Actor-Critic Algorithms (from Berkeley CS294)

Aaron Lou

May 2018

1 Definitions and introduction

We define the values Q, V, A (where Q is the quality, V is the expected value, and A is the advantage) under policy π as

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

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

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

We can reformulate the gradient of J, the expected value of the total reward, by

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

as opposed to the original policy gradient value of

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

We attempt to fit a model to V^π to help us with our policy gradient techniques. This is because

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

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

This is our "critic" neural network $s \to \widehat{V}^\pi_\phi(s)$

2 Policy Evaluation

We note that our value of V is given by

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

and our J is given by

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

Using **monte carlo** policy evaluation, we can repeatedly sample and get a value of

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

and our respective training data is

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

and we denote $y_{i,t}$ as the individual elements of this data.

We wish to minimize the mean square error

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_{i} ||\widehat{V}_{\phi}^{\pi}(s_i) - y_i||^2$$

Using **boot strap** method, we can just assume that the other terms of our training data can be evaluated based on our model, so we have

$$y_{i,t} = r(s_{i,t}, a_{i,t} + \widehat{V}_{\phi}^{\pi}(s_{i,t+1}))$$

and our training data becomes

$$\left\{ \left(s_{i,t}, r(s_{i,t}, a_{i,t}) + \widehat{V}_{\phi}^{\pi}(s_{i,t+1}) \right) \right\}$$

Where we just apply regression again.

3 Training algorithms

Batch Actor-Critic Algorithm

1. Sample $\{s_i, a_i\}$ from $\pi_\theta(a \mid s)$
2. Fit $\widehat{V}_{\phi}^{\pi}(s)$ to sampled reward sums

3. Evaluate $\widehat{A}^{\pi}(s_i, a_i) = r(s_i, a_i) + \widehat{V}^{\pi}_{\phi}(s'_i) - \widehat{V}^{\pi}_{\phi}(s_i)$
4. Calculate the gradient $\nabla_{\theta} J(\theta) \approx \sum_i \nabla_{\theta} \log \phi_{\theta}(a_i \mid s_i) \widehat{A}^{\pi}(s_i, a_i)$
5. Update $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$

Where we have

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

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

A note about *Discount Factors*:

We have $y_{i,t} \approx r(s_{i,t}, a_{i,t}) + \widehat{V}^{\pi}_{\phi}(s_{i,t+1})$, but need to consider what happens when $T \to \infty$. We can add a $\gamma \in [0, 1]$ (mostly 0.99) to the function as

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

In particular, this creates a "death state" in our MDP. We have two options, causality and no causality, respectively.

Option 1: Our gradient is given by

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

Option 2: Our gradient is given by

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

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

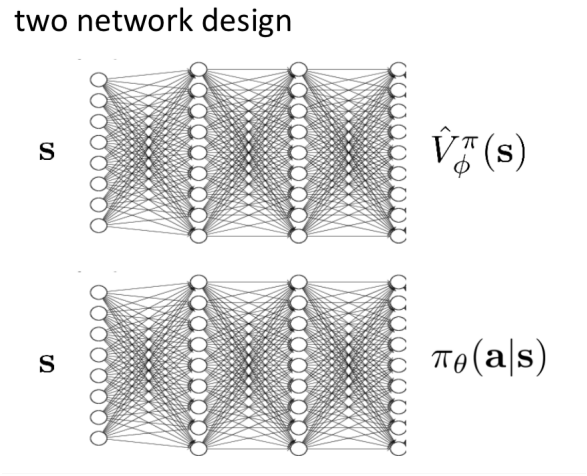
We normally use the first option, since we care about the infinite horizon case. We use the second option only sometimes.

Online Actor - Critic Algo

1. Take action $a \sum \pi_{\theta}(a|s)$ and get the values(s, a, s', r)
2. Update $\widehat{V}^{\pi}_{\phi}$ with $r + \gamma \widehat{V}^{\pi}_{\phi}(s')$
3. Evaluate $\widehat{A}^{\pi}(s, a) = r(s, a) + \gamma \widehat{V}^{\pi}_{\phi}(s') - \widehat{V}^{\pi}_{\phi}(s)$
4. Calculate $\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \log \pi_{\theta}(a \mid s) \widehat{A}^{\pi}(s, a)$
5. Take gradient step $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$.

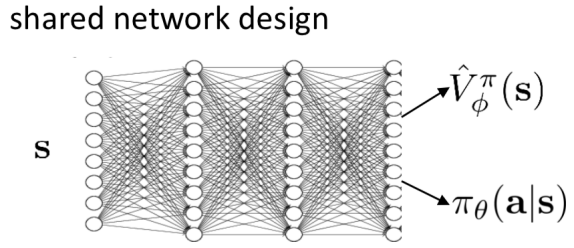
4 Architecture Design for Online AC Algo

Two network design



Simple, but doesn't make use of shared features.

One network design



Much hard to train (two different gradients).

Parallelism on Online AC

Synchronized: use multiple threads threads to get many points at each time step. Update the gradient using all of them.

Asynchronized: create a server of parameters. As each thread completes its actions, send the parameters to be updated. Note that this assumes that we can update current values with old derivations.

5 Critis as baselines

State-dependent

We can add in $\widehat{V}$ as the baseline to our original policy gradient as follows:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})((\sum_{t'=t}^{T}\gamma^{t'-t}r(s_{i,t'}, a_{i,t'})) - \widehat{V}_\phi^\pi(s_{i,t}))$$

And it has no bias and lower variance.

Action - dependent

Under our current values, we have

$$\widehat{A}^\pi(s_t, a_t) = \sum_{t'=t}^{\infty}\gamma^{t'-t}r(s_{t'}, a_{t'}) - V_\phi^\pi(s_t)$$

This has no bias but high variance, but if we pick

$$\widehat{A}^\pi(s_t, a_t) = \sum_{t'=t}^{\infty}\gamma^{t'-t}r(s_{t'}, a_{t'}) - Q_\phi^\pi(s_t)$$

This goes to 0 if critic is right, but is not correct. However, we can add a correcting term to be

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

N-step returns

If we combine

$$\widehat{A}_C^\pi(s_t, a_t) = r(s_t, a_t) + \gamma\widehat{V}_\phi^\pi(s_{t+1}) - \widehat{V}_\phi^\pi(s_t)$$
$$\widehat{A}_C^\pi(s_t, a_t) = \sum_{t'=t}^{\infty}\gamma^{t'-t}r(s_{t'}, a_{t'}) - \widehat{V}_\phi^\pi(s_t)$$

We can cut off at n, where n is the number of steps we look ahead

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

Generalized Advantage Estimation
We can take another more general value

$$\widehat{A}_{GAE}^\pi(s_t, a_t) = \sum_{i=1}^{\infty}\omega_n\widehat{A}_n^\pi(s_t, a_t)$$

Where $\omega_n \propto \lambda^{n-1}$. We have

$$\widehat{A}^{\pi}_{GAE}(s_t, a_t) = r(s_t, a_t) + \gamma((1-\lambda)\widehat{V}^{\pi}_{\phi}(s_{t+1}) + \lambda(r(t+1, a_{t+1}) \dots$$

$$\implies \widehat{A}^{\pi}_{GAE}(s_t, a_t) = \sum_{t'=t}^{\infty} (\gamma\lambda)^{t'-t} \delta_t$$

Where $\delta_t = r(s_{t'}, a_{t'}) + \gamma \widehat{V}^{\pi}_{\phi}(s_{t'+1}) - \widehat{V}^{\pi}_{\phi}(s_{t'})$.