Trust Region Policy Optimization

Introduction：

信赖域策略优化，该算法和自然的 PG 方法很接近，但同时对于大型非线性策略比如神经网络很有效。PG 可以被分为 3 种方法：(1)策略迭代 policy iteration 根据价值函数进行估计。(2)策略梯度下降，从样本 trajectories 中获得期望累计回报值的梯度用作策略的估计。(3)自由多目标优化方法(广撒点)像交叉熵方法 the cross-entropy method (CEM) 和协方差矩阵自适应法 covariance matrix adaptation (CMA)。

https://blog.csdn.net/mmc2015/article/details/81783448

交叉熵方法是一种蒙特卡洛方法，主要用来优化和重要性采样。和进化算法类似，在空间中按照某种规则撒点，获得每个点的误差，再根据这些误差信息决定下一轮撒点的规则。交叉熵方法之所以叫这个名字，是因为该方法（从理论上来说）目标是最小化随机撒点得到的数据分布与数据实际分布的交叉熵（等价于最小化 KL 距离），尽量使采样分布（撒的点）与实际情况同分布。

该方法适当选取撒点规则就可以适应多目标优化等情况，在组合优化中也有许多应用。本文主要讨论 CEM 在强化学习的策略优化中的应用。

https://blog.csdn.net/rui307/article/details/78744807

CMA 是一种随机的，不需要计算梯度的数值优化算法。主要用来解决非线性、非凸的优化问题，属于进化算法的一类，具有随机性。

对于连续控制问题，像 CMA 等算法可以成功学习控制策略。连续的 gradient-based 优化方法在具有大量参数的监督学习任务下可以成功学习函数估计器。

TRPO 解决了带大量参数的无模型策略搜索带来的重大挑战。

由于重要性采样(importance sampling)的关系我们希望每次更新的时候策略分布之间差距并不是很大，这实际上是一种约束，即我们希望能每次更新的时候不大幅度地改变分布的形态，基于这种考虑 openai 的前辈们提出了 TRPO 算法，但是 TRPO 算法会有一些缺陷，他拿二次函数去近似约束条件，拿一次函数近似待优化的损失函数，这种近似会造成收敛上的困难。

我们知道，根据策略梯度方法，参数更新方程式为：

$$\theta_{new} = \theta_{old} + \alpha \nabla_{\theta} J$$

由方程式得到更新方法对于步长十分敏感。那什么样的步长叫做合适的步长呢，试想我们如果能找到一种步长，使他每次更新时都能保证回报函数单调递增，这样的步长就是好步长。TRPO 的核心就是解决学习率使得单调的问题。

TRPO 的做法是将新的策略所对应的回报函数分解成旧的策略所对应的回报函数+其他项。只要新的策略所对应的其他项大于等于零，那么新的策略就能保证回报函数单调不减。

我们用 τ 来表示一个状态行为序列，或者说一条轨迹 s0 r0 s1 r1 s2 r2...，那么在策略Π′下的期望即时回报可以看做是如下式子：

$$\eta(\tilde{\pi}) = E_{\tau|\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t (r(s_t))]$$

Preliminaries：

η(π) 表示在策略Π下的期望即时回报值. η(π) = E[V(s0)]在策略Π下，

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where}$$
$$s_0 \sim \rho_0(s_0),\ a_t \sim \pi(a_t|s_t),\ s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] \quad (1)$$

证明如下：

Proof. First note that $A_\pi(s,a) = \mathbb{E}_{s' \sim P(s'|s,a)}\left[r(s) + \gamma V_\pi(s') - V_\pi(s)\right]$. Therefore,

$$\begin{aligned}
&\mathbb{E}_{\tau|\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] \\
&= \mathbb{E}_{\tau|\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t (r(s_t) + \gamma V_\pi(s_{t+1}) - V_\pi(s_t))\right] \\
&= \mathbb{E}_{\tau|\tilde{\pi}}\left[-V_\pi(s_0) + \sum_{t=0}^{\infty} \gamma^t r(s_t)\right] \\
&= -\mathbb{E}_{s_0}\left[V_\pi(s_0)\right] + \mathbb{E}_{\tau|\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right] \\
&= -\eta(\pi) + \eta(\tilde{\pi})
\end{aligned}$$

