ACTOR-CRITIC METHODS: A2C

NOTES

February 24, 2020

Actor-critic methods aim at combining the strong points of actor-only (e.g., gradient) and critic-only methods (e.g., TD, MC), by incorporating value function approximation in the policy gradient methods. We already saw the potential of using value function approximation for picking baseline for variance reduction. Another more obvious place to incorporate Q-value approximation is for approximating Q-function in the policy gradient expression. Let $J(\pi) = \mathbb{E}_{s\sim\rho}[V^\pi(s)] := \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r_t]$. Then, the policy gradient [1]

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi}\left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right]$$

where $d^\pi(s) = \lim_{T\to\infty}\sum_{t=0}^{T}\gamma^t \mathbb{P}(s_t = s|\pi,\rho)$ where ρ is the initial state distribution.

In the vanilla policy gradient algorithm, we essentially approximated the Q-value function by Monte-Carlo estimation. In every iteration, we sampled multiple independent trajectory to implicitly do a Q-value estimation. Thus, as the policy changes, a new gradient is estimated independently of past estimates. In the actor-critic method, we use Q-function approximation in the gradient estimate. The actor-critic algorithm simultaneously/alternatively updates the Q-function approximation as well as policy parameters, as it sees more samples. Over iterations, as policy changes, the Q-function approximations also improves with more sample observations.

Actor-Critic methods are thus a version of policy iteration with function approximation.

- The "Critic" estimates the value function. This could be the action-value (the Q-value) or state-value (the V-value).
- The "Actor" updates the policy distribution in the direction suggested by the Critic (such as with policy gradients).

1 Simple example of Actor-Critic method

In this section, we consider the critic to estimate the state-action value function Q^π using temporal difference. We use a parametric representation for both π and Q^π. Let θ be the parameters of the policy (i.e., π_θ) and ω be the parameters of the Q-function (i.e., Q_ω). For each sample (s_t, a_t, r_t, s_{t+1}), we update the estimated value function as

$$\begin{aligned}\omega_{t+1} &= \omega_t + \alpha_\omega \nabla_\omega (Q_{\omega_t}(s_t,a_t) - r_t - \gamma \max_{a'} Q_{\omega_t}(s_{t+1},a'))^2 \\ &= \omega_t + \alpha_\omega \delta_t \nabla_\omega Q_{\omega_t}(s_t,a_t)\end{aligned}$$

where $\delta_t := Q_{\omega_t}(s_t,a_t) - r_t - \gamma \max_{a'} Q_{\omega_t}(s_{t+1},a')$ is the temporal-difference error (TD error). Note that in the derivation of the gradient, we ignored the dependence w.r.t. ω in the target value. Gradient TD is called semi-gradient since it does not use a real gradient.

Given a function approximation of the Q-function, we can approximate the gradient using a single sample (s_t, a_t, r_t, s_{t+1}) as $\hat{g} = \nabla_\theta \log \pi_{\theta_t}(s_t,s_t) Q_{\omega_{t+1}}(s_t,a_t)$. The policy is thus updated as follows:

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \log \pi_{\theta_t}(s_t,s_t) Q_{\omega_{t+1}}(s_t,a_t)$$

The algorithm is reported in Alg. 1. We can do better than this, see A2C.

2 Advantage Actor Critic (A2C) [2]

We have seen in class that we can write the policy gradient theorem with a baseline b^π

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi}\left[\nabla_\theta \log \pi_\theta(a|s)\left(Q^\pi(s,a) - b^\pi(s)\right)\right]$$

This equality holds for any baseline $b^\pi(s)$ that depends only on the state. We have seen that we can chose $V^\pi(s)$ to be the baseline, leading to the following definition of policy gradient theorem

$$\begin{aligned}\nabla_\theta J(\pi_\theta) &= \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi}\left[\nabla_\theta \log \pi_\theta(a|s)\left(Q^\pi(s,a) - V^\pi(s)\right)\right] \\ &= \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi}\left[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a)\right]\end{aligned}$$

where $A^\pi(s,a)$ is called *advantage function* and it is defined as

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

The advantage function denotes the advantage or disadvantage of performing action a in state s. Instead of estimating the Q-function we can directly estimate the advantage function.

Consider the TD error of a policy π

$$\delta^\pi = r(s,a) + \gamma V^\pi(s') - V^\pi(s)$$

it is easy to show that it is an unbiased estimate (conditioned on s and a) of the advantage function $A^\pi(s,a)$

$$\mathbb{E}[\delta^\pi|s,a] = \mathbb{E}_{s'}[r(s,a) + \gamma V^\pi(s')|s,a] - V^\pi(s) = Q^\pi(s,a) - V^\pi(s)$$

This shows that we can estimate the advantage function by computing the TD error. Note that to compute the TD error we just need to approximate the value function.

