Are Deep Policy Gradient Algorithms Truly Policy Gradient Algorithms?

Andrew Ilyas*[1], Logan Engstrom*[1], Shibani Santurkar[1], Dimitris Tsipras[1], Firdaus Janoos[2], Larry Rudolph[1,2], and Aleksander Mądry[1]

[1]MIT [2]Two Sigma
{ailyas,engstrom,shibani,tsipras,madry}@mit.edu
rudolph@csail.mit.edu, firdaus.janoos@twosigma.com

【强化学习 27】RethinkPolicyGradient

张楚珩
清华大学 交叉信息院博士在读

RethinkPolicyGradient是我瞎编的，目的就是给个代号便于区分，这篇文章不是讲的一个算法，而是对于现在最先进的Policy Gradient方法提出了一些质疑。

原文传送门

Ilyas, Andrew, et al. "Are Deep Policy Gradient Algorithms Truly Policy Gradient Algorithms?." arXiv preprint arXiv:1811.02553 (2018).

特色

这篇文章读完个人主要有两点比较大的收获。第一点是解答了个人在实现一些state-of-the-art算法时的疑惑，算法简单地实现并不能有很好的效果，需要对照着baseline目录去采用各种小技巧才能达到报告的效果。第二点是更深入了理解算法实际应用时的运行状态和理论动机之间的差异。

背景

策略梯度类的方法最关键的就是有如下的策略梯度估计

$$\widehat{g}_\theta = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{(s_t,a_t)\in\tau} \nabla_\theta \log \pi_\theta(a_t|s_t)\cdot A_{\pi_\theta}(s_t,a_t)\right],$$

注意到上式中待优化的参数出现在了Expectation的下面，这是不好优化的，因此我们找一个替代的优化目标

$$\max_\theta \ \mathbb{E}_{(s_t,a_t)\sim\pi}\left[\frac{\pi_\theta(a_t|s_t)}{\pi(a_t|s_t)}A_\pi(s_t,a_t)\right] \qquad \left(=\mathbb{E}_{\pi_\theta}\left[A_\pi\right]\right),$$

而该目标只有在更新的策略 π_θ 距离原来的策略 π 不太远的时候成立，因此TRPO和PPO分别给出

了一些限制每一步更新产生的策略变化不要太大的方法。

比如TRPO给出了这样的优化方案，直接限制新旧策略平均的KL散度（如果不熟悉，可以参看本专栏前面的文章）

$$\begin{aligned}\max_{\theta} \quad & \mathbb{E}_{(s_t,a_t)\sim\pi}\left[\frac{\pi_\theta(a_t|s_t)}{\pi(a_t|s_t)}\widehat{A}_\pi(s_t,a_t)\right] \\ \text{s.t.} \quad & D_{KL}(\pi_\theta(\cdot\mid s)||\pi(\cdot\mid s))\le\delta, \quad \forall s\ .\end{aligned}$$

PPO给出这样的优化方案，当策略变化太大的之后，就不再给相应的梯度了

$$\max_{\theta}\ \mathbb{E}_{(s_t,a_t)\sim\pi}\left[\min\left(\text{clip}\left(\rho_t,1-\varepsilon,1+\varepsilon\right)\widehat{A}_\pi(s_t,a_t),\ \rho_t\widehat{A}_\pi(s_t,a_t)\right)\right]$$

过程

1. **代码实现十分关键**

这一点对于自己去实现过强化学习算法的同学应该深有体会，我在RND（见本专栏上一篇）文章作者的博客里面也看到他们也十分强调算法的具体实现。

Implementation Matters

Big-picture considerations like susceptibility to the noisy-TV problem are important for the choice of a good exploration algorithm. However, we found that getting seemingly-small details right in our simple algorithm made the difference between an agent that never leaves the first room and an agent that can pass the first level. To add stability to the training, we avoided saturation of the features and brought the intrinsic rewards to a predictable range. We also noticed **significant improvements in performance of RND every time we discovered and fixed a bug** (our favorite one involved accidentally zeroing an array which resulted in extrinsic returns being treated as non-episodic; we realized this was the case only after being puzzled by the extrinsic value function looking suspiciously periodic). Getting such details right was a significant part of achieving high performance even with algorithms conceptually similar to prior work. This is one reason to prefer simpler algorithms where possible.

