法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

第9讲　信赖域系方法

强化学习

主讲人：叶梓

上海交通大学博士

主要研究方向：机器学习、深度学习、人工智能

本章内容

- 信赖域系方法背景
- 信赖域系方法发展路线图
- TRPO方法
- PPO方法
- DPPO方法简介
- ACER方法简介
- 案例

Trust Region系方法的背景

- 本讲介绍一个相对独立的方法分支，它主要由OpenAI主导。
- 相关大牛：John Schulman（cs294课程的主讲人）、Abbeel的学生。
- 近年来，学界开始将优化方法中的信赖域（Trust region）方法引入增强学习，并在各种实验场景中取得了良好的效果。
  - 其中典型的有TPRO，和受其影响衍生出的一系列前沿算法，如：PPO，ACER等。
  - 其中，PPO已成为OpenAI的默认强化学习算法。

Trust Region系方法的背景

- 信赖域系强化学习方法，来源于优化论中的信赖域方法和机器学习中强化学习的结合。
- 信赖域方法，在Nocedal和Wright的《Numerical Optimization》一书中介绍了解优化问题的两种策略：line search和trust region。本质上它们的作用都是在优化迭代过程中从当前点找寻下一点。
- 它们的最大区别是：先确定步长还是先确定方向。
  - Line search方法先确定方向d，再确定步长。
  - trust region方法则先把搜索范围缩小到一个小的范围，小到能够用另一个函数（Model function）去近似目标函数（Objective function），然后通过优化这个model function来得到参数更新的方向及步长。

Trust Region系方法的背景

- 回顾一下PG方法，它的基本思想是考虑由参数θ控制的随机策略π(θ)，然后通过优化与策略相关的目标函数J(θ)（比如累积折扣回报和）来更新策略的参数。
- 这种policy-based方法与RL中另一个重要分支value-based方法相比，不仅避免了值函数（Value function）误差导致的策略退化（Policy degradation），而且更加易于用在连续动作空间问题。
- 但是它也有一些缺点：
  - 一是数据效率（Data efficiency）或是说样本利用率低；
  - 二是方差（Variance）大，它会使得训练困难基于无法收敛。

Trust Region系方法的背景

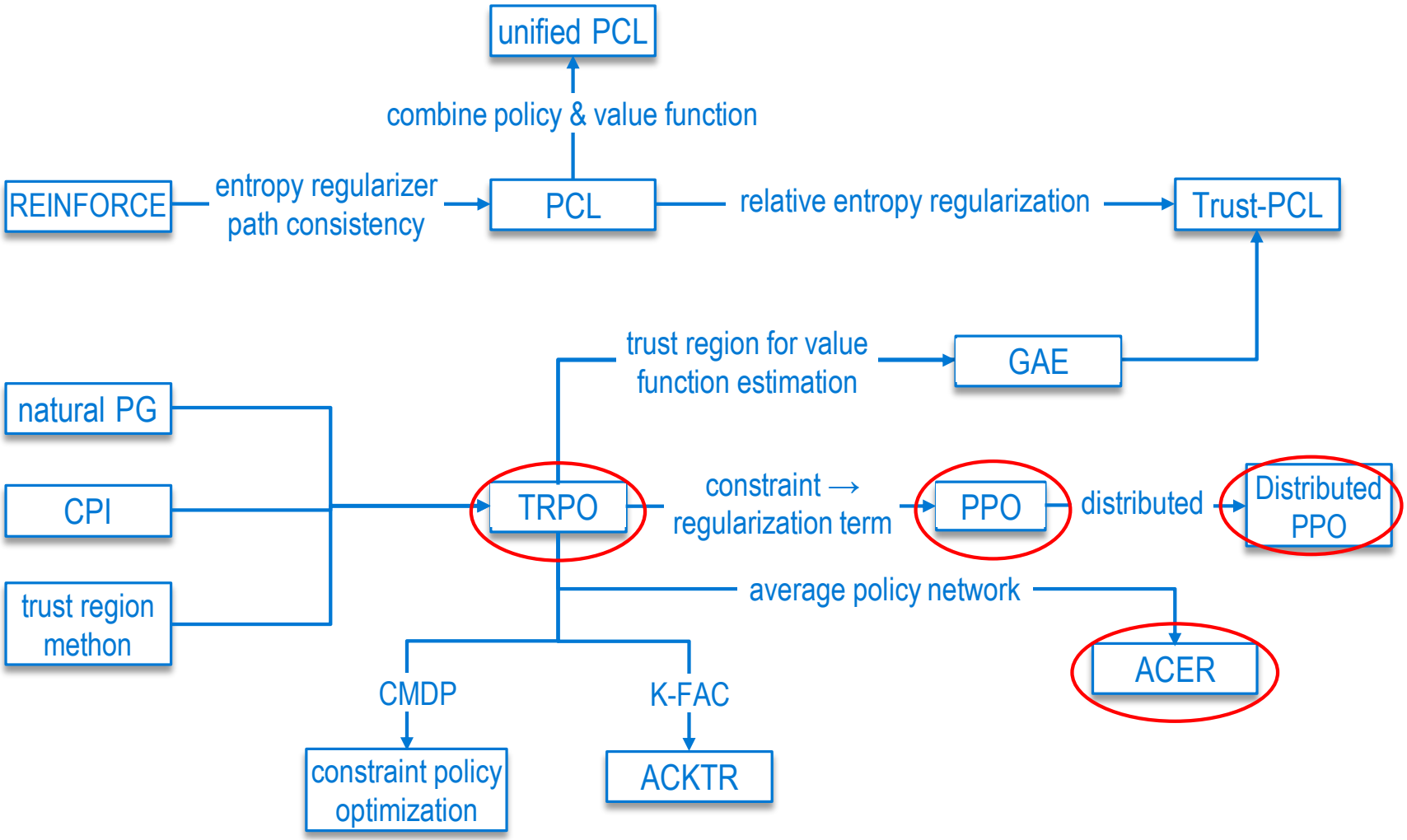
- 针对这些缺点，其中一个比较经典而有效的方法是在原始PG方法中引入value-based方法对值函数进行估计。
- 这就是行动者-评论家（Actor-Critic）方法，同时在估计的值函数减去baseline来减少方差。
- 但是，AC算法在策略搜索过程中如果碰到大的平坦区域（Plateau）就会导致收敛速度过慢，或者过早收敛。
- 原因是这些地方梯度很小，或不直接指向最优解。

