Trust Region Policy Optimization and Proximal Policy Optimization

Sunmook Choi felixchoi@korea.ac.kr

September 4, 2023

1 Trust Region Policy Optimization

2 Proximal Policy Optimization

2.1 Limitation of TRPO

2.2 Clipped Surrogate Objective

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The main objective that we want to maximize is the following:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min\big(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big] \tag{2.1}$$

where epsilon is a hyperparameter, say $\epsilon = 0.2$. In the objective, the term $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ clips the ratio, that is,

$$\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) = \begin{cases} 1+\epsilon, & \text{if } r_t(\theta) > 1+\epsilon; \\ r_t(\theta), & \text{if } 1-\epsilon \le r_t(\theta) \le 1+\epsilon; \\ 1-\epsilon, & \text{if } r_t(\theta) < 1-\epsilon. \end{cases}$$

This clipping provides the same motivation as the trust region in TRPO, trying not to change the policy too much at each policy update. Assume that the advantage-function estimator $\hat{A}_t$ is positive. This also implies that the action at timestep t is approximated to be better than the expected value. Hence, we need to encourage the chosen action at timestep t. Therefore, since we are maximizing the objective, we would like to make the ratio as large as possible. However, if the policy changes too much, then it may lead to a disaster, and hence, it is clipped at $1+\epsilon$. Notice that the minimum operation keeps the target unaffected by the value $1-\epsilon$. On the other hand, assume that the advantage-function estimator $\hat{A}_t$ is negative. This would imply that the action at timestep t is approximated to be worse than the expected value, so that the action should be discouraged by reducing the probability ratio. Since $\hat{A}_t < 0$, notice that reducing the probability ratio is equivalent to maximizing the objective. Likewise, in order to protect the network being disastrous, it is now clipped at $1-\epsilon$. In this case, notice that the minimum operation keeps the target unaffected by the value $1+\epsilon$.

With this observation, the objective function Eq. 2.1 can be restated as below:

$$L^{CLIP}(\theta) = \begin{cases} \hat{\mathbb{E}}_t\big[\min\big(r_t(\theta),\ (1+\epsilon)\big)\hat{A}_t\big] & \text{if } \hat{A}_t \ge 0 \\ \hat{\mathbb{E}}_t\big[\min\big(r_t(\theta),\ (1-\epsilon)\big)\hat{A}_t\big] & \text{if } \hat{A}_t < 0 \end{cases}. \tag{2.2}$$

This expression might give you a better intuition to understand the objective.

2.2.1 Adaptive KL Penalty Coefficient

The paper [3] provides an alternative to the clipped surrogate objective, or in addition to it. The approach uses a penalty on KL divergence and adapts the penalty coefficient. The instantiation of this algorithm is as follows:

- Using several epochs of minibatch SGD, optimize the KL-penalized objective

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[r_t(\theta)\hat{A}_t - \beta \, \mathrm{KL}\big(\pi_{\theta_{\text{old}}}(\cdot|s_t) \| \pi_\theta(\cdot|s_t)\big) \right] \tag{2.3}$$

- Compute $d = \hat{\mathbb{E}}_t \left[\mathrm{KL}\big(\pi_{\theta_{\text{old}}}(\cdot|s_t) \| \pi_\theta(\cdot|s_t)\big) \right]$
  - If $d < d_{\text{targ}}/1.5$, then $\beta \leftarrow \beta/2$.
  - If $d > d_{\text{targ}} \times 1.5$, then $\beta \leftarrow \beta \times 2$.

From the steps above, we consider the role of the constant d_{targ}. In the paper, the authors describe that the constant d_{targ} is some target value of the KL divergence each policy update, that is, the target of the amount of changes at each policy update. The authors tried to make the amount of change $d = \hat{\mathbb{E}}_t \left[\mathrm{KL}\big(\pi_{\theta_{\text{old}}}(\cdot|s_t) \| \pi_\theta(\cdot|s_t)\big) \right]$ close to d_{targ} by adapting the coefficient β. In the case of $d < d_{\text{targ}}/1.5$, the change in policy update should be encouraged, so that the coefficient β is decreased. In the other case of $d > d_{\text{targ}} \times 1.5$, the change in policy update should be discouraged, so that the coefficient β is increased. The authors said that the coefficient β in the algorithm quickly adjusts. However, they found out that the KL penalty performed worse than the clipped surrogate objective.

