i,t}, a_{i,t}) + \hat{V}_\phi(s_{i,t+1}) \right) \right\}$

supervised regression: $\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi(s_i) - y_i \right\|^2$

sometimes referred to as a “bootstrapped” estimate

Slide adapted from Sergey Levine
Aside: discount factors

\[ y_{i,t} \approx r(s_{i,t}, a_{i,t}) + \hat{V}_\phi(s_{i,t+1}) \]

\[ \mathcal{L}(\phi) = \frac{1}{2} \sum_i \| \hat{V}_\phi(s_i) - y_i \|^2 \]

what if \( T \) (episode length) is \( \infty \)?
\( \hat{V}_\phi \) can get infinitely large in many cases

simple trick: better to get rewards sooner than later

\[ y_{i,t} \approx r(s_{i,t}, a_{i,t}) + \gamma \hat{V}_\phi(s_{i,t+1}) \]

discount factor \( \gamma \in [0, 1] \) (0.99 works well)

\( \gamma \) changes the MDP:

\[ p(s' | s, a) = (1 - \gamma) \]
N-step returns

\[
\hat{A}_C^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}^\pi_\phi(s_{t+1}) - \hat{V}^\pi_\phi(s_t)
\]

\[
\hat{A}_{MC}^\pi(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}^\pi_\phi(s_t)
\]

+ lower variance  
- higher bias if value is wrong (it always is)  
+ no bias  
- higher variance (because single-sample estimate)

Can we combine these two, to control bias/variance tradeoff?

\[
\hat{A}_n^\pi(s_t, a_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}^\pi_\phi(s_t) + \gamma^n \hat{V}^\pi_\phi(s_{t+n})
\]

choosing \(n > 1\) often works better!
Policy evaluation example

AlphaGo, Silver et al. 2016

reward: game outcome

value function $\hat{V}_\phi^\pi(s_t)$:

expected outcome given board state
The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

Actor-critic

Case studies: robotics & RLHF
Why is there so many RL algorithms?

Different tradeoffs:
• Continuous vs discrete actions
• Is it easier to learn the environment or the policy?
• Sample complexity

Off or on policy algorithms:
• **Off policy**: able to improve the policy without generating new samples from that policy
• **On policy**: each time the policy is changed, even a little bit, we need to generate new samples
Can policy gradients reuse old data?

Policy gradient: \[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \]

1. fit a model to estimate return
2. generate samples (i.e. run the policy)
3. improve the policy
   \[ \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \]
Importance sampling

\[ \theta^* = \arg\max_\theta E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

what if we don’t have samples from \( \pi_\theta(\tau) \)?

we have samples from \( \tilde{\pi}(\tau) \)

\[ J(\theta) = E_{\tau \sim \tilde{\pi}(\tau)} \left[ \frac{\pi_\theta(\tau)}{\tilde{\pi}(\tau)} r(\tau) \right] \]

\[ E_{x \sim p(x)} [f(x)] = \int p(x) f(x) dx \]

\[ \int p(x) \frac{q(x)}{q(x)} f(x) dx = \int q(x) \frac{p(x)}{q(x)} f(x) dx \]

\[ E_{x \sim q(x)} [\frac{p(x)}{q(x)} f(x)] \]
Importance sampling in policy gradient

policy gradient: \[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \]

\[ \theta^* = \arg \max_\theta E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ J(\theta') = E_{\tau \sim \pi_{\theta'}(\tau)} \left[ \frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} r(\tau) \right] \]

\[ \nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_{\theta'}(\tau)} \left[ \frac{\nabla_{\theta'} \pi_{\theta'}(\tau)}{\pi_\theta(\tau)} r(\tau) \right] = E_{\tau \sim \pi_{\theta'}(\tau)} \left[ \frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right] \]

\[ = E_{\tau \sim \pi_{\theta'}(\tau)} \left[ \left( \frac{\prod_{t=1}^{T} \pi_{\theta'}(a_t|s_t)}{\prod_{t=1}^{T} \pi_\theta(a_t|s_t)} \right) \left( \sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \]
Problem with importance sampling in (policy gradient)

\[ \nabla_{\theta'} J(\theta') = E_{\pi' \sim \pi(\tau)} \left[ \left( \frac{\prod_{t=1}^{T} \pi_{\theta'}(a_t | s_t)}{\prod_{t=1}^{T} \pi_{\theta}(a_t | s_t)} \right) \left( \sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \]

Let’s try it in code!