我们把(1)展开，得:

$$\begin{aligned}
\eta(\tilde{\pi}) &= \eta(\pi) + \sum_{t=0}^{\infty} \sum_{s} P(s_t = s|\tilde{\pi}) \sum_{a} \tilde{\pi}(a|s) \gamma^t A_\pi(s,a) \\
&= \eta(\pi) + \sum_{s} \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\tilde{\pi}) \sum_{a} \tilde{\pi}(a|s) A_\pi(s,a) \\
&= \eta(\pi) + \sum_{s} \rho_{\tilde{\pi}}(s) \sum_{a} \tilde{\pi}(a|s) A_\pi(s,a). \quad (2)
\end{aligned}$$

其中，$\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \ldots,$

注意这里 s 是由新分布策略Π′产生的，对新分布有很强的依赖性。但新的策略无从所知。为此，TRPO 采取了一些技巧来解决这个问题。

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_{s} \rho_\pi(s) \sum_{a} \tilde{\pi}(a|s) A_\pi(s,a). \quad (3)$$

保守策略迭代更新：

$$\pi_{\text{new}}(a|s) = (1-\alpha)\pi_{\text{old}}(a|s) + \alpha\pi'(a|s). \quad (5)$$

Kakade and Langford derived the following lower bound:

$$\begin{aligned}
&\eta(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2 \\
&\text{where } \epsilon = \max_s \left|\mathbb{E}_{a \sim \pi'(a|s)}\left[A_\pi(s,a)\right]\right|. \quad (6)
\end{aligned}$$

一般随机政策的单调改进保证：

我们使用计算策略π和π~的距离代替 α ^2，同时适当地调整常数。该结果对于实用性具有很好的保证。

$$\eta(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - CD_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}),$$
$$\text{where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}. \tag{9}$$

与自身分布的 KL 散度为 0， $\eta(\pi_i) = M_i(\pi_i),$

所以，

$\eta(\pi_{i+1}) \geq M_i(\pi_{i+1})$ by Equation (9)
$\eta(\pi_i) = M_i(\pi_i)$, therefore,
$$\eta(\pi_{i+1}) - \eta(\pi_i) \geq M_i(\pi_{i+1}) - M(\pi_i). \tag{10}$$

其中，

$\eta(\pi_0) \leq \eta(\pi_1) \leq \eta(\pi_2) \leq \ldots$. To see this, let $M_i(\pi) = L_{\pi_i}(\pi) - CD_{\mathrm{KL}}^{\max}(\pi_i, \pi)$. Then

因此，在迭代中最大化 Mi，可以保证真是的目标函数η单调不减。
算法是一种最小化最大化算法 minorization-maximization (MM) algorithm。

Algorithm 1 Policy iteration algorithm guaranteeing non-decreasing expected return η

Initialize π_0.
for $i = 0, 1, 2, \ldots$ until convergence **do**
Compute all advantage values $A_{\pi_i}(s, a)$.
Solve the constrained optimization problem

$$\pi_{i+1} = \arg\max_{\pi} \left[L_{\pi_i}(\pi) - CD_{\mathrm{KL}}^{\max}(\pi_i, \pi) \right]$$
$$\text{where } C = 4\epsilon\gamma/(1-\gamma)^2$$
$$\text{and } L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$$

end for

参数化的策略优化方法：

因此，我们使用以下优化目标以保证真实目标函数 η 的优化：

$$\underset{\theta}{\text{maximize}}\left[L_{\theta_{\text{old}}}(\theta)-CD_{\text{KL}}^{\max}(\theta_{\text{old}},\theta)\right].$$

由于惩罚因子 C 导致步长过小，一种稳健的解决办法是对新策略、旧策略的 KL 散度进行约束，被称为信赖域约束：

$$\begin{aligned}&\underset{\theta}{\text{maximize}}\,L_{\theta_{\text{old}}}(\theta) &(11)\\&\text{subject to } D_{\text{KL}}^{\max}(\theta_{\text{old}},\theta)\le\delta.\end{aligned}$$

