近端策略优化算法

卢艳峰

September 2, 2023

前言

- 学习近端策略优化算法（**proximal policy optimization，PPO**）
- 由于原始训练语料中文比重不高，**bigscience/bloom-1b1** 在 **RLHF** 的第一步表现不佳，正在寻找其他模型替代。

PPO

- **PPO** 是一种带自益的策略梯度算法：一方面，用一个含参函数近似价值函数，然后利用这个价值函数的近似值来估计回报值；另一方面，利用估计得到的回报值估计策略梯度，进而更新策略参数。这种算法被称为执行者 / 评论者算法（**actor-critic algorithm**）。

PPO

- **PPO** 是 **TRPO** 算法的改进版，实现更加简洁，而且更快。**PPO** 使用截断的方法在目标函数中进行限制，以保证新的参数和旧的参数的差距不会太大。
- **TRPO** 和 **PPO** 都属于在线策略学习算法，即优化目标中包含重要性采样的过程，但其时只是用到了上一轮策略的数据，而不是过去所有策略的数据。
- **PPO** 和 **TRPO** 的作者是同一人。

PPO

Policy gradient methods work by computing an estimator of the policy gradient and plugging it into a stochastic gradient ascent algorithm. The most commonly used gradient estimator has the form

$$\hat{g} = \hat{\mathbb{E}}_t \Big[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \Big] \tag{1}$$

where π_θ is a stochastic policy and $\hat{A}_t$ is an estimator of the advantage function at timestep t. Here, the expectation $\hat{\mathbb{E}}_t[\ldots]$ indicates the empirical average over a finite batch of samples, in an algorithm that alternates between sampling and optimization. Implementations that use automatic

PPO

One style of policy gradient implementation, popularized in [Mni+16] and well-suited for use with recurrent neural networks, runs the policy for T timesteps (where T is much less than the episode length), and uses the collected samples for an update. This style requires an advantage estimator that does not look beyond timestep T. The estimator used by [Mni+16] is

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{10}$$

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 Equation (10) when $\lambda = 1$:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{11}$$

$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{12}$$

A proximal policy optimization (PPO) algorithm that uses fixed-length trajectory segments is shown below. Each iteration, each of N (parallel) actors collect T timesteps of data. Then we

PPO

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$, so $r(\theta_{\text{old}}) = 1$. TRPO maximizes a "surrogate" objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right]. \tag{6}$$

The superscript CPI refers to conservative policy iteration [KL02], where this objective was proposed. Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update; hence, we now consider how to modify the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1.

The main objective we propose is the following:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{7}$$

where epsilon is a hyperparameter, say, $\epsilon = 0.2$. The motivation for this objective is as follows. The

PPO

finite-horizon estimators in [Mni+16]. If using a neural network architecture that shares parameters between the policy and value function, we must use a loss function that combines the policy surrogate and a value function error term. This objective can further be augmented by adding an entropy bonus to ensure sufficient exploration, as suggested in past work [Wil92; Mni+16]. Combining these terms, we obtain the following objective, which is (approximately) maximized each iteration:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right], \tag{9}$$

where c_1, c_2 are coefficients, and S denotes an entropy bonus, and L_t^{VF} is a squared-error loss $(V_\theta(s_t) - V_t^{\mathrm{targ}})^2$.

PPO

shown below. Each iteration, each of N (parallel) actors collect T timesteps of data. Then we construct the surrogate loss on these NT timesteps of data, and optimize it with minibatch SGD (or usually for better performance, Adam [KB14]), for K epochs.

Algorithm 1 PPO, Actor-Critic Style

for iteration$=1, 2, \ldots$ **do**
  for actor$=1, 2, \ldots, N$ **do**
    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 wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

PPO 实践

- 离散动作空间：**http://10.4.7.189/auto_implementation/ppo-discrete.html**
- 连续动作空间：**http://10.4.7.189/auto_implementation/ppo-continuous.html**

计划

- 阅读 **GLM** 论文，加深对大语言模型的认识。
- 用一个中文预训练模型跑完上述流程。
- 将中文通用的问答数据和师妹用 **ChatGPT** 生成的数据混合作为我们的训练集。
- 用 **ChatGLM** 生成的答案作为 **rejected** 答案。

参考

- **bigscience/bloom-1b1：https://huggingface.co/bigscience/bloom-1b1**
- **PPO：https://arxiv.org/abs/1707.06347**
- 肖智清，强化学习原理与**Python**实现
- **thu-ml/tianshou：https://github.com/thu-ml/tianshou**
- 动手学强化学习：**https://hrl.boyuai.com/**

Thanks

分享人：卢艳峰