Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

【强化学习算法 3】TRPO

张楚珩
清华大学 交叉信息院博士在读

原文传送门：

Kakade, Sham, and John Langford. "Approximately optimal approximate reinforcement learning." ICML. Vol. 2. 2002.（前序工作）

Schulman, John, et al. "Trust region policy optimization." International Conference on Machine Learning. 2015.

特色：policy gradient类的方法如果顺着梯度方向走的步长太大的话，不能保证每步更新产生的新策略都更好，算法很容易不稳定。这里找到了一种方法能够稳定地一步步改进策略，使得policy gradient类算法更稳定。

分类：Model-free、Policy-based、On-policy、Continuous State Space、Continuous Action Space、Support High-dim Input

理论依据：

如果只是控制***参数空间***的步长的话，步长过小更新地慢，效率低下；步长大的话，算法就会不稳定；甚至不能找到一个好的固定步长。根据前序工作中的理论，如果能够保证***策略空间***上变化不太大的话（保证的方法就是下面的 α 不要太大），就能保证每一步更新得到的新策略都比旧策略更好。

考虑最小化cost，目标是 $\eta(\pi)$ ，有

$$\eta(\pi_{new}) - \eta(\pi_{old}) \leq L_{\pi_{old}}(\pi_{new}) + \frac{2\epsilon\gamma}{(1-\gamma(1-\alpha))(1-\gamma)}\alpha^2 \quad (1)$$

其中 $L_{\pi_{old}}(\pi) = \sum_s \rho_{\pi_{old}}(s) \sum_a \pi(a|s) A_{\pi_{old}}(s,a)$ ， $\pi_{new}(a|s) = (1-\alpha)\pi_{old}(a|s) + \alpha\pi'(a|s)$ ， $\pi' = arg\min_{\pi} L_{\pi_{old}}(\pi)$ ， $\epsilon = \max_s |\mathbb{E}_{a\sim\pi'}[A_{\pi_{old}}(s,a)]|$

过程：

有了前面的理论，那么每次优化(1)式右边的量就好了。但是仍然需要做一些变形。

1. 由于这里使用神经网络来近似的策略，从 π_{old} 到 π_{new} 那样的更新没法做得到，因此，对于参数 α 做一些近似。 $\alpha^2 \to D_{TV}^{max}(\pi_{old}, \pi_{new}) \to D_{KL}^{max}(\pi_{old}, \pi_{new}) \to \overline{D_{KL}^{\rho_{old}}}(\pi_{old}, \pi_{new})$
2. 由于 ϵ 也不好估计，因此 $\epsilon \to C = \frac{2\epsilon\gamma}{(1-\gamma)^2}(\text{regularisation factor}) \to \delta(\text{hard constraint})$
3. 对于 $L_{\pi_{old}}(\pi) \to L_{\theta_{old}}(\theta) \to \nabla_\theta L_{\theta_{old}}(\theta)|_{\theta=\theta_{old}}(\theta-\theta_{old})$ ，对于 $\overline{D_{KL}^{\rho_{old}}}(\pi_{old}, \pi_{new}) \to \frac{1}{2}(\theta-\theta_{old})^T A(\theta_{old})(\theta-\theta_{old})$ ，其中 $A(\theta_{old}) = \nabla_\theta^2 \mathbb{E}_{s\sim\rho_{old}} D_{KL}[\pi_{\theta_{old}}(s)||\pi_\theta(s)]|_{\theta=\theta_{old}}$

每步优化：

$$\underset{\theta}{\text{minimize}}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right] \qquad (15)$$
$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)\,\|\,\pi_\theta(\cdot|s))\right]\le\delta.$$

$$\underset{\theta}{\text{minimize}}\left[\nabla_\theta L_{\theta_{\text{old}}}(\theta)\big|_{\theta=\theta_{\text{old}}}\cdot(\theta-\theta_{\text{old}})\right] \qquad (18)$$
$$\text{subject to } \frac{1}{2}(\theta_{\text{old}}-\theta)^T A(\theta_{\text{old}})(\theta_{\text{old}}-\theta)\le\delta,$$
$$\text{where } A(\theta_{\text{old}})_{ij}=$$
$$\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\mathbb{E}_{s\sim\rho_\pi}\left[D_{\text{KL}}(\pi(\cdot|s,\theta_{\text{old}})\,\|\,\pi(\cdot|s,\theta))\right]\Big|_{\theta=\theta_{\text{old}}}$$

算法：

每步on-policy采样，计算advantages，然后解上述优化问题，更新policy的权值。

用到的其他技术：

1. 不去把Hessian矩阵 A 计算处理储存，而是构造一个黑盒子直接以 $O(n)$ （ n 是神经网络参数数目）来计算 $A\cdot v$ 该矩阵和任意向量的乘积；
2. 优化问题是linear program with quadratic constraint，能够使用conjugate gradient先得到最优解的方向 $\sqrt{\frac{2\delta}{v^TAv}}$，并且得到满足约束的最大步长，然后用line search求得最优解。

另外：

优化问题需要 $Q_{\theta_{\text{old}}}(s,a)$ 项，文中提到了两种方式得到。一种方式（single path）就是每次采样整条轨迹，然后把未来的return作为Q值。另一种方式（vine）在每个状态上还会多做一些rollout，得到variance更小的Q值估计，但是需要环境支持能够回退到任意一个过去的状态。

编辑于 2018-09-19

强化学习 (Reinforcement Learning)　算法（书籍）　算法