Actor Critic Methods - II

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : easwar.subramanian@tcs.com / cs5500.2020@iith.ac.in

October 18, 2022

Overview

Review

Optimization in Policy Space

- **Value-based control**: Estimate Q^*, and implement greedy policy
  - Not possible to implement in continuous action spaces
- **Policy-based methods**: Optimize directly in policy space
  - Parametrize policies by parameter θ
  - Optimize expected total return using gradient of expected total return w.r.t. θ
  - Can handle continuous action spaces
  - But can have slow convergence due to high variance

Can both types of methods be combined ?

Towards Actor-Critic Algorithms

The policy gradient estimate with temporal structure (takes causality into account) is given by

$$\nabla_\theta J(\theta) \quad = \quad \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t|s_t) G_{t:\infty}(\tau)\right]$$

where

$$G_{a:b}(\tau) = \sum_{t=a}^{b} \gamma^t r_{t+1}$$

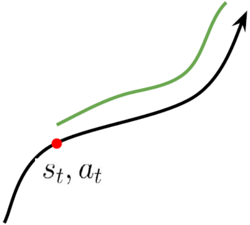
Sample estimate is given by

$$\nabla_\theta J(\theta) \approx \frac{1}{K}\sum_{i=1}^{K}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t^{(i)}|s_t^{(i)}) \cdot \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'+1}^{(i)}\right]\right]$$

- This gradient estimate is the starting point of actor-critic algorithms

The Critic

$$\nabla_\theta J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t^{(i)} | s_t^{(i)}) \cdot \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'+1}^{(i)} \right] \right]$$



The inner summation is an estimate of $Q^{\pi_\theta}(s_t, a_t)$!!

Figure Source: Jie-Han-Chen:SlideShare

Policy Gradient Theorem

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\left\{\nabla_\theta \log \pi_\theta(a_t|s_t) G_{t:\infty}(\tau)\right\}\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\mathbb{E}_{\tau\sim\pi_\theta}\left(\left\{\nabla_\theta \log \pi_\theta(a_t|s_t) G_{t:\infty}(\tau)\right\}\middle|\, s_t, a_t\right)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\left\{\nabla_\theta \log \pi_\theta(a_t|s_t)\, \underbrace{\mathbb{E}_{\tau\sim\pi_\theta}\left(G_{t:\infty}(\tau)\middle|\, s_t, a_t\right)}_{??}\right\}\right]
\end{aligned}
$$

Policy Gradient Theorem

Policy Gradient Theorem

For suitable objective function $J(\theta)$, we have,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}\left\{\nabla_\theta \log \underbrace{\pi_\theta(a_t|s_t)}_{\text{Actor}}\underbrace{Q^{\pi_\theta}(s_t, a_t)}_{\text{Critic}}\right\}\right]$$

- Replacaes the single path reward by $Q^{\pi_\theta}(s_t, a_t)$
- Policy gradient theorem applies to start state objective, average reward objective and average value objective

(More on this later !!)

Advantage Function and its Estimators

Policy Gradient with Baseline

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \left\{\nabla_\theta \log \pi_\theta(a_t|s_t)(G_{t:\infty} - b(s_t))\right\}\right] \\
&= \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \left\{\nabla_\theta \log \pi_\theta(a_t|s_t)(G_{t:\infty} - \underbrace{\mathbb{E}_{\pi_\theta}(G_{t:\infty}|s_t))}_{??}\right\}\right] \\
&= \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \left\{\nabla_\theta \log \pi_\theta(a_t|s_t)(Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t))\right\}\right]
\end{aligned}
$$

Advantage Function

- Advantage function

$$A^{\pi_\theta}(s,a) \stackrel{\text{def}}{=} Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}\left\{\nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t,a_t)\right\}\right]$$

How can we estimate the advantage function using samples ?

Vanilla Policy Gradient Algorithm

Algorithm Vanilla Policy Gradient Algorithm

1: Initialize policy network π with parameters θ_1 learning rate α and baseline b

2: **for** $n = 1$ to N **do**

3: Sample K trajectories by executing the policy π_{θ_n}

4: At each time step of each trajectory compute $G_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t+1}$ and advantage estimate $A_t = G_t - b(s_t)$

5: Calculate gradient estimate

$$\nabla_{\theta_n} J(\theta_n) \approx \frac{1}{K} \sum_{i=1}^{K} \left[\sum_{t=0}^{\infty} \nabla_{\theta_n} \log \pi(a_t^{(i)} | s_t^{(i)}) A_t \right]$$

6: Perform gradient update

$$\theta_{n+1} = \theta_n + \alpha \nabla_{\theta_n} J(\theta_n)$$

7: **end for**

Advantage Function Estimate

$$A^{\pi_\theta} = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t+1} - b(s_t)$$

where

$$b(s_t) = \frac{1}{K} \sum_{i=1}^{K} G_{t:\infty}(\tau^{(i)})$$

(time dependent baseline)

