Actor-Critic Algorithms

CS 285

Instructor: Sergey Levine
UC Berkeley
Recap: policy gradients

REINFORCE algorithm:
1. sample \( \{\tau^i\} \) from \( \pi_\theta(a_t|s_t) \) (run the policy)
2. \( \nabla_\theta J(\theta) \approx \sum_i \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a^i_t|s^i_t) \left( \sum_{t'=t}^T r(s^i_{t'}, a^i_{t'}) \right) \right) \)
3. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

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

“reward to go”

\[
\hat{Q}^\pi(x_t, u_t) = \sum_{t'=t}^T r(x_{t'}, u_{t'})
\]
Improving the policy gradient

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

"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

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) Q(s_{i,t}, a_{i,t}) \]
What about the baseline?

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

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

\[ b_t = \frac{1}{N} \sum_{i} Q(s_{i,t}, a_{i,t}) \text{ average what?} \]

\[ V(s_t) = E_{a_t \sim \pi_\theta(a_t | s_t)}[Q(s_t, a_t)] \]
State & state-action value functions

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

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

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

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

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

unbiased, but high variance single-sample estimate

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

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

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

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

let’s just fit \( V^\pi(s) \)!
Policy evaluation

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

\[ J(\theta) = E_{s_1 \sim p(s_1)}[V^\pi(s_1)] \]

how can we perform policy evaluation?

Monte Carlo policy evaluation (this is what policy gradient does)

\[ V^\pi(s_t) \approx \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \]

\[ V^\pi(s_t) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \]

(requires us to reset the simulator)
Monte Carlo evaluation with function approximation

\[ V^\pi(s_t) \approx \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \]

not as good as this: \[ V^\pi(s_t) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \]

but still pretty good!

training data: \[ \{(s_{i,t}, \sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}))\} \]

\[ y_{i,t} \]

supervised regression: \[ \mathcal{L}(\phi) = \frac{1}{2} \sum_{i} \left\| \hat{V}_\phi^\pi(s_i) - y_i \right\|^2 \]

the same function should fit multiple samples!
Can we do better?

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: \( \{ (s_{i,t}, r(s_{i,t}, a_{i,t}) + \hat{V}_\phi(s_{i,t+1})) \} \)

\( y_{i,t} \)

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

sometimes referred to as a “bootstrapped” estimate
Policy evaluation examples

**TD-Gammon, Gerald Tesauro 1992**

**AlphaGo, Silver et al. 2016**

---

**reward:** game outcome

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

**expected outcome given board state**
From Evaluation to Actor Critic
An actor-critic algorithm

batch actor-critic algorithm:

1. sample \( \{s_i, a_i\} \) from \( \pi_\theta(a|s) \) (run it on the robot)
2. fit \( \hat{V}_\phi^\pi(s) \) to sampled reward sums
3. evaluate \( \hat{A}_\pi^\pi(s_i, a_i) = r(s_i, a_i) + \hat{V}_\phi^\pi(s'_i) - \hat{V}_\phi^\pi(s_i) \)
4. \( \nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}_\pi^\pi(s_i, a_i) \)
5. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

\[
y_{i,t} \approx \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'})
\]

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

\[
V^\pi(s_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(s_{t'}, a_{t'})|s_t]
\]
Aside: discount factors

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

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

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

simple trick: better to get rewards sooner than later

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

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

\( \gamma \) changes the MDP:

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

\[ \hat{p}(s'|s, a) = \gamma p(s'|s, a) \]
Aside: discount factors for policy gradients

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

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

with critic:

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

what about (Monte Carlo) policy gradients?

option 1:

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

not the same!

option 2:

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

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

(later steps matter less)
Which version is the right one?

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

option 2:  \[
\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma^{t-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \left( \sum_{t'=t}^{T} \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)
\]

this is what we actually use... why?

Iteration 2000

later steps don’t matter if you’re dead!

Further reading: Philip Thomas, Bias in natural actor-critic algorithms. ICML 2014
Actor-critic algorithms (with discount)

batch actor-critic algorithm:
1. sample \( \{s_i, a_i\} \) from \( \pi_\theta(a|s) \) (run it on the robot)
2. fit \( \hat{V}_\phi^\pi(s) \) to sampled reward sums
3. evaluate \( \hat{A}^\pi(s_i, a_i) = r(s_i, a_i) + \gamma \hat{V}_\phi^\pi(s'_i) - \hat{V}_\phi^\pi(s_i) \)
4. \( \nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i) \)
5. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

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. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
Actor-Critic Design Decisions
Architecture design

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. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

- no shared features between actor & critic

- + simple & stable

+ two network design

- shared network design
Online actor-critic in practice

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 $r + \gamma \hat{V}_\phi^\pi(s')$  
   works best with a batch (e.g., parallel workers)
3. evaluate $\hat{A}_\pi^\pi(s, a) = r(s, a) + \gamma \hat{V}_\phi^\pi(s') - \hat{V}_\phi^\pi(s)$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \hat{A}_\pi^\pi(s, a)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

synchronized parallel actor-critic

asynchronous parallel actor-critic

get $(s, a, s', r)$
update $\theta$
get $(s, a, s', r)$
update $\theta$
Can we **remove** the on-policy assumption entirely?

