Actor-Critic Algorithms

CS 294-112: Deep Reinforcement Learning
Sergey Levine
Class Notes

1. Homework 1 due today (11:59 pm)!
   • Don’t be late!

2. Homework 2 is out today
   • Start early!

3. Remember to start forming final project groups
Today’s Lecture

1. Improving the policy gradient with a critic
2. The policy evaluation problem
3. Discount factors
4. The actor-critic algorithm
   • Goals:
     • Understand how policy evaluation fits into policy gradients
     • Understand how actor-critic algorithms work
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”
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”

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

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

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

\[V(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q(s_t, a_t)]\]
What about the baseline?

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

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

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

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

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

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

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

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

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

the better this estimate, the lower the variance

unbiased, but high variance single-sample estimate
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?

\( Q^\pi, V^\pi, A^\pi? \)

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

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

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

\[ y_{i,t} \]

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
Policy evaluation examples

**TD-Gammon, Gerald Tesauro 1992**

**AlphaGo, Silver et al. 2016**

---

**Figure 1.** An illustration of the multilayer perceptron architecture used in TD-Gammon’s neural network. This architecture is also used in the popular backpropagation learning procedure. Figure reproduced from [8].

**Figure 2.** An illustration of the normal opening position in backgammon. TD-Gammon has sparked a near-universal conversion in the way experts play certain opening rolls. For example, with an opening roll of 4-1, most players have now switched from the traditional move of 13-8, 6-5, to TD-Gammon’s preference, 13-8, 24-23. TD-Gammon’s analysis is given in Table 2.

---

reward: game outcome  
value function $\hat{V}_\phi^\pi(s_t)$:  
expected outcome given board state

---

reward: game outcome  
value function $\hat{V}_\phi^\pi(s_t)$:  
expected outcome given board state
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 \( \hat{V}_\phi(s) \) to sampled reward sums
3. evaluate \( A^\pi(s_i, a_i) = r(s_i, a_i) + \hat{V}_\phi(s'_i) - \hat{V}_\phi(s_i) \)
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. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

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

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

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

\[ \hat{p}(s'|s, a) = (1 - \gamma) \]
\[ \tilde{p}(s'|s, a) = \gamma p(s'|s, a) \]

\( \gamma \) changes the MDP:

episodic tasks
continuous/cyclical tasks
Aside: discount factors for policy gradients

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

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

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:

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

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

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

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

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

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

this is what we actually use...

why?

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(s) to sampled reward sums
3. evaluate \hat{A}_\pi(s_i, a_i) = r(s_i, a_i) + \gamma \hat{V}_\phi(s'_i) - \hat{V}_\phi(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 \textbf{a} \sim \pi_\theta(a|s), get (s, a, s', r)
2. update \hat{V}_\phi using target \textbf{r} + \gamma \hat{V}_\phi(s')
3. evaluate \hat{A}_\pi(s, a) = r(s, a) + \gamma \hat{V}_\phi(s') - \hat{V}_\phi(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)
Break
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 $\hat{A}_\pi(s, a) = r(s, a) + \gamma \hat{V}_\phi^\pi(s') - \hat{V}_\phi^\pi(s)$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \hat{A}_\pi(s, a)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

synchronized parallel actor-critic

asynchronous parallel actor-critic
Critics as state-dependent baselines

Actor-critic:

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

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

Policy gradient:

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

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

Can we use \( \hat{V}_\phi^\pi \) and still keep the estimator unbiased?

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

+ no bias
+ lower variance (baseline is closer to rewards)

You’ll implement this for HW2!
Control variates: action-dependent baselines

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

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

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

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

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

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

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

Gu et al. 2016 (Q-Prop) – we’ll talk more about variance reduction later
Eligibility traces & n-step returns

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

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

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!

- higher bias if value is wrong (it always is)
- higher variance (because single-sample estimate)

+ lower variance
+ no bias

bigger variance

cut here before variance gets too big!

smaller variance
Generalized advantage estimation

Do we have to choose just one $n$?

Cut everywhere all at once!

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

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

Weighted combination of $n$-step returns

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

\[ w_n \propto \lambda^{n-1} \]

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

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

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

similar effect as discount!

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

remember this?

discount = variance reduction!

Schulman, Moritz, Levine, Jordan, Abbeel ‘16
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

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