Actor Critic Notes

1 Prelude

In policy gradient, $\hat{Q}_{i,t}$ is the estimate of expected reward if we take action $a_{i,t}$ in state $s_{i,t}$. We can lower the variance of policy gradient by using the true expected reward-to-go:

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

We can then use that in place of $\hat{Q}_{i,t}$:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) Q(s_{i,t}, a_{i,t}) \tag{2}$$

We can also define $b = \frac{1}{N} \sum_i Q(s_{i,t}, a_{i,t})$, and then we have:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) (Q(s_{i,t}, a_{i,t}) - b) \tag{3}$$

We can also use $V(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q(s_t, a_t)]$ as the baseline:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) (Q(s_{i,t}, a_{i,t}) - V(s_{i,t})) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A(s_{i,t}, a_{i,t}) \tag{4}$$

The better advantage estimate is, the lower the variance.

1.1 Value function fitting

We can rewrite Q^π as $Q^\pi(s_t, a_t) = r(s_t, a_t) + \sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]$, which is equivalent to:

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[V^\pi(s_{t+1})] \tag{5}$$

which could further be approximated as:

$$Q^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) \tag{6}$$

which leads to the following approximation:

$$A^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t) \tag{7}$$

We can then use NN to fit $V^\pi(s_t)$. This is called policy evaluation. We can write the RL objective as:

$$J(\theta) = E_{s_1 \sim p(s_1)}[V^\pi(s_1)] \tag{8}$$

Monte-carlo policy evaluation:

$$V^\pi(s_t) \approx \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \tag{9}$$

for single trajectory and:

$$V^\pi(s_t) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \tag{10}$$

1.2 Monte-Carlo evaluation with function approximation

Single trajectory might not be as good as multiple trajectory rollout, but still pretty good.
training data: $\left\{\left(s_{i,t}, \sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})\right)\right\}$. We can denote the second term as $y_{i,t}$, and use the supervised regression to train the network: $\mathcal{L}(\phi) = \frac{1}{2}\sum_i ||\hat{V}_\phi^\pi(s_i) - y_i||^2$
We can further rewrite:

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

which directly uses previously fitted value function to get the estimate, which is called "bootstrap" estimate.

2 Batch actor-critic algorithm

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

in which $V^\pi(s_{i,t}) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_{i,t}]$.

2.1 Infinite episode length

$y_{i,t} \approx r(s_{i,t}, a_{i,t}) + \gamma\hat{V}_\phi^\pi(s_{i,t+1})$ that includes a discount factor that reduces future rewards. The introduction of γ changes the MDP and adds another "death" state that once the agent is trapped (with probability $1-\gamma$), it will never get out. For Monte Carlo policy gradients, we have:
Option 1:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left(\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{i,t'}, a_{i,t'})\right) \tag{11}$$

Option 2:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_i|s_i)\right) \left(\sum_{t'=1}^{T} \gamma^{t'-1} r(s_{i,t'}, a_{i,t'})\right) \tag{12}$$

Option 2 is basically equivalent to:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \gamma^{t-1}\nabla_\theta \log \pi_\theta(a_i|s_i) \left(\sum_{t'=1}^{T} \gamma^{t'-t} r(s_{i,t'}, a_{i,t'})\right) \tag{13}$$

that takes a lighter discount the policy gradient that influences future decisions. (later steps don't matter if you're dead)
Discount factor serves to reduce variance of policy gradient at the cost of introducing more bias.

The modified batch actor-critic algorithm would therefore be:

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

online actor-critic algorithm:

- take action $a \sim \pi_\theta(a|s)$, get (s, a, s', r)
- update $\hat{V}^\pi$ using target $r + \gamma \hat{V}^\pi(s')$
- evaluate $\hat{A}^\pi(s, a) = r(s, a) + \gamma \hat{V}^\pi(s') - \hat{V}^\pi(s)$
- $\nabla_\theta J(\theta) \approx \nabla_\theta \log_{\pi_\theta}(a|s) \hat{A}^\pi(s, a)$
- $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

2.2 Architecture Design

2.2.1 Two network design

simple & stable but no shared features between actor & critic.