**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 $\hat{A}_\pi^\pi(s, a) = r(s, a) + \gamma \hat{V}_\phi^\pi(s') - \hat{V}_\phi^\pi(s)$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \hat{A}_\pi^\pi(s, a)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

form a **batch** by using old previously seen transitions

**off-policy actor-critic**

- get $(s, a, s', r) \leftarrow$
- update $\theta \leftarrow$
- get $(s, a, s', r) \leftarrow$
- update $\theta \leftarrow$
- transitions that we saw in prior time steps
Let’s see what that looks like

online actor-critic algorithm:

1. take action $a \sim \pi_\theta(a|s)$, get $(s, a, s', r)$, store in $R$
2. sample a batch $\{s_i, a_i, r_i, s'_i\}$ from buffer $R$
3. update $\hat{V}_\phi$ using targets $y_i = r_i + \gamma \hat{V}_\phi(s'_i)$ for each $s_i$
4. evaluate $A^\pi(s_i, a_i) = r(s_i, a_i) + \gamma \hat{V}_\phi(s'_i) - \hat{V}_\phi(s_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum \nabla_\theta \log \pi_\theta(a_i|s_i) A^\pi(s_i, a_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

This algorithm is broken!

Can you spot the problems?

$\mathcal{L}(\phi) = \frac{1}{N} \sum \|\hat{V}_\phi(s_i) - y_i\|^2$

replay buffer
Fixing the value function

online actor-critic algorithm:

1. take action $a \sim \pi_\theta(a|s)$, get $(s, a, s', r)$, store in $\mathcal{R}$
2. sample a batch $\{s_i, a_i, r_i, s'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{V}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(s'_i)$ for each $s_i$
4. evaluate $\hat{A}_\pi^\pi(s_i, a_i) = r(s_i, a_i) + \gamma \hat{V}_\phi^\pi(s'_i) - \hat{V}_\phi^\pi(s_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta log \pi_\theta(a_i|s_i) \hat{A}_\pi^\pi(s_i, a_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

not the right target value

not the action $\pi_\theta$ would have taken!

where does this come from?

3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(s'_i)$ for each $s_i, a_i$

$= r_i + \gamma \hat{Q}_\phi^\pi(s'_i, a'_i)$

not from replay buffer $\mathcal{R}$!

$a'_i \sim \pi_\theta(a'_i|s'_i)$

$L(\phi) = \frac{1}{N} \sum_i \left\| \hat{Q}_\phi^\pi(s_i, a_i) - y_i \right\|^2$

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

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

“total reward we get if we take $a_t$ in $s_t$...
... and then follow the policy $\pi^\pi$
Fixing the policy update

online actor-critic algorithm:
1. take action \( a \sim \pi_\theta(a|s) \), get \((s, a, s', r)\), store in \( \mathcal{R} \)
2. sample a batch \( \{s_i, a_i, r_i, s'_i\} \) from buffer \( \mathcal{R} \)
3. update \( \hat{Q}^\pi_\phi \) using targets \( y_i = r_i + \gamma \hat{Q}^\pi_\phi(s'_i, a'_i) \) for each \( s_i, a_i \)
4. evaluate \( \hat{A}^\pi(s_i, a_i) = Q(s_i, a_i) - \hat{V}^\pi_\phi(s_i) \)
5. \( \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i) \)
6. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

not the action \( \pi_\theta \) would have taken!
use the same trick, but this time for \( a_i \) rather than \( a'_i \)!
sample \( a^\pi_i \sim \pi_\theta(a|s_i) \)
\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(a^\pi_i|s_i) \hat{A}^\pi(s_i, a^\pi_i) \]
\rightarrow \text{not from replay buffer } \mathcal{R}! \quad \text{higher variance, but convenient}
why is higher variance OK here?
in practice: \( \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(a^\pi_i|s_i) \hat{Q}^\pi_\phi(s_i, a^\pi_i) \)
What else is left?

online actor-critic algorithm:
1. take action \( a \sim \pi_\theta(a|s) \), get \((s, a, s', r)\), store in \( \mathcal{R} \)
2. sample a batch \( \{s_i, a_i, r_i, s'_i\} \) from buffer \( \mathcal{R} \)
3. update \( \hat{Q}_\phi^\pi \) using targets \( y_i = r_i + \gamma \hat{Q}_\phi^\pi(s'_i, a'_i) \) for each \( s_i, a_i \)
4. \( \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(a_i^\pi|s_i) \hat{Q}_\phi^\pi(s_i, a_i^\pi) \) where \( a_i^\pi \sim \pi_\theta(a|s_i) \)
5. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

Is there any remaining problem?