Importance sampling

\[ E_{x \sim p(x)}[f(x)] = \int p(x)f(x)dx \]

\[ \int p(x) \frac{q(x)}{q(x)} f(x)dx = \int q(x) \frac{p(x)}{q(x)} f(x)dx \]

\[ E_{x \sim q(x)}[\frac{p(x)}{q(x)} f(x)] \]
Solution?

\[ \theta' \leftarrow \arg \max (\theta' - \theta) \nabla_{\theta} J(\theta) \quad \text{s.t.} \quad \|\theta' - \theta\|^2 \leq \epsilon \]

Stay close to the previous policy!

\[ \theta' \leftarrow \arg \max (\theta' - \theta) \nabla_{\theta} J(\theta) \quad \text{s.t.} \quad D_{KL}(\pi_{\theta'}, \pi_{\theta}) \leq \epsilon \]
Trust region policy optimization (TRPO)

Apply all the tricks:
- Use advantage function to reduce the variance
- Use importance sampling to take multiple gradient steps
- Constrain the optimization objective in the policy space

Our optimization problem in Equation (13) is exactly equivalent to the following one, written in terms of expectations:

\[
\begin{align*}
\text{maximize} & \quad \mathbb{E}_{s \sim \rho_{\text{old}}, a \sim q} \left[ \frac{\pi_\theta(a|s)}{q(a|s)} Q_{\text{old}}(s, a) \right] \\
\text{subject to} & \quad \mathbb{E}_{s \sim \rho_{\text{old}}} \left[ D_{\text{KL}}(\pi_\theta(\cdot|s) \parallel \pi_{\text{old}}(\cdot|s)) \right] \leq \delta.
\end{align*}
\]
Proximal policy optimization (PPO)

Apply all the tricks:
• Use advantage function to reduce the variance
• Use importance sampling to take multiple gradient steps
• Constrain the optimization objective in the policy space

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_t(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}$, so $r(\theta_{old}) = 1$. TRPO maximizes a “surrogate” objective

$$L^{CPI}(\theta) = \mathbb{E}_t \left[ \frac{\pi_t(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \hat{A}_t \right] = \mathbb{E}_t \left[ r_t(\theta) \hat{A}_t \right].$$  \hspace{1cm} (6)

The superscript $CPI$ refers to conservative policy iteration [KL02], where this objective was proposed. Without a constraint, maximization of $L^{CPI}$ would lead to an excessively large policy update; hence, we now consider how to modify the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1.

The main objective we propose is the following:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[ \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]$$  \hspace{1cm} (7)
Examples

TRPO applied to continuous control

PPO applied to Dota
The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

**Actor-critic**

Case studies: robotics & RLHF
REINFORCE algorithm:
1. sample \( \{\tau^i\} \) from \( \pi_\theta(a_t|s_t) \) (run the policy)
2. \( \nabla_\theta J(\theta) \approx \sum_i \left( \sum_t \nabla \log \pi_\theta(a_t^i|s_t^i) \right) \left( \sum_t r(s_t^i, a_t^i) \right) \)
3. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

online actor-critic algorithm:
1. take action \( a \sim \pi_\theta(a|s) \), get \( (s, a, s', r) \)
2. update \( \hat{V}_\phi^\pi \) using target \( r + \gamma \hat{V}_\phi^\pi(s') \)
3. evaluate \( \hat{A}^\pi(s, a) = r(s, a) + \gamma \hat{V}_\phi^\pi(s') - \hat{V}_\phi^\pi(s) \)
4. \( \nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \hat{A}^\pi(s, a) \)
5. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

Can we make it more off-policy friendly?

Slide adapted from Sergey Levine
The Plan

- Policy gradients recap
- Variance reduction continued
- Policy gradients tricks
- Actor-critic
- Case studies: robotics & RLHF
Case study: PPO applied to robotics
Case study: PPO applied to robotics

Solving the Rubik’s Cube with a robot hand is still not easy. Our method currently solves the Rubik’s Cube 20% of the time when applying a maximally difficult scramble that requires 26 face rotations. For simpler scrambles that require 15 rotations to undo, the success rate is 60%. When the Rubik’s Cube is dropped or a timeout is reached, we consider the attempt failed. However, our network is capable of solving the Rubik’s Cube from any initial condition. So if the cube is dropped, it is possible to put it back into the hand and continue solving.
Case study: PPO applied to robotics

We train neural networks to solve the Rubik’s Cube in simulation using reinforcement learning and Kociemba’s algorithm for picking the solution steps. A domain randomization enables networks trained solely in simulation to transfer to a real robot.

Simulator physics. We randomize simulator physics parameters such as geometry, friction, gravity, etc. See Section B.1 for details of their ADR parameterization.