我们考虑一种平均 KL 散度作为近似估计：

$$\overline{D}_{\text{KL}}^{\rho}(\theta_1,\theta_2):=\mathbb{E}_{s\sim\rho}\left[D_{\text{KL}}(\pi_{\theta_1}(\cdot|s)\parallel\pi_{\theta_2}(\cdot|s))\right].$$

We therefore propose solving the following optimization problem to generate a policy update:

$$\begin{aligned}&\underset{\theta}{\text{maximize}}\,L_{\theta_{\text{old}}}(\theta) &(12)\\&\text{subject to } \overline{D}_{\text{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}},\theta)\le\delta.\end{aligned}$$

对于目标和约束的 Sample-Based 估计：

该部分介绍目标和约束函数在使用蒙特卡洛方法进行近似。

We first replace $\sum_s \rho_{\theta_{\text{old}}}(s)\left[\ldots\right]$ in the objective by the expectation $\frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[\ldots\right]$. Next, we replace the advantage values $A_{\theta_{\text{old}}}$ by the Q-values $Q_{\theta_{\text{old}}}$ in Equation (13),

利用重要性采样对动作分布进行的处理：

https://zhuanlan.zhihu.com/p/41217212

https://www.jianshu.com/p/5118458fc061

重要性采样是，使用另外一种分布来逼近所求分布一种方法。

重要性采样（IS）就是，如果原先的变量 x 应该服从 $x_i\sim\pi(x)$，但是我们对 $\pi(x)$ **难以采样**，因此我们采用一个与 $\pi(x)$ **相近且易于采样的分布** $p(x)$ 对 x 进行采样。由于是近似，这样的话会出现有的地方 $p(x_i)>\pi(x_i)$，而有的地方 $p(x_i)<\pi(x_i)$。因此我们对采得的每个样本都加上一个权重 $\dfrac{\pi(x_i)}{p(x_i)}$，比原分布概率大的样本就减小它的权重，比原分布概率小的样本我们增加它所占的权重，最后再对权重归一化：

$$S=\frac{\sum\frac{\pi(x_i)}{p(x_i)}f(x_i)}{\sum\frac{\pi(x_i)}{p(x_i)}}$$

> 重要性采样需要解决的问题是：如何对一个已知的概率分布得到样本，即抽样（sampling）

$$\sum_a \tilde{\pi}_\theta(a|s_n) A_{\theta_{old}}(s_n, a) = E_{a\ q}\left[\frac{\tilde{\pi}_\theta(a|s_n)}{q(a|s_n)} A_{\theta_{old}}(s_n, a)\right]$$

通过利用两个技巧，我们再利用 $\frac{1}{1-\gamma} E_{s\ \rho_{\theta_{old}}}[\cdots]$ 代替 $\sum_s \rho_{\theta_{old}}(s)[\cdots]$；取 $q(a|s_n) = \pi_{\theta_{old}}(a|s_n)$ ；

因此优化问题可以写成以下形式：

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{\text{old}}}(s,a)\right] \qquad (14)\\ &\text{subject to}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s) \parallel \pi_\theta(\cdot|s))\right] \le \delta. \end{aligned}$$

此外，就是实践中用样本平均代替期望，用经验估计代替 Q 值。

以下介绍两种不同的方法进行估计。

1. Single path

该方法是基于对个体轨迹的采样。

在该方法的估计下，我们采样一个状态序列 s0 ～ ρ 0，其中有一部分是和策略 π θ old 一致的，比如 s0, a0, s1, a1, . . . , sT 1, aT 1, sT。因此，q(a|s) = π θ old (a|s)，Q θ old (s, a)是根据状态动作对(st, at)和折扣累积的估计值。

2. Vine 藤曼

该方法涉及到构建一个 roll-out 集合，在 roll-out 集合中对于每一个状态执行多个动作。该方法在策略迭代的背景下有更多的扩展。

我们之后在其中选择 N 个状态作为子集 s1, s2, . . . , sN，称为 roll-out 集合，对于集合中的每一个状态 sn 我们采样 K 个动作，对于动作 k，k ～ q(·|sn)。实践中，我们发现 q(·|sn) = πθi (·|sn)可以在连续问题上运作的很好。