\( s_i \) didn’t come from \( p_\theta(s) \)
nothing we can do here, just accept it

**intuition:** we want optimal policy on \( p_\theta(s) \)
but we get optimal policy on a *broader* distribution
Some implementation details

online actor-critic algorithm:

1. take action $a \sim \pi_\theta(a|s)$, get $(s, a, s', r)$, store in $\mathcal{R}$
2. sample a batch $\{s_i, a_i, r_i, s'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{Q}_\phi^\pi(s'_i, a'_i)$ for each $s_i, a_i$
4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(a_i^\pi|s_i) \hat{Q}_\phi^\pi(s_i, a_i^\pi)$ where $a_i^\pi \sim \pi_\theta(a|s_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

could also use reparameterization trick to better estimate the integral

lots of fancier ways to fit Q-functions (more on this in next two lectures)

Example practical algorithm:

We’ll also learn about algorithms that do this with deterministic policies later!
Critics as Baselines
Critics as state-dependent baselines

Actor-critic: \[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left( r(s_{i,t}, a_{i,t}) + \gamma \hat{V}^\pi_\phi(s_{i,t+1}) - \hat{V}^\pi_\phi(s_{i,t}) \right) \]

- + lower variance (due to critic)
- - not unbiased (if the critic is not perfect)

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

- + no bias
- - higher variance (because single-sample estimate)

can we use \( \hat{V}^\pi_\phi \) and still keep the estimator unbiased?

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

- + no bias
- + lower variance (baseline is closer to rewards)
Control variates: action-dependent baselines

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

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

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

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

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

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left( \hat{Q}_{i,t} - Q^\pi_\phi(s_{i,t}, a_{i,t}) \right) + \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta E_{a_t \sim \pi_\theta(a_t|s_{i,t})} [Q^\pi_\phi(s_{i,t}, a_t)] \]

use a critic without the bias (still unbiased), provided second term can be evaluated

Gu et al. 2016 (Q-Prop)
Eligibility traces & n-step returns

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

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

+ 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?

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

choosing \( n > 1 \) often works better!
Generalized advantage estimation

Do we have to choose just one \( n \)?

Cut everywhere all at once!

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

\[
\hat{A}_{\text{GAE}}^{\pi}(s_t, a_t) = \sum_{n=1}^{\infty} w_n \hat{A}_n^{\pi}(s_t, a_t)
\]

weighted combination of \( n \)-step returns

How to weight? Mostly prefer cutting earlier (less variance)

\[
\hat{A}_{\text{GAE}}^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma((1 - \lambda)\hat{V}_\phi^{\pi}(s_{t+1}) + \lambda r(s_{t+1}, a_{t+1}) + \gamma((1 - \lambda)\hat{V}_\phi^{\pi}(s_{t+2}) + \lambda r(s_{t+2}, a_{t+2}) + \ldots)
\]

\[
\hat{A}_{\text{GAE}}^{\pi}(s_t, a_t) = \sum_{t'=t}^{\infty} (\gamma \lambda)^{t'-t} \delta_{t'}
\]

\[
\delta_{t'} = r(s_{t'}, a_{t'}) + \gamma \hat{V}_\phi^{\pi}(s_{t'+1}) - \hat{V}_\phi^{\pi}(s_{t'})
\]

similar effect as discount!

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

remember this?

discount = variance reduction!

Schulman, Moritz, Levine, Jordan, Abbeel '16
Review, Examples, and Additional Readings
Review

• Actor-critic algorithms:
  • Actor: the policy
  • Critic: value function
  • Reduce variance of policy gradient

• Policy evaluation
  • Fitting value function to policy

• Discount factors
  • Carpe diem Mr. Robot
  • ...but also a variance reduction trick

• Actor-critic algorithm design
  • One network (with two heads) or two networks
  • Batch-mode, or online (+ parallel)

• State-dependent baselines
  • Another way to use the critic
  • Can combine: n-step returns or GAE
Actor-critic examples

• High dimensional continuous control with generalized advantage estimation (Schulman, Moritz, L., Jordan, Abbeel ‘16)
• Batch-mode actor-critic
• Blends Monte Carlo and function approximator estimators (GAE)
Actor-critic examples

- Asynchronous methods for deep reinforcement learning (Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu ‘16)
- Online actor-critic, parallelized batch
- N-step returns with N = 4
- Single network for actor and critic
Actor-critic suggested readings

• Classic papers

• Deep reinforcement learning actor-critic papers
  • Schulman, Moritz, L., Jordan, Abbeel (2016). High-dimensional continuous control using generalized advantage estimation: batch-mode actor-critic with blended Monte Carlo and function approximator returns