Custom physics. We model additional physical robot effects that are not modelled by the simulator, for example, action latency or motor backlash. See [77, Appendix C.2] for implementation details of these models. We randomize the parameters in these models in a similar way to simulator physics randomizations.

Adversarial. We use an adversarial approach similar to [82, 83] to capture any remaining unmodeled physical effects in the target domain. However, we use random networks instead of a trained adversary. See Section B.3 for details on implementation and ADR parameterization.

Observation. We add Gaussian noise to policy observations to better approximate observation conditions in reality. We apply both correlated noise, which is sampled once at the start of an episode and uncorrelated noise, which is sampled at each time step. We randomize the parameters of the added noise. See Section B.4 for details of their ADR parameterization.

Vision. We randomize several aspects in ORBB [16] to control the rendered scene, including lighting conditions, camera positions and angles, materials and appearances of all the objects, the texture of the background, and the post-processing effects on the rendered images. See Section B.5 for details.
Case study: PPO applied to robotics

ADR starts with a single, nonrandomized environment, wherein a neural network learns to solve Rubik’s Cube. As the neural network gets better at the task and reaches a performance threshold, the amount of domain randomization is increased automatically. This makes the task harder, since the neural network must now learn to generalize to more randomized environments. The network keeps learning until it again exceeds the performance threshold, when more randomization kicks in, and the process is repeated.
Case study: PPO applied to robotics
Case study: PPO applied to LLMs (speculations)

RL from human feedback (RLHF)
Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output $s$ to prompt $p$

The reward is higher when humans prefer the output

SAN FRANCISCO, California (CNN) -- A magnitude 4.2 earthquake shook the San Francisco... overturn unstable objects.

An earthquake hit San Francisco. There was minor property damage, but no injuries.

The Bay Area has good weather but is prone to earthquakes and wildfires.

$s_1$

$R(s_1; p) = 0.8$

$s_2$

$R(s_2; p) = 1.2$
Regularizing pre-trained model

• **Challenge:** how do we ensure that $R(s; p)$ prefer natural language generations?

• Since $R(s; p)$ is trained on natural language inputs, it might fail to assign low scores to unnatural $s$.

• **Solution:** add regularization term to $R(s; p)$ that penalizes outputs that deviate from natural language.

\[
\hat{R}(s; p) := R(s; p) - \beta \log \left( \frac{p^{RL}(s)}{p^{PT}(s)} \right)
\]

- This is a penalty which prevents us from diverging too far from the pretrained model.
RLHF + PPO *(speculations)*

**Prompt**

*Explain the moon landing to a 6 year old in a few sentences.*

**Completion**

GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

InstructGPT

People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

---

GPT-3 models aren’t trained to follow user instructions. Our InstructGPT models (highlighted) generate much more helpful outputs in response to user instructions.
**RLHF + PPO (speculations)**

**Step 1**
Collect demonstration data, and train a supervised policy.

30k tasks!

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

**Step 2**
Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

**Step 3**
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.
Case study: PPO applied to LLMs *(speculations)*

- What would you try next?
- Is HF more important than RL?
- What do you think the challenges are?

<table>
<thead>
<tr>
<th><strong>Prompt</strong></th>
<th>Write a short poem about a wise frog.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Completion</strong></td>
<td>GPT-3</td>
</tr>
<tr>
<td></td>
<td>Write a short story in which a character has two different names.</td>
</tr>
<tr>
<td></td>
<td>Write a short story in which you try to get something back that you have lost.</td>
</tr>
<tr>
<td></td>
<td>Write a short story in which a character has a bad dream.</td>
</tr>
<tr>
<td><strong>InstructGPT</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The frog is a symbol of wisdom</td>
</tr>
<tr>
<td></td>
<td>He knows all the secrets of the world</td>
</tr>
<tr>
<td></td>
<td>He is a master of disguise</td>
</tr>
<tr>
<td></td>
<td>And a great teacher of life</td>
</tr>
<tr>
<td></td>
<td>He is a symbol of transformation</td>
</tr>
<tr>
<td></td>
<td>And the bringer of change</td>
</tr>
<tr>
<td></td>
<td>He is the frog who has seen it all</td>
</tr>
<tr>
<td></td>
<td>And knows the meaning of it all</td>
</tr>
</tbody>
</table>
Recap

Key learning goals:

• Practical policy gradient implementation tricks & case studies
• Understanding a generic actor-critic method

PG implementation:

• Variance reduction
• Importance sampling and trust region
• RLHF and robotics applications

AC method:

• Uses the advantage fn
Next

Do we even need a policy?

Can we be even more off-policy?

Q learning and its applications