2.3 Algorithm

First, a variance-reduced advantage-function estimator $\hat{A}_t$ should be defined. One style of estimators is the one that is popularized in [4], which runs the policy for fixed timesteps T (where T is much less than the episode length). The estimator is

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{2.4}$$

where t specifies the time index in $[0, T]$, within a given length-T trajectory segment. Generalizing this choice, we can use a truncated version of generalized advantage estimation, which reduces to Eq. 2.4 when $\lambda = 1$:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{2.5}$$

$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{2.6}$$

Remark 1 (Generalized Advantage Estimation)**.** Generalized advantage estimation is explored in the paper [5]. With the definition $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, consider taking the sum of k of these δ terms, which is denoted by $\hat{A}_t^{(k)}$:

$$\hat{A}_t^{(1)} := \delta_t \qquad\qquad = -V(s_t) + r_t + \gamma V(s_{t+1}) \tag{2.7}$$

$$\hat{A}_t^{(2)} := \delta_t + \gamma\delta_{t+1} \qquad\qquad = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \tag{2.8}$$

$$\hat{A}_t^{(3)} := \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3}) \tag{2.9}$$

$$\vdots$$

$$\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+1} = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \tag{2.10}$$

Taking $k \to \infty$, we get

$$\hat{A}_t^{(\infty)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+1} = -V(s_t) + \sum_{l=0}^{\infty} \gamma^l r_{t+l}, \tag{2.11}$$

which is simply the empirical returns minus the value function baseline. The generalized advantage estimator GAE(γ, λ) is defined as the exponentially-weighted average of these k-step estimators:

$$\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \cdots\right) && (2.12)\\
&= (1-\lambda)\left(\delta_t + \lambda(\delta_t + \gamma\delta_{t+1}) + \lambda^2(\delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2}) + \cdots\right) && (2.13)\\
&= (1-\lambda)\Big(\delta_t(1 + \lambda + \lambda^2 + \cdots) + \gamma\delta_{t+1}(\lambda + \lambda^2 + \lambda^3 + \cdots) && (2.14)\\
&\qquad + \gamma^2\delta_{t+2}(\lambda^2 + \lambda^3 + \lambda^4 + \cdots) + \cdots\Big) && (2.15)\\
&= (1-\lambda)\left(\delta_t\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}\left(\frac{\lambda^2}{1-\lambda}\right) + \cdots\right) && (2.16)\\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} && (2.17)
\end{aligned}$$

This construction is analogous to the one used to define TD(λ) [1], however TD(λ) is an estimator of the value function, whereas here we are estimating the advantage function. There are two notable special cases of this formula:

$$\begin{aligned}
\mathrm{GAE}(\gamma, 0): \quad & \hat{A}_t := \delta_t && = r_t + \gamma V(s_{t+1}) - V(s_t) && (2.18)\\
\mathrm{GAE}(\gamma, 1): \quad & \hat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} && = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) && (2.19)
\end{aligned}$$

Now we construct the final objective. When using a neural network for a state-value function approximator, we must use a loss function that combines the policy surrogate and a value function error term. Furthermore, we may use an entropy bonus to ensure sufficient exploration. The following objective is maximized each iteration:

$$L^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\Big[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\Big], \tag{2.20}$$

where c_1, c_2 are coefficients, S denotes an entropy, and L_t^{VF} is a squared-error loss $(V_\theta(s_t) - V_t^{targ})^2$. A proximal policy optimization (PPO) algorithm that uses fixed-length trajectory segments is shown Algorithm 2.1.

References

[1] Andrew Barto and Richard S. Sutton, *Reinforcement Learning: An Introduction* (2nd ed.), The MIT Press, 2018.

[2] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, *Trust Region Policy Optimization*, ICML 2015.

Algorithm 2.1 PPO, Actor-Critic Style

for iteration= $1, 2, \ldots$ **do**
  for actor= $1, 2, \ldots, N$ **do** ▷ N Parallel actors
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L w.r.t. θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv:1707.06347, 2017.

[4] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, *Asynchronous Methods for Deep Reinforcement Learning*, ICML 2016

[5] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, *High-Dimensional Continuous Control Using Generalized Advantage Estimation*, ICLR 2016

[6] OpenAI Spinning Up, `https://spinningup.openai.com/en/latest/algorithms/ppo.html`