Lecture 13: Policy Optimization III: Variants of Actor-Critic and Code

Bolei Zhou

The Chinese University of Hong Kong

bzhou@ie.cuhk.edu.hk

March 11, 2020

Today's Plan

1. Review on policy gradient
2. Actor critic and advantage actor critic
3. Natural policy gradient
4. Sample code

Review on Policy Gradient

1. policy optimization is to maximize $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$
2. The gradient is

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) = \mathbb{E}_\tau[R(\tau)\nabla_\theta \log P(\tau;\theta)]$$

   1. After decomposing τ we derived that

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau\Big[\Big(\sum_{t=0}^{T-1} r_t\Big)\Big(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\Big)\Big]$$

   2. After applying the causality, we have

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau\Big[\sum_{t=0}^{T-1} G_t \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)\Big]$$

Reducing Variance Using a Critic

1. The update is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{T-1} G_t \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)\Big]$$

2. In practice, G_t is a sample from Monte Carlo policy gradient, which is the unbiased but noisy estimate of $Q^{\pi_\theta}(s_t, a_t)$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{T-1} Q^{\pi_\theta}(s,a) \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)\Big]$$

3. Instead we can use a critic to estimate the action-value function,

$$Q_{\mathbf{w}}(s,a) \approx Q^{\pi_\theta}(s,a)$$

4. Then the update becomes

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{T-1} Q_{\mathbf{w}}(s_t,a_t) \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)\Big]$$

Compatible Function Approximation

Theorem (Compatible Function Approximation Theorem)

If the following two conditions are satisfied:

1. Value function approximator is compatible to the policy

$$\nabla_w Q_w(s, a) = \nabla_\theta \log \pi_\theta(s, a)$$

2. Value function parameters w minimize the mean-squared error

$$\epsilon(w) = E_{\pi_\theta}[(Q^{\pi_\theta}(s, a) - Q_w(s, a))^2]$$

then the policy gradient is exact

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[Q_w(s, a)\nabla_\theta \log \pi_\theta(s, a)]$$

RS Sutton, et al. Policy gradient methods for reinforcement learning with function approximation. NIPS'01

Compatible Function Approximation

1. Given a parameterized policy, compatible function approximation theorem can be used to derive an appropriate form of the value-function parameterization.
    1. For example, consider a policy as Gibbs distribution where ϕ_{sa} is the feature vector for state-action pair s,a

$$\pi_\theta(s,a) = \frac{e^{\theta^T \phi_{sa}}}{\sum_b e^{\theta^T \phi_{sb}}} \tag{1}$$

    2. Then meeting the compatibility condition requires

$$\nabla_w Q_w(s,a) = \nabla_\theta \log \pi_\theta(s,a) = \phi_{sa} - \sum_b \pi_\theta(s,b)\phi_{sb}$$

    3. so the compatible parameterization of $Q_w(s,a)$ is

$$Q_w(s,a) = w^T \left(\phi_{sa} - \sum_b \pi_\theta(s,b)\phi_{sb}\right) = w^T \nabla_\theta \log \pi_\theta(s,a)$$

Action-Value Actor-Critic Algorithm

1. Using a linear value function approximation: $Q_{\mathbf{w}}(s,a) = \psi(s,a)^T\mathbf{w}$
    1. Critic: update $\mathbf{w}$ by a linear TD(0)
    2. Actor: update θ by policy gradient

Algorithm 1 Simple QAC

```
1: for each step do
2:     generate sample $s, a, r, s', a'$ following $\pi_\theta$
3:     $\delta = r + \gamma Q_{\mathbf{w}}(s', a') - Q_{\mathbf{w}}(s, a)$        #TD error
4:     $\mathbf{w} \leftarrow \mathbf{w} + \beta\delta\psi(s, a)$
5:     $\theta \leftarrow \theta + \alpha\nabla_\theta \log \pi_\theta(s, a) Q_{\mathbf{w}}(s, a)$
6: end for
```

Reducing the Variance of Actor-Critic by a Baseline

1. Recall Q-function / state-action-value function:

$$Q^{\pi,\gamma}(s,a) = \mathbb{E}_\pi[r_1 + \gamma r_2 + ...|s_1 = s, a_1 = a]$$

2. State value function can serve as a great baseline

$$\begin{aligned} V^\pi(s) =& \mathbb{E}_\pi[r_1 + \gamma r_2 + ...|s_1 = s] \\ =& \mathbb{E}_{a\sim\pi}[Q^\pi(s,a)] \end{aligned}$$