- Unbiased estimate but variance is high due to the fact that it is a single sample estimate
- But, we can't roll out trajectories from state s_t as we also need a the algorithm to be online

Estimator for Advantage Function

- Consider the definition of advantage function

$$A^{\pi}(s,a) \stackrel{\text{def}}{=} Q^{\pi}(s,a) - V^{\pi}(s)$$

- Try having function approximator V_ϕ for V^{π_θ}
- Consider one-step TD error for V^{π_θ}

$$\delta_t^{\pi_\theta} = r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$$

$$\begin{aligned} \mathbb{E}_{\pi_\theta}(\delta^{\pi_\theta} | s_t, a_t) &= \underbrace{E_{\pi_\theta}(r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) | s_t, a_t)}_{??} - V^{\pi_\theta}(s_t) \\ &= Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t) = A^{\pi_\theta}(s_t, a_t) \end{aligned}$$

- The one-step TD error is an unbiased estimate of the advantage function

$$\therefore \quad \nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \left\{\nabla_\theta \log \pi_\theta(a_t|s_t) \delta_t^{\pi_\theta}\right\}\right]$$

Estimator for Advantage Function

- In practice, use the approximate TD error using the function approximator V_ϕ (for V^{π_θ}) as an estimate of the advantage function

$$A^{\pi_\theta}(s_t, a_t) \approx r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

- Note : If we fit V_ϕ using Fitted V iteration, the approximator is biased

Actor Critic Algorithms

Batch Actor Critic Algorithm

Algorithm Batch Actor-Critic Algorithm

1: Initialize critic ϕ, actor θ
2: **for** Repeat over several transitions **do**
3: Sample K transitions (s_i, a_i, r_i, s'_i) using π_θ
4: Fit $V_\phi(s_i)$ to sampled reward sums from s_i
5: Evaluate the advantage function (for all K samples) using

$$A^{\pi_\theta}(s_i, a_i) \approx r_i + \gamma V_\phi(s'_i) - V_\phi(s_i)$$

6: Update actor $\theta \leftarrow \theta + \alpha \sum_{i=1}^{K} \nabla_\theta \log \pi_\theta(a_i|s_i) A^{\pi_\theta}(s_i, a_i)$
7: **end for**

The V function can be fitted using fitted V iteration

Online Actor Critic Algorithm

Algorithm Online Actor-Critic Algorithm

1: Initialize state s, critic ϕ, actor θ
2: **for** Repeat over several transitions **do**
3: Let a be the action suggested by policy π_θ at state s
4: Take action a, observe reward r and next state s' and get a transition (s, a, r, s')
5: Fit $V_\phi(s)$ using target $r + V_\phi(s')$
6: Evaluate the advantage function using

$$A^{\pi_\theta}(s,a) \approx r + \gamma V_\phi(s') - V_\phi(s)$$

7: Compute $\nabla_\theta J(\theta) \leftarrow \nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)$
8: Update actor $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
9: **end for**

- Fitting V_ϕ has moving target and data correlation problem
- The gradient update of the actor in Step 6 has lot of variance (single sample estimate)

Advantage Actor Critic Algorithms

Steps 5 and 7 works best with a batch (parallel workers)

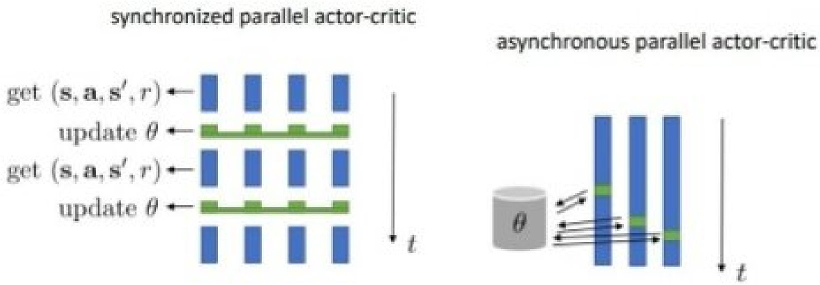


Figure Source: UCB : Sergey Levine

On Applicabilty of A3C Algorithms

- The A3C (with its synchronous version) requires multiple worker threads to simulate samples for gradient computation
- Useful when simulators are available. Instantiate multiple copies of simulator
- In many real applications, this can be an expensive step
  - ★ Navigation of physical robots (require many physical robots)
  - ★ Driving a car (requires samples generated from multiple cars)

Towards n-step returns

- One step TD error based Advantage estimate

$$A_{\mathrm{C}}^{\pi_\theta}(s,a) \approx r + \gamma V_\phi(s') - V_\phi(s)$$

  - ★ Low variance
  - ★ Biased due to the use of function approximators

- Monte Carlo based Advantage estimate

$$A_{\mathrm{MC}}^{\pi_\theta}(s,a) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t+1} - b(s)$$