2.2.2 Shared network design

shared network for value and critic

2.3 Off-policy actor-critic

Actor-critic works best with a batch(e.g. parallel workers)
Asynchronous update: will perform update if there are enough transitions no matter which parameter is used for generating them (might be generated by slightly older actors)

2.3.1 Off-policy AC

Use a replay buffer to store historical transitions:

- take action $a \sim \pi_\theta(a|s)$, get (s, a, s', r) and store in $\mathcal{R}$
- sample a batch $\{s_i, a_i, r_i, s'_i\}$ from buffer $\mathcal{R}$
- update $\hat{V}^\pi$ using target $y = r + \gamma \hat{V}^\pi(s')$ for each s_i
- evaluate $\hat{A}^\pi(s, a) = r(s, a) + \gamma \hat{V}^\pi(s') - \hat{V}^\pi(s)$
- $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log_{\pi_\theta}(a_i|s_i) \hat{A}^\pi(s, a)$
- $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

This algorithm is broken!

We can use $\hat{Q}^\pi_\phi$ instead of the value function to perform the update:

- take action $a{\sim}\pi_\theta(a|s)$, get (s, a, s', r) and store in $\mathcal{R}$
- sample a batch $\{s_i, a_i, r_i, s'_i\}$ from buffer $\mathcal{R}$
- update $\hat{Q}^\pi_\phi$ using target $y = r + \gamma \hat{Q}^\pi_\phi(s'_i, a'_i)$ for each s_i, a_i
- evaluate $\hat{A}^\pi(s, a) = Q(s_i, a_i) - \hat{V}^\pi(s)$
- $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log_{\pi_\theta}(a^\pi_i|s_i)\hat{A}^\pi(s, a)$
- $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

We can do the same for policy gradient by taking action a^π_i instead of sampling from replay buffer $\mathcal{R}$. Variance might be higher but it can be reduced as we no longer have to interact with the simulator. We can even drop step 4 as we do not care about higher variance (can remove the baseline):

- take action $a{\sim}\pi_\theta(a|s)$, get (s, a, s', r) and store in $\mathcal{R}$
- sample a batch $\{s_i, a_i, r_i, s'_i\}$ from buffer $\mathcal{R}$
- update $\hat{Q}^\pi_\phi$ using target $y = r + \gamma \hat{Q}^\pi_\phi(s'_i, a'_i)$ for each s_i, a_i
- $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log_{\pi_\theta}(a^\pi_i|s_i)\hat{Q}^\pi(s_i, a^\pi_i)$ where $a^\pi_i{\sim}\pi_\theta(a|s_i)$
- $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

s_i didn't come from $p_\theta(s)$ and is not a big issue as we will get optimal policy on a broader distribution.

2.4 Critic as state-dependent baselines

We can use $\hat{V}^\pi_\phi$ and still keep the estimator unbiased:

$$\nabla_\theta \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left(\left(\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{i,t'}, a_{i,t'}) \right) - \hat{V}^\pi_\phi(s_{i,t}) \right) \tag{14}$$

We can even use $\hat{Q}^\pi_\phi$ to further lower the variance though we have to account for the extra error term: (Q-prop)

$$\nabla_\theta \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})(\hat{Q}_{i,t} - \hat{Q}^\pi_\phi(s_{i,t}, a_{i,t})) + \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta E_{a{\sim}\pi_\theta(a_t|s_{i,t})}[Q^\pi_\phi(s_{i,t}, a_{i,t})] \tag{15}$$

2.5 Eligibility trace and n-step returns

Actor critic uses:

$$\hat{A}^\pi_C(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}^\pi_\phi(s_{t+1}) - \hat{V}^\pi_\phi(s_t) \tag{16}$$

which has lower variance and higher bias if the value is wrong compared to policy gradient. The policy gradient uses Monte-carlo advantage estimator:

$$\hat{A}^\pi_{\text{MC}}(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}^\pi_\phi(s_t) \tag{17}$$

which has no bias but higher variance due to single-sample estimate. We can use n-step return estimator to lower the variance:

$$\hat{A}^\pi_n(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}^\pi_\phi(s_t) + \gamma^n \hat{V}^\pi_\phi(s_{t+n}) \tag{18}$$

choosing $n > 1$ often works better

2.6 Generalized Advantage Estimation

$$\hat{A}^{\pi}_{\text{GAE}}(s_t, a_t) = \sum_{n=1}^{\infty} \omega_n \hat{A}^{\pi}_n(s_t, a_t) \tag{19}$$

We can use exponential falloff ($\omega_n \propto \lambda^{n-1}$) as we prefer to cut earlier that can result in less variance:

$$\hat{A}^{\pi}_{\text{GAE}}(s_t, a_t) = \sum_{t'=t}^{\infty} (\gamma\lambda)^{t'-t} \delta_{t'} \tag{20}$$

in which $\delta_{t'} = r(s_{t'}, a_{t'}) - \hat{V}^{\pi}_{\phi}(s_{t'}) + \gamma \hat{V}^{\pi}_{\phi}(s_{t'+n})$