3. Advantage function: combing Q with baseline V

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

4. Then the policy gradient becomes:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)]$$

Advantage Actor-Critic

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)]$$

1. Critic should estimate the advantage function
2. Using two function approximators and two sets of parameter vectors,

$$V_{\mathbf{v}}(s) \approx V^\pi(s)$$
$$Q_{\mathbf{w}}(s,a) \approx Q^\pi(s,a)$$

3. Updating both parameters by TD learning or MC

Advantage Actor-Critic

1. For the true value function $V^{\pi_\theta}(s)$, the TD error δ^{π_θ}

$$\delta^{\pi_\theta} = r(s,a) + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

is an unbiased estimate of the advantage function

$$\begin{aligned}\mathbb{E}_{\pi_\theta}[\delta^{\pi_\theta}|s,a] =& \mathbb{E}_{\pi_\theta}[r + \gamma V^{\pi_\theta}(s')|s,a] - V^{\pi_\theta}(s)\\ =& Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)\\ =& A^{\pi_\theta}(s,a)\end{aligned}$$

2. So we can use the TD error to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\delta^{\pi_\theta}]$$

3. It requires only one set of critic parameter κ that is estimated by TD

$$\delta_v = r + \gamma V_\kappa(s') - V_\kappa(s)$$

Critic at Different Time-Scales

1. Critic can estimate linear value function $V_\kappa(s) = \psi(s)^T\kappa$ from many targets as follows
2. For MC, the update is

$$\Delta\kappa = \alpha(G_t - V_\kappa(s))\psi(s)$$

3. For TD(0), the update is

$$\Delta\kappa = \alpha(r + \gamma V_\kappa(s') - V_\kappa(s))\psi(s)$$

4. For k-step return, the update is

$$\Delta\kappa = \alpha(\sum_{i=0}^{k}\gamma^i r_{t+i} + \gamma^k V_\kappa(s_{t+k}) - V_\kappa(s_t))\psi(s)$$

Actors at Different Time-Scales

1. The policy gradient can also be estimated at many time-scales

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) A^{\pi_\theta}(s, a)]$$

2. Monte-Carlo Actor-Critic policy gradient uses error from complete return

$$\Delta\theta = \alpha(G_t - V_\kappa(s_t))\nabla_\theta \log \pi_\theta(s_t, a_t)$$

3. TD Difference Actor-Critic policy gradient uses TD error

$$\Delta\theta = \alpha(r + \gamma V_\kappa(s_{t+1}) - V_\kappa(s_t))\nabla_\theta \log \pi_\theta(s_t, a_t)$$

4. k-step return Actor-Critic policy gradient uses the k-step return error

$$\Delta\theta = \alpha(\sum_{i=0}^{k} \gamma^i r_{t+i} + \gamma^k V_\kappa(s_{t+k}) - V_\kappa(s_t))\nabla_\theta \log \pi_\theta(s_t, a_t)$$

Summary of Policy Gradient Algorithms

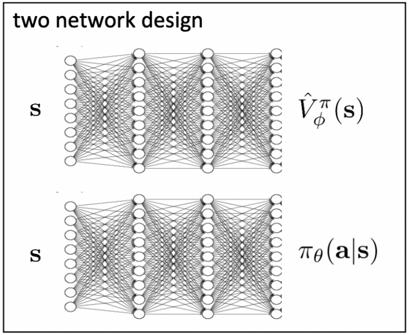
1. The policy gradient has many forms

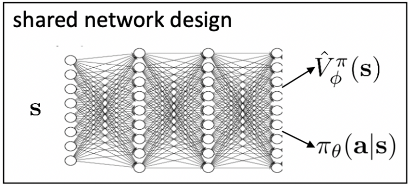
$$\begin{aligned}\nabla_\theta J(\theta) =& \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) G_t] \text{ — REINFORCE}\\ =& \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\mathbf{w}}(s,a)] \text{ — Q Actor-Critic}\\ =& \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^{\mathbf{w}}(s,a)]\text{— Advantage Actor-Critic}\\ =& \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \delta]\text{— TD Actor-Critic}\end{aligned}$$

2. Critic uses policy evaluation (e.g. MC or TD learning) to estimate $Q^\pi(s,a)$, $A^\pi(s,a)$, or $V^\pi(s,a)$

Implementing Actor-Critic Function Approximators

1. We can have two separate functions to approximate value function and policy function, or use a shared network design (feature extraction is shared but output two heads), as below:


two network design
$\mathbf{s}$
$\hat{V}^{\pi}_{\phi}(\mathbf{s})$
$\mathbf{s}$
$\pi_\theta(\mathbf{a}|\mathbf{s})$


shared network design
$\mathbf{s}$
$\hat{V}^{\pi}_{\phi}(\mathbf{s})$
$\pi_\theta(\mathbf{a}|\mathbf{s})$