  - ★ High variance
  - ★ No bias

Towards n-step returns

- We considered the critic who provides one-step TD error

$$\delta_t^{(1)} = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

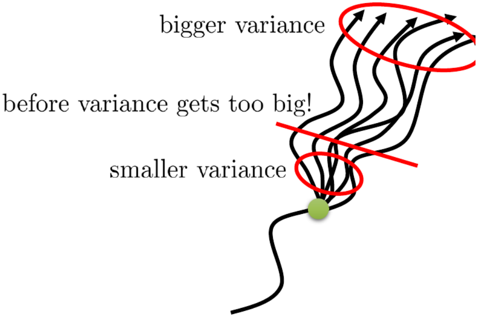
  as feedback to the actor

- We could also consider a critic that provides n-step TD error as feedback to the actor where the n-step TD error

$$\delta_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}) - V(s_t)$$

- In theory, $\delta_t^{(n)}$ is also an unbiased estimate of A^{π_θ} if $V = V^{\pi_\theta}$
- Gives rise to a method called Generalized Advantage Estimation (GAE)

Towards n-step returns



$$A_n^{\pi_\theta}(s_t, a_t) \approx \sum_{t'=t}^{t+n} \gamma^{t'-t} r(s'_t, a'_t) + \gamma^n V_\phi(s_{t+n}) - V_\phi(s_t)$$

Figure Source: UCB : Sergey Levine

TD-λ as Targets

- We could also consider the TD(λ) error given by

$$\delta_t^{(\lambda)} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1}\delta_t^{(n)}$$

  for the critic formulation (again unbiased in theory)
- The critic itself can be updated using TD(λ)
- Both TD(λ) (critic and the feedback) updates can be implemented using eligibility traces

A3C

- Asynchronous methods for deep Asynchronous methods for deep reinforcement learning (2016)
- Online actor crtic and paralleized batch
- N-step returns with N = 4 steps
- Single network for actor and critic

Different Policy Gradient Formulations

Gradient of the performance measure is given by

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) \Psi_t \right]$$

1. $\Psi_t = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = G_0$, Total reward of the trajectory
2. $\Psi_t = \sum_{t'=t}^{\infty} \gamma^{t'} r_{t'+1} = G_{t:\infty}$,Total reward following action a_t
3. $\Psi_t = \sum_{t'=t}^{\infty} \gamma^{t'} r_{t'+1} - b(s_{t'}) = G_{t:\infty} - b(s_t)$, Baseline version of the previous formula
4. $\Psi_t = \gamma^t Q^{\pi_\theta}(s_t, a_t)$, State action value function
5. $\Psi_t = \gamma^t A^{\pi_\theta}(s_t, a_t) = \gamma^t \left[Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t) \right]$, Advantage function
6. $\Psi_t = \gamma^t \left[r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t) \right]$, TD residual

Towards Deterministic Policy Gradient Formulations

Stationary Distribution of Markov Chain

- Given a MDP $< \mathcal{M} = \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ and a policy π_θ, we have an induced Markov chain given by $< \mathcal{S}, \mathcal{P}^{\pi_\theta} >$
- Imagine that you can travel along the Markov chain's states forever, and eventually, as the time progresses, the probability of you ending up with at state s from state s_0 (start state) becomes unchanged and is given by

$$d^{\pi_\theta}(s) = \lim_{t \to \infty} \mathbb{P}(s_t = s | s_0, \pi_\theta)$$

- The entity $d^{\pi_\theta}(s)$ is the limiiting (stationary as well) distribution of Markov chain and is assumed to independent of s_0
- Existence of such stationary distribution can be guaranteed under certain some conditions on the Markov chain

Objective Function Formulations

- In episodic environments, we can use the value of the start state as the objective function given by

$$J_1(\theta) = V^{\pi_\theta}(s) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right]$$

- In **continuing** environments we have a slightly different formulation for the objective function given by,

$$J_{avV}(\theta) = \sum_{s} d^{\pi_\theta}(s) V^{\pi_\theta}(s) = \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q^{\pi_\theta}(s,a)$$

  where

$$d^{\pi_\theta}(s) = \lim_{t \to \infty} \mathbb{P}(s_t = s | s_0, \pi_\theta)$$

- **Idea** : Average of $V^{\pi_\theta}(s)$ computed using $d^{\pi_\theta}(s)$ as weights (for all $s \in \mathcal{S}$).
- Average is computed from the tail of episodic sequence starting at state s_0
- Second equality uses the relationship between V^{π_θ} and Q^{π_θ}

Stochastic Policy Gradient Theorem

Stochastic Policy Gradient Theorem
For any differentiable policy π_θ, for any of the policy objective functions $J(\theta) = J_1(\theta)$, $\frac{1}{1-\gamma} J_{avV}(\theta)$, the gradient estimate of the objective function with respect to the parameter θ, under some conditions, is given by,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}\left\{\nabla_\theta \log \underbrace{\pi_\theta(a_t|s_t)}_{\text{Actor}} \underbrace{Q^{\pi_\theta}(s_t, a_t)}_{\text{Critic}}\right\}\right]$$