对于每个动作 a(n, k)，我们从 roll-out 集合中估计 Q θ i(sn, a(n,k))。

而对于单个状态 sn，Lθold 表示为：

$$L_n(\theta) = \sum_{k=1}^{K} \pi_\theta(a_k|s_n)\hat{Q}(s_n, a_k), \qquad (15)$$

在大型连续状态空间情况下，可以使用带权重的 Importance sampling(重要采样)：

$$L_n(\theta) = \frac{\sum_{k=1}^{K} \frac{\pi_\theta(a_{n,k}|s_n)}{\pi_{\theta_{\text{old}}}(a_{n,k}|s_n)} \hat{Q}(s_n, a_{n,k})}{\sum_{k=1}^{K} \frac{\pi_\theta(a_{n,k}|s_n)}{\pi_{\theta_{\text{old}}}(a_{n,k}|s_n)}}, \qquad (16)$$

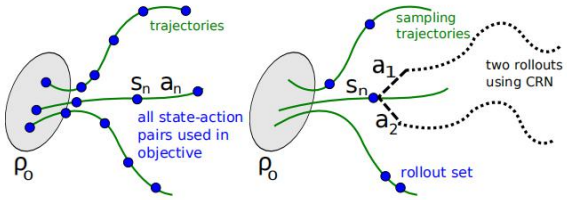

Figure 1. Left: illustration of single path procedure. Here, we generate a set of trajectories via simulation of the policy and incorporate all state-action pairs (s_n, a_n) into the objective. Right: illustration of vine procedure. We generate a set of "trunk" trajectories, and then generate "branch" rollouts from a subset of the reached states. For each of these states s_n, we perform multiple actions (a_1 and a_2 here) and perform a rollout after each action, using common random numbers (CRN) to reduce the variance.

藤曼方法相较于单路径方法的优点是是在给定替代目标的 q 值样本数量相同的情况下，我们对目标的局部估计的方差要小得多。

vine 方法的缺点是，对于每种优势估计，我们必须执行更多的调用。

Practical Algorithm：

算法重复执行以下步骤：

1. 使用单路径方法或者藤曼方法采集一群状态动作对对其 Q 值进行蒙特卡洛估计。
2. 取平均值。
3. 我们使用共轭梯度算法，然后进行直线搜索。

这种分析估计器在大规模设置中具有计算优势，因为它消除了从一批 trajectories 中存储稠密的 Hessian 或所有策略梯度的需要。

- The constraint on $D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta)$ is hard for numerical optimization and estimation, so instead we constrain $\overline{D}_{\mathrm{KL}}(\theta_{\mathrm{old}}, \theta)$.

Connections with Prior Work：

Experiments：

We designed our experiments to investigate the following questions:

1. What are the performance characteristics of the *single path* and *vine* sampling procedures?
2. TRPO is related to prior methods (e.g. natural policy gradient) but makes several changes, most notably by using a fixed KL divergence rather than a fixed penalty coefficient. How does this affect the performance of the algorithm?
3. Can TRPO be used to solve challenging large-scale problems? How does TRPO compare with other methods when applied to large-scale problems, with regard to final performance, computation time, and sample complexity?

1. Simulated Robotic Locomotion

The states of the robots are their generalized positions and velocities, and the controls are joint torques.

1. *Swimmer*. 10-dimensional state space, linear reward for forward progress and a quadratic penalty on joint effort to produce the reward $r(x, u) = v_x - 10^{-5}\|u\|^2$. The swimmer can propel itself forward by making an undulating motion.

2. *Hopper*. 12-dimensional state space, same reward as the swimmer, with a bonus of $+1$ for being in a non-terminal state. We ended the episodes when the hopper fell over, which was defined by thresholds on the torso height and angle.

3. *Walker*. 18-dimensional state space. For the walker, we added a penalty for strong impacts of the feet against the ground to encourage a smooth walk rather than a hopping gait.