Natural Policy Gradient

1. Policy gradient is the steepest ascend in **parameter space** with Euclidean metric:

$$d^* = \nabla_\theta J(\theta) = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg\max J(\theta + d), s.t. ||d|| \leq \epsilon$$

   1. Drawback: it is sensitive to the parameterization of the policy function

2. Steepest ascend in **distribution space (policy output)** with KL-divergence as constraint

$$d^* = \arg\max J(\theta + d), s.t. KL[\pi_\theta || \pi_{\theta+d}] = c \tag{2}$$

   1. Fixing KL-divergence as a constant c make sure we move along the distribution space with constant speed, regardless the curvature (thus robust to the reparametrization of the model as we only care about the distribution induced by the parameter)

KL-Divergence as Metric for Two Distributions

1. KL-divergence measure the "closeness" of two distributions.

$$KL[\pi_\theta||\pi_{\theta'}] = E_{\pi_\theta}[\log \pi_\theta] - E_{\pi_\theta}[\log \pi_{\theta'}]$$

2. Although as KL-divergence is non-symmetric and thus not a true metric, we can use it anyway. This is because as d goes to zero, KL-divergence is asymptotically symmetric. So, within a local neighbourhood, KL-divergence is approximately symmetric
3. We can prove that the second-order Taylor expansion of KL-divergence is

$$KL[\pi_\theta||\pi_{\theta+d}] \approx \frac{1}{2} d^T F d$$

where F is the **Fisher Information Matrix** as second order derivative of KL divergence $E_{\pi_\theta}[\nabla \log \pi_\theta \nabla \log \pi_\theta^T]$

Natural Policy Gradient

$$d^* = \arg\max J(\theta + d), s.t. KL[\pi_\theta || \pi_{\theta+d}] = c$$

1. Write the above as Lagrangian form, with the $J(\theta + d)$ approximated by its first order Tyler expansion and the constraint KL-divergence approximated by its second order Tyler expansion

$$\begin{aligned} d^* &= \arg\max_d J(\theta + d) - \lambda(KL[\pi_\theta || \pi_{\theta+d}] - c) \\ &\approx \arg\max_d J(\theta) + \nabla_\theta J(\theta)^T d - \frac{1}{2}\lambda d^T F d + \lambda c \end{aligned}$$

2. To solve this maximization we set its derivative wrt. d to zero, then we have the natural policy gradient: $d = \frac{1}{\lambda} F^{-1} \nabla_\theta J(\theta)$

Natural Policy Gradient

1. Natural policy gradient is a **second-order** optimization that is more accurate and works regardless of how the model is parameterized (model invariant).

$$\theta_{t+1} = \theta_t + \alpha F^{-1} \nabla_\theta J(\theta)$$

    1. where $F = E_{\pi_\theta(s,a)}[\nabla \log \pi_\theta(s,a) \nabla \log \pi_\theta(s,a)^T]$ is the Fisher information matrix, also as the second-order derivative of the KL-divergence
    2. F measures the curvature of the policy(distribution) relative to the model parameter θ

2. Natural policy gradient produces the same policy change **regardless of the model parameterization**.
3. Idea behind TRPO. A detailed read on natural gradient: `https://wiseodd.github.io/techblog/2018/03/14/natural-gradient/`

Code example

1. REINFORCE code on CartPole: https://github.com/cuhkrlcourse/RLexample/blob/master/policygradient/reinforce.py
2. Policy Gradient on Pong : https://github.com/cuhkrlcourse/RLexample/blob/master/policygradient/pg-pong-pytorch.py
3. Policy Gradient with Baseline on Pong: https://github.com/cuhkrlcourse/RLexample/blob/master/policygradient/pgb-pong-pytorch.py
4. Actor Critic on Pong: https://github.com/cuhkrlcourse/RLexample/blob/master/policygradient/ac-pong-pytorch.py

Next Week: State of the Arts on Policy Optimization

1. Policy Graident→TRPO→ACKTR→PPO
    1. **TRPO**: Trust region policy optimization. Schulman, L., Moritz, Jordan, Abbeel. 2015
    2. **ACKTR**: Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba. 2017
    3. **PPO**: Proximal policy optimization algorithms. Schulman, Wolski, Dhariwal, Radford, Klimov. 2017
2. Q-learning→DDPG→TD3→SAC
    1. **DDPG**: Deterministic Policy Gradient Algorithms, Silver et al. 2014
    2. **TD3**: Addressing Function Approximation Error in Actor-Critic Methods, Fujimoto et al. 2018
    3. **SAC**: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, Haarnoja et al. 2018