# Proximal Policy Optimization Algorithms

发表于 2018-07-03 | 更新于 2018-12-30 | 分类于 Reinforcement Learning | 阅读次数： 19 | Valine： 0



OpenAI 在 *Trust Region Policy Optimization* 的基础上发表了 *Proximal Policy Optimization Algorithms* 论文，提出了一种新颖的目标函数 (surrogate objective function)，通过用随机梯度下降的方法来优化这个函数达到优化策略的目的，称之为 Proximal Policy Optimization (PPO)。它有着一些 TRPO 的优点但是比 TRPO 实现起来更加简单。

OpenAI 在 2018 年 6 月 25 日发文称，OpenAI Five 击败 Dota 2 业余团队，其中主要的算法便是用了大规模 PPO 算法，使用了 256 个 GPU 和 128,000 个 CPU

## 策略优化 Policy Optimization

先简单回顾一下两种策略优化的方法

### 策略梯度下降法 Policy Gradient (PG) Methods

策略梯度下降方法就是用随机梯度下降的优化算法去计算策略梯度的估计，最常见的一种梯度估计形式是

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

其中 $\pi_\theta$ 是随机策略，$\hat{A}_t$ 是 $t$ 时刻的优势函数，$\hat{\mathbb{E}}_t$ 表明需要在一系列采样中进行经验平均，交替进行采样与优化。$\hat{g}$ 是以下目标损失函数的求导形式：

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \Big[ \log \pi_\theta(a_t|s_t) \hat{A}_t \Big]$$

我们的目标就是优化这个目标函数，使其最大化。尽管可以用同一个轨迹 (trajectory) 来应用多次优化损失函数，但这么做的话会让策略更新幅度过于巨大，这对整个算法是毁灭性的。

### 置信域方法 Trust Region Methods

在 TRPO 中，替代的目标函数最大化时需要受到一个约束来限制策略的更新幅度，让新旧策略的 KL 距离不超过一定阈值：

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \Big[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \Big] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)]] \le \delta \end{aligned}$$

这个带约束的优化问题可以使用共轭梯度算法近似地解决。TRPO 中还可以使用惩罚项来代替约束，即以下无约束的优化问题：

$$\underset{\theta}{\text{maximize}} \, \hat{\mathbb{E}}_t \Big[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t - \beta \, \mathrm{KL}[\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)] \Big]$$

TRPO 更常用的是第一种硬约束 (hard constraint) 形式因为第二种中的系数 $\beta$ 并不是非常好选取，如果是对于不同的问题也没有统一的方法来选取固定的值。所以我们不能将这个系数看作是固定的系数。

## 裁剪的替代目标函数 Clipped Surrogate Objective

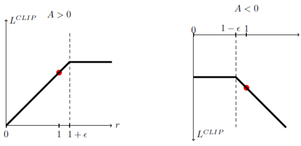
我们用 $r_t(\theta)$ 表示概率分布的比率 $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$，可以看出 $r_t(\theta_{old}) = 1$，TRPO 想要最大化的就是这个替代目标函数：

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \Big[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \Big] = \hat{\mathbb{E}}_t \Big[ r_t(\theta) \hat{A}_t \Big]$$

上标 $CPI$ 代表 conservative policy iteration。如果没有 TRPO 里的约束，这个目标函数会非常快地增长。因此我们考虑对这个目标函数做一下修改，使得当 $r_t(\theta)$ 偏离 1 时增加一些惩罚：

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

其中 $\epsilon$ 是一个超参数，假设 $\epsilon = 0.2$。这个目标函数的外层是一个 min 函数，第一项为 $L^{CPI}$，第二项 $\mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$ 将 $L^{CPI}$ 目标函数修改了一下，它把概率分布比率进行裁剪，使得这个比率的变动限制在 $[1-\epsilon, 1+\epsilon]$ 的范围之内，最后通过外层的 min 函数取裁剪过与未裁剪过的目标函数的最小值，所以最终结果 $L^{CLIP}(\theta)$ 便是 $L^{CPI}$ 的下界。我们可以发现在 $\theta_{old}$ 的附近（也就是 $r = 1$ 的附近），$L^{CLIP}(\theta)$ 与 $L^{CPI}(\theta)$ 一阶相等。$L^{CLIP}(\theta)$ 与概率比率的关系可以用下图表示：




当 $A > 0$ 时，表示当前行为的选取比较好，但策略的更新幅度不能太大，当比率超过 $1+\epsilon$ 时，目标函数不在增长达到最大，而如果比率是减小的，也就是说策略此时变得更坏，那就不用管它。

当 $A < 0$ 时，表示当前行为的选取的不好，同理，选取这个行为的策略需要减少，但减少幅度也不能太大，当比率小于 $1-\epsilon$ 时，则进行裁剪，使目标函数不再减小。

图中红色的点表示优化的起始位置，也就是 $r = 1$ 的地方。

## 适应性 KL 惩罚系数 Adaptive KL Penalty Coefficient

除了上一小节提到的 clipped surrogate objective 之外，我们介绍另一种方法，使用 KL 散度的惩罚项，但与 TRPO 不同的是，我们为 KL 散度的系数增加一些适应性变化，让 KL 散度与我们事先定义的 $d_{targ}$ 在每一次策略更新时相接近。尽管这一算法的效果没有 clipped surrogate objective 好，但我们还是将他写出来，具体算法为：

1. 用 mini-batch 的随机梯度下降方法，优化带 KL 散度惩罚项的目标函数

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \Big[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t - \beta \, \mathrm{KL}[\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)] \Big]$$

2. 计算 $d = \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)]]$

   if $d < d_{targ}/1.5$, $\beta \leftarrow \beta/2$

   if $d > d_{targ} \times 1.5$, $\beta \leftarrow \beta \times 2$

我们有时会发现 KL 散度会严重偏离 $d_{targ}$，但这并不常见，同时 $\beta$ 也会很快就调整过来。参数 1.5 与 2 还有 $\beta$ 的初始值在实践中并不是非常重要，因为算法会很快进行调整适应。

## 算法

经过一些小的改变就能将上面小节中提到的两个替代的目标函数运用到策略梯度算法中。为了减少优势函数的方差，我们经常会使用状态价值函数 $V(s)$。如果让策略与状态价值函数共享同一个神经网络的参数，我们需要将策略与状态价值函数的误差项整合到一个损失函数中，同时也可以增加一个信息熵奖励项来增加探索。将这三项组合一下，可以得到如下的目标函数：

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

其中 $c_1$，$c_2$ 为系数，$S$ 代表信息熵奖励，$L_t^{VF}$ 代表平方误差 $(V_\theta(s_t) - V_t^{targ})^2$

下面算法为使用固定轨迹长度的 PPO 算法

**Algorithm 1** PPO, Actor-Critic Style

```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy π_θold in environment for T timesteps
        Compute advantage estimates Â_1, ..., Â_T
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
    θ_old ← θ
end for
```

每一次迭代，$N$ 个 actor 并行地采样 $T$ 步数据，然后再用这 $NT$ 步数据构建替代的损失函数，并使用 mini-batch 的随机梯度下降算法进行优化 $K$ 次。

## 参考

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

https://www.zhihu.com/question/63067895/answer/211279961



# RL # PG

‹ Trust Region Policy Optimization

Proximal Policy Optimization 代码实现 ›