Actor Critic Methods – II

Easwar Subramanian

TCS Innovation Labs, Hyderabad

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

Novemer 01, 2021

Overview

Actor Critic Methods - Recap

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 ?

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

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

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

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

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

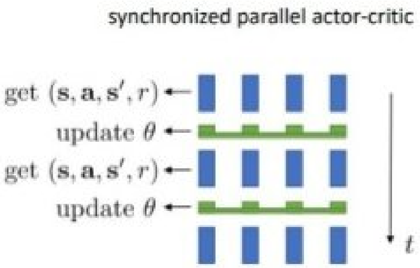
$$A^{\pi_\theta}(s_i, a_i) \approx r_i + \gamma V_\phi(s'_i) - V_\phi(s_i)$$

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 Algorithms

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



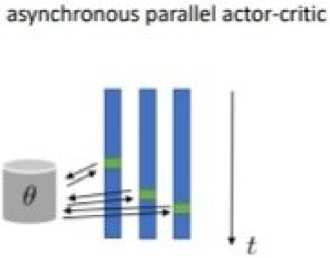


Figure Source: UCB : Sergey Levine

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)

Results

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

Towards N-Step Estimators

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

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

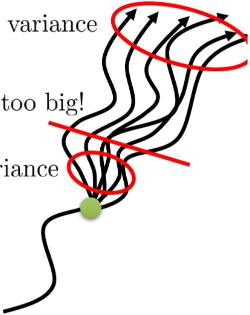
  as feedback to the actor

- 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

Policy Gradient Formulations : Summary

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

Objective Function 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}

TCS Research & Innovation

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,

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

Deterministic Policy Gradient Algorithm

Deterministic Policy Gradient Algorithm : Key Ideas

- Thus far, considered the policy function $\pi(\cdot|s)$ as a probability distribution over actions space and thus considered stochastic policies
- **Deterministic policy gradient algorithms (DPG)** instead models the policy as a deterministic decision : $a = \pi(s)$
- Specifically DDPG, an off-policy actor-critic algoirthm, can be thought of as DQN for continuous action space setting
- Interleaves between learning optimal action-value function $Q^*(s, a)$ and learning optimal policy $\pi^*(s)$
- Uses Bellman equation to learn $Q^*(s, a)$ and policy gradients to learn $\pi^*(s)$

Deterministic Policy Gradient Algorithm : Key Ideas

- Bellman equation is the starting point for learning optimal action-value function $Q^*(s, a)$.
- Optimal action-value function int the DQN setting is learnt using the following MSBE function

$$L_i(\phi_i) = \left[\mathbb{E}_{(s,a,r,s')\in D}\left(Q_{\phi_i}(s,a) - \underbrace{r + \max_{a'} Q_{\phi_i'}(s', a')}_{\text{target}}\right)\right]^2$$

- However, in the DDPG setting, we calculate the max over actions using the policy netwoek as follows,

$$L_i(\phi_i) = \left[\mathbb{E}_{(s,a,r,s')\in D}\left(Q_{\phi_i}(s,a) - \underbrace{r + Q_{\phi_i'}(s', \pi_\theta(s'))}_{\text{target}}\right)\right]^2$$

Deterministic Policy Gradient Algorithm : Key Ideas

- Policy is learnt by recoginizing that we are looking for a deterministic policy $\pi_\theta(s)$ that gives an action that maximizes $Q_\phi(s,a)$. Achieved by, performing gradient ascent on the following objective function

$$\max_\theta \mathbb{E}_{s \in \mathcal{D}} Q_\phi(s, \pi_\theta(s))$$

- Because the policy that is being learnt is deterministic, to make DDPG policies explore better, we add noise to their actions at training time.
  - ★ OU noise
  - ★ zero-mean Gaussian noise
- Target networks are updated using Polyak averaging
- The idea of deterministic policy gradient has connections to the stochastic policy gradient setting (in the limiting case)

Deep Deterministic Policy Gradient (DDPG)

Algorithm Deep Deterministic Policy Gradient

Initialize state s, critic ϕ, actor θ and replay buffer
Initialize target critic $\phi' \leftarrow \phi$, target actor $\theta' \leftarrow \theta$
for Repeat over several episodes **do**
    Initialize a random process N for exploration (eg. Ornstein-Uhlenbeck process), and observe initial state s
    for Repeat over transitions **do**
        Apply action $a = \pi_\theta(s) + N_t$, observe reward r and next state s', and store the transition (s, a, r, s') in the replay buffer
        Sample a random minibatch of transitions (s_i, a_i, r_i, s'_i) from the buffer
        Compute SARSA target values $y_i = r_i + Q_{\phi'}(s'_i, \pi_{\theta'}(s'_i))$
        Update critic by minimizing MSE loss $\frac{1}{n}\sum_i (y_i - Q_\phi(s_i, a_i))^2$
        Update actor using sampled deterministic policy gradient $\frac{1}{n}\sum_i \nabla_a Q_\phi(s_i, \pi_\theta(s_i)) \nabla_\theta \pi_\theta(s_i)$
        Perform soft updates on target networks
            $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
            $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
    end for
end for