我们使用 Figure 3 所示的神经网络结构图表达策略：

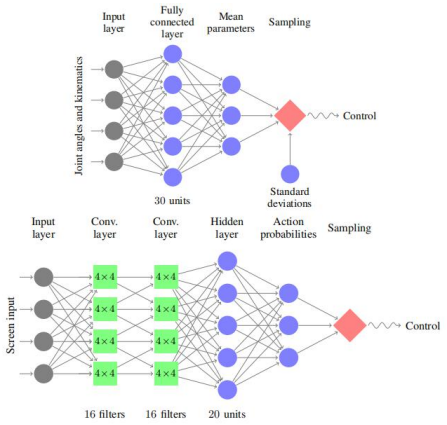

Figure 3. Neural networks used for the locomotion task (top) and for playing Atari games (bottom).

Results:

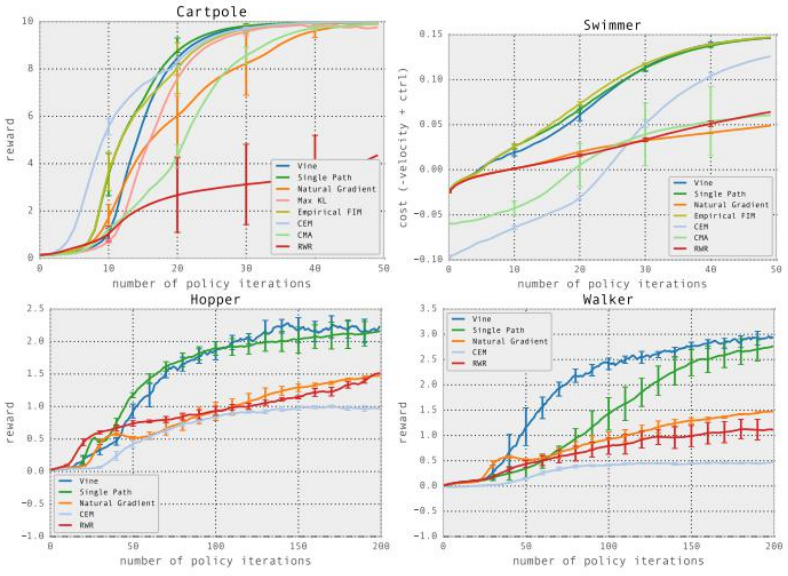

Figure 4. Learning curves for locomotion tasks, averaged across five runs of each algorithm with random initializations. Note that for the hopper and walker, a score of -1 is achievable without any forward velocity, indicating a policy that simply learned balanced standing, but not walking.

2. Playing Games from Images

	B. Rider	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S. Invaders*
Random	354	1.2	0	−20.4	157	110	179
Human (Mnih et al., 2013)	7456	31.0	368	−3.0	18900	28010	3690
Deep Q Learning (Mnih et al., 2013)	4092	168.0	470	20.0	1952	1705	581
UCC-I (Guo et al., 2014)	5702	380	741	21	20025	2995	692
TRPO - single path	1425.2	10.8	534.6	20.9	1973.5	1908.6	568.4
TRPO - vine	859.5	34.2	430.8	20.9	7732.5	788.4	450.2

Table 1. Performance comparison for vision-based RL algorithms on the Atari domain. Our algorithms (bottom rows) were run once on each task, with the same architecture and parameters. Performance varies substantially from run to run (with different random initializations of the policy), but we could not obtain error statistics due to time constraints.

卷积神经网络如 Figure 3 所示，16 个信道，窗宽为 2 的 2 个卷积层，连接一个 20 个神经元的全连接层，总共有 33,500 参数个数。

在 16 核电脑上 30 小时跑了 500 回合。

Discussion：

我们证明了一个算法的单调改进，该算法在 KL 散度损失的情况下反复优化策略的期望收益的局部近似。

我们的分析还提供了一个统一策略梯度和策略迭代方法的视角。

使用更复杂的策略，包括具有隐藏状态的递归策略，可以进一步使在部分观察设置中将状态估计和控制滚入相同的策略成为可能。