Proximal Policy Optimization

Sunmook Choi

Dept. of Mathematics
Korea University

Proximal Policy Optimization

TRPO Goal: $\max_{\theta} \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right]$ subject to $\hat{\mathbb{E}}_t \left[\text{KL}(\pi_{\theta_{\text{old}}}(\cdot|s_t) \,\|\, \pi_\theta(\cdot|s_t)) \right] \leq \delta$

However, TRPO is too complicated to implement and too heavy to compute:

* TRPO algorithm contains 2nd-order optimization (natural policy gradient).

The motivation of PPO is the same as that of TRPO.

- How can we update the policy as big as possible?
- But we want to update not too much so that we can prevent an accidental disaster.
- At the same time, we want to make it easy to implement and to have less computation.

Proximal Policy Optimization, OpenAI 2017

Clipped Surrogate Objective

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t,\ clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right]$$

- $\hat{A}_t$: an advantage-function estimator

Notice that, for $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, the update equation is equal to the following:

- For $\hat{A}_t > 0$, $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ (1+\epsilon)\hat{A}_t\right)\right]$: *Encouraging* the chosen action a_t.
- For $\hat{A}_t < 0$, $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ (1-\epsilon)\hat{A}_t\right)\right]$: *Discouraging* the chosen action a_t.
- The clipping value $\epsilon > 0$ is a hyperparameter.

Adaptive KL Penalty Coefficient

The authors provide 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.

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

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

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

The hyperparameter d_{targ} denotes the target of the amount of changes at each policy update.

Advantage Function Estimator

- One style of estimating the advantage function is to run the policy for fixed T timesteps:

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

which is used in the A3C paper.

- Another style is to use a truncated version of generalized advantage estimation (GAE), a generalization of the choice above, which reduces to the one above when $\lambda = 1$:

$$\begin{aligned} &\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \\ &\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned}$$

Recall: TD(λ) (오승상 교수님 강화학습 강의자료 p.62)

- n-step return
  - $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$
  - $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$
- $G_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$ for $\lambda \in [0, 1]$.
  - the exponentially-weighted average of n-step returns $G_t^{(n)}$
  - TD(λ) is a method of updating value function as follows:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t^\lambda - V(S_t)]$$

  - If $\lambda = 0$, then it is equal to the original TD update equation.

Generalized Advantage Estimation

With the definition $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$,

* $\hat{A}_t^{(1)} = \delta_t = -V(s_t) + r_t + \gamma V(s_{t+1})$
* $\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+1} = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})$
* $\hat{A}_t^{(\infty)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+1} = -V(s_t) + \sum_{l=0}^{\infty} \gamma^l r_{t+l} \quad \longleftarrow \quad$ an advantage-function estimator.

$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$: the generalized advantage estimator

* the exponentially-weighted average of these k-step estimators
* $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \cdots\right) = \cdots = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$

J. Schulman et. al., *High-Dimensional Continuous Control Using Generalized Advantage Estimation*, ICLR 2016

Practical Implementation

Final objective: $L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$

- $L_t^{VF}(\theta) = (V_\theta(s_t) - V_t^{targ})^2$: to learn the state-value function
  - $V_t^{targ} = r_{t+1} + \gamma V_\theta(s_{t+1})$: TD-target
  - $V_t^{targ} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-1} r_T = G_t$: MC-target
- $S[\pi_\theta](s_t) = -\sum_{a \in \mathcal{A}} \pi_\theta(a|s_t) \log \pi_\theta(a|s_t)$: entropy bonus to ensure exploration