Proximal Policy Optimization

Sunmook Choi

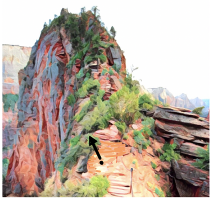
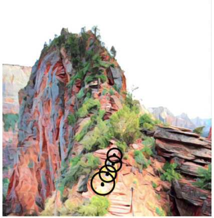
Dept. of Mathematics
Korea University

Trust Region Policy Optimization (TRPO)

- TRPO is an actor-critic algorithm that updates policy on a trust region, which is a Kullback-Leibler divergence constraint in the policy space.
- Policy updates on the trust region guarantee monotonic improvement of the expected return.

Intuition

- While updating policy with gradient ascent, we find the steepest direction and step forward. However, if the stepsize is too big, then it may occur a disaster.
- The search space is limited into a trust region, which ensures to find a better policy.
- By continuing iterations,
  it eventually reaches a local or global optimum.

J. Schulman et al., *Trust Region Policy Optimization*, ICML 2015

TRPO Optimization

- TRPO tries to maximize the expected return: $\eta(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}]$
- Instead, it maximizes a surrogate objective with a series of theoretically justified approximations.

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

  * $\hat{A}_t$: an advantage-function estimator
  * If $\hat{A}_t > 0$, then we *encourage* the chosen action a_t.
  * If $\hat{A}_t < 0$, then we *discourage* the chosen action a_t.
- TRPO updates parameter θ within the KL constraint.
  That is, it maximizes the objective while restricting the amount of change in the policy.
- However, TRPO is too complicated to implement and too heavy to compute.
  * Finding a trust region contains 2nd-order optimization (natural policy gradient).

Proximal Policy Optimization

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 it not too much so that we can prevent an accidental disaster.

In addition, we want to make it easy to implement and to have less computation.

Clipped Surrogate Objective

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

Let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, then the update equation is as below. ($\epsilon > 0$: clipping hyperparameter)

- If $\hat{A}_t > 0$, $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t,\ (1+\epsilon) \hat{A}_t \right) \right]$: we *encourage* the chosen action a_t.
- If $\hat{A}_t < 0$, $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t,\ (1-\epsilon) \hat{A}_t \right) \right]$: we *discourage* the chosen action a_t.

J. Schulman et al., *Proximal Policy Optimization*, OpenAI 2017

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 $\lambda = 1$:

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

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

* $\hat{A}_t^{(1)} = \delta_t = -V(s_t) + r_t + \gamma V(s_{t+1})$
* $\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l} = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})$
* $\hat{A}_t^{(\infty)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = -V(s_t) + \sum_{l=0}^{\infty} \gamma^l r_{t+l} \quad \longleftarrow$ 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: $\max_{\theta} L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$

PPO is practically implemented as an actor-critic algorithm.

- $L_t^{VF}(\theta) = (V_\theta(s_t) - V_t^{targ})^2$: 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