RND作者的博客里面也提到了Implementation matters（不过说的不完全是同一回事）

对于PPO的实现来说文中作者列举出了很多与算法本身无关的实现小技巧，但实际上这些小技巧大大提升了PPO算法的性能，这些技巧包括

- Value function clipping：PPO提议的是在actor更新的时候使用clipped目标，但是baseline的实现里面对于critic的目标也做了clip

$$L^V = \min\left[\left(V_{\theta_t} - V_{targ}\right)^2, \left(\text{clip}\left(V_{\theta_t}, V_{\theta_{t-1}} - \varepsilon, V_{\theta_{t-1}} + \varepsilon\right) - V_{targ}\right)^2\right],$$

- Reward Scaling：得到的奖励还要去除以统计的方差
- Orthogonal initialization and layer scaling：不仅没有使用一般神经网络默认的Xavier初始化，而且使用了每层都不一样的正交初始化方案
- Adam learning rate annealing：学习率会逐渐减小（感觉和CV领域的标准做法差不多）

文章中的实验表明，后三个都大大提升了PPO的性能。由于这些小技巧并没有太大可解释性，文章后面都测试不使用这些小技巧的原始PPO算法（PPO-M）。

2. 梯度估计地究竟好不好？

- 在实际的应用中，每次参数更新方向（梯度的估计方向）相互之间的平均cosine similarity几乎为零（完全等于零代表各个方向完全随机，没有联系）
- 在实际的应用中，每次参数更新方向与真实的梯度方向间平均cosine similarity几乎为零

由此文章给出结论说现在流行的梯度估计非常差。不过个人对此结论持保留意见，原因如下。

文章没有给出cosine similarity的定义，我查了一下维基百科，定义如下

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

文章看样子是在参数空间上做的similarity，众所周知，神经网络参数基本上是million的级别，在这么高维度的参数上面做cosine similarity个人感觉是比较有问题的。神经网络的参数本来就是高度兼

并的，有大量维度上的数值没啥作用，可能真正有关键作用的维度只有10%（我随口一说），如果这些维度估计的策略梯度都把方向弄对了，很简单算一下，cosine similarity也不会超过0.1。

3. 用估计的状态价值函数做baseline究竟有没有达到预期效果？

- 估计的状态价值函数能有效拟合给定的目标，并且具有泛化能力（因为这其实就是一个监督学习设定）
- 估计的状态价值函数做baseline能减小一点点方差，但是距离理想效果还差的比较远（比较的”理想效果“是用实际的状态价值函数来做baseline，这是用蒙特卡洛的return跑了好多次来估计的，当然效果拔群，但是成本太高）

个人读完这个部分反而使得自己更加坚信了baseline的效果，也对于其效果有了一个更加直观的认识。

4. 替代目标是不是一个好的目标？

- 在训练的初期，替代目标函数的形态和真实目标比较一致；在训练的后期开始不太一致，甚至经常梯度的方向就是反着的（所以能不能在训练得差不多的模型上再用真实目标函数细调？）
- 即使样本量很大的时候，TRPO的目标函数比较好地逼近真实目标，PPO的则没有（有时候甚至是反着的）

5. Trust region方法实际中有没有成功地把参数限制在trust region里？

- TRPO算法由于主要限制的是（相对于不同状态）平均KL散度，因此实际运行时 $\max(\frac{\pi_\theta}{\pi_0})$ 并没有被限制住，但是 $\overline{KL}$ 和 $KL_{\max}$ 是被限制住了的。
- PPO-M（没加小技巧的PPO）算法 $\max(\frac{\pi_\theta}{\pi_0})$ 限制的也不是很好，$\overline{KL}$ 甚至有一种越训越大的趋势，但是加了小技巧的PPO却能够较好地把更新限制在trust region内。

感谢 @裴郢珺 的文章推荐。

彩蛋：现在使用相同的句式也需要引用的么？

Are Deep Policy Gradient Algorithms Truly Policy Gradient Algorithms?

Andrew Ilyas*[1], Logan Engstrom*[1], Shibani Santurkar[1], Dimitris Tsipras[1], Firdaus Janoos[2], Larry Rudolph[1,2], and Aleksander Mądry[1]

[1]MIT [2]Two Sigma

{ailyas,engstrom,shibani,tsipras,madry}@mit.edu
rudolph@csail.mit.edu, firdaus.janoos@twosigma.com

4 Implementation Matters in Policy Gradient Methods

8 Acknowledgments

SS was supported by the National Science Foundation (NSF) under grants IIS-1447786, IIS-1607189, and CCF-1563880, and the Intel Corporation. DT was supported in part by the NSF grant CCF-1553428 and the NSF Frontier grant CNS-1413920. AI was supported in part by NSF awards CCF-1617730 and IIS-1741137. LE was supported in part by an MIT-IBM Watson AI Lab research grant. AM was supported in part by an Alfred P. Sloan Research Fellowship, a Google Research Award, and the NSF grants CCF-1553428 and CNS-1815221.

We thank the authors of [13] and [2] for inspiring the title of the paper and Section 4, respectively.

? ? ? ! !

[2] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *CoRR*, abs/1611.04231, 2016.

[13] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? *CoRR*, abs/1806.00451, 2018.

编辑于 2018-11-15

强化学习 (Reinforcement Learning)