Similarly to before, we use θ to denote the parameters of the policy (i.e., π_θ) and ω for the parameters of the V-function (i.e., V_ω). We can thus use gradient TD to update the V-function and the policy gradient theorem to update the policy. For each sample (s_t, a_t, r_t, s_{t+1}) we update

$$\begin{aligned}\delta_t &= r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t) \\ \omega_{t+1} &= \omega_t + \beta\delta_t \nabla_\omega V_{\omega_t}(s_t) \\ \theta_{t+1} &= \theta_t + \alpha \underbrace{\delta_t}_{\approx A^\pi(s_t,a_t)} \nabla_\theta \log \pi_{\theta_t}(a_t|s_t)\end{aligned}$$

The algorithm is reported in Alg. 2.

The main issue with this approach is that the policy and V-function are updated at each step. This is inefficient for the modern ML tools (e.g., deep learning). We would like to use batch updates.

Batch A2C and practical implementation. The idea is simple, use batch TD. Collect m trajectories $h^i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, \ldots, s_{T_i})$. For each state s_t^i compute the target using:

$$y_t^i = \sum_{k=t}^{T_i} \gamma^{k-t} r_k + \gamma^{T_i - t + 1} V_\omega(s_{T_i}) \cdot 1(s_{T_i} \text{ is terminal})$$

Algorithm 1: Actor-Critic with TD

for $t = 1, \ldots, T$ **do**

$a_t \sim \pi^\theta(s_t, \cdot)$ and observe r_t and s_{t+1}

Compute temporal difference

$$\delta_t = r_t + \gamma Q_\omega(s_{t+1}, a_{t+1}) - Q_\omega(s_t, a_t)$$

Update Q estimate

$$\omega = \omega + \beta\delta_t \nabla_\omega q_\omega(s_t, a_t)$$

Update policy

$$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) Q_\omega(s_t, a_t)$$

end

Algorithm 2: A2C online

for $t = 1, \dots, T$ **do**

$a_t \sim \pi_\theta(\cdot|s_t)$ and observe r_t and s_{t+1}

Compute temporal difference

$$\delta_t = r_t + \gamma v_\nu(s_{t+1}) - v_\nu(s_t)$$

Update V estimate

$$\omega = \omega + \beta \delta_t \nabla_\omega v_\omega(s_t)$$

Update policy

$$\theta = \theta + \alpha \delta_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

end

Then, we can compute the new estimate of the V-function by solving the following regression problem

$$\omega' = \arg\min_\omega \sum_{i=1}^{m} \sum_{t=1}^{T_i} (V_\omega(s_t^i) - y_t^i)^2$$

Now that we have a new estimate of the V-function, we can update the policy. For each state s_t^i, we compute the TD error as

$$\delta_t^i = y_t^i - V_\omega(s_t^i)$$

and we update the policy as

$$\theta' = \theta + \alpha \sum_{i=1}^{m} \sum_{t=1}^{T_i} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \delta_t^i$$

Entropy regularization. In order to enforce exploration, A2C usually uses an entropy regularization term in the update of the policy. The regularization term is $\Omega(\pi(\cdot|s)) = \sum_a \pi(a|s) \log \pi(a|s)$. The complete code of batch A2C is reported in Alg. 3. Normally, the regularization parameter λ_e is set to 0.001.

Algorithm 3: Batch A2C with entropy regularization

for $n = 1, \dots, N$ **do**

Collect m trajectories by using policy π_{θ_n}

$$\forall i \in [m], \quad h^i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, \dots, s_{T_i})$$

Compute $\forall i, t$

$$y_t^i = \sum_{k=t}^{T_i} \gamma^{k-t} r_k + \gamma^{T_i - t + 1} V_\omega(s_{T_i}) \cdot 1(s_{T_i} \text{ is terminal})$$

$$\delta_t^i = y_t^i - V_\omega(s_t^i)$$

Compute new estimate of V

$$\omega_{n+1} = \arg\min_\omega \sum_{i=1}^{m} \sum_{t=1}^{T_i} (V_\omega(s_t^i) - y_t^i)^2$$

Update policy

$$\theta_{n+1} = \theta_n + \alpha \sum_{i=1}^{m} \sum_{t=1}^{T_i} \Big[\nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \delta_t^i + \lambda_e \Omega(\pi_\theta(\cdot|s_t^i)) \Big]$$

end

References

[1] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[2] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org, 2016.