Trust Region系方法的背景

- 此后，Kakade在2001年的论文《Natural Policy Gradient》将自然梯度引入强化学习中的PG方法。策略参数的更新方向就变为自然梯度。
- Peters等人在2005年的论文《Natural Actor-Critic》中讨论了它与AC框架的结合。
- 之后在论文《Reinforcement Learning of Motor Skills with Policy Gradients》中对这些工作有些总结。

- Trust Region系方法之间关系大体如下：

Trust Region系方法的发展路线图



TRPO方法

- 根据策略梯度方法，参数更新方程式为：

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

- 如何更新步长α是策略梯度算法的关键问题，当步长不合适时，更新的参数所对应的策略是一个更差的策略。
- 当利用这个更差的策略进行采样学习时，再次更新的参数会更差，因此很容易导致越学越差。
- 直观的理解，所谓合适的步长是指当策略更新后，回报函数的值不能变得更差。

TRPO方法

- 如何找到新的策略使得新的回报函数的值单调增，或单调不减？
- Trust region policy optimization(TRPO)是一种能保证策略单调改进的迭代方法。
- 我们用τ来表示一个状态行为序列（轨迹：$s_0, a_0, s_1, a_1 \ldots$），那么某种策略下的期望回报可以看做是如下式子。

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

τ := (s_0, a_0, s_1, a_0, . . .), and the notation $E_{\tau\sim\tilde{\pi}}$[.] indicates that actions are sampled from $\tilde{\pi}$ to generate τ.

TRPO方法

- 既然希望回报函数单调不减，一个朴素的想法就是：能不能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。

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

- 这里，π表示旧的策略，用$\tilde{\pi}$表示新的策略；A是之前介绍过的“优势函数”：

$$A_{\pi}(s,a) = Q_{\pi}(s,a) - V_{\pi}(s) = E_{s' \sim P(s'|s,a)}[r(s) + \gamma V^{\pi}(s') - V^{\pi}(s)]$$

- 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。

TRPO方法

$$\mathbb{E}_{s_t\sim\tilde{\pi}|n_t=0}\left[\bar{A}(s_t)\right]=\mathbb{E}_{s_t\sim\pi|n_t=0}\left[\bar{A}(s_t)\right]$$

关于上式的证明：

$$E_{\tilde{\pi}}[\sum_{t=0}^{\infty}\gamma^t A_\pi(s_t,a_t)]$$

$$=E_{\tilde{\pi}}[\sum_{t=0}^{\infty}\gamma^t(r(s_t)+\gamma V_\pi(s_{t+1})-V_\pi(s_t))]$$

按A的定义

$$=E_{\tilde{\pi}}[\sum_{t=0}^{\infty}\gamma^t r(s_t)]+E_{\tilde{\pi}}[\sum_{t=0}^{\infty}\gamma^t(\gamma V_\pi(s_{t+1})-V_\pi(s_t))]$$

$$=E_{\tilde{\pi}}[\sum_{t=0}^{\infty}\gamma^t r(s_t)]+E_{\tilde{\pi}}[-V_\pi(s_0)]$$

$$=\eta(\tilde{\pi})-\eta(\pi)$$

?

$$=-\mathbb{E}_{s_0}[V_\pi(s_0)]+\mathbb{E}_{\tau|\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t)\right]$$

$$\gamma V_\pi(s_1)-V_\pi(s_0)+\gamma^2 V_\pi(s_2)-\gamma V_\pi(s_1))+\gamma^3 V_\pi(s_3)-\gamma^2 V_\pi(s_2)\ldots$$

TRPO方法

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

- 为体现出策略项，需要前面说的回报函数分解公式进行进一步的分解。

$$\eta(\tilde{\pi}) = \eta(\pi) + \underbrace{\sum_{t=0}^{\infty}}_{A}\underbrace{\sum_{s}}_{B}P(s_t = s|\tilde{\pi})\underbrace{\sum_{a}}_{C}\tilde{\pi}(a|s)\gamma^t A_\pi(s,a)$$

- A：对所有序列步骤进行累加；
- B：对于所有状态，按其在策略下第t步的出现概率加权；
- C：对于当前状态下所有可能动作，按其概率（即策略）加权。

$\sum(a\sum b)=\sum\sum(ab)$

$\sum[k*f(i)]=k\sum f(i)$;(k为常数,或与i无关的式子)

TRPO的第一个技巧

- 定义：

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

  - 它指的是：基于策略的s状态出现概率的折扣和。

- 则有：

$$\eta(\tilde{\pi})=\eta(\pi)+\sum_{t=0}^{\infty}\sum_{s}P(s_t=s|\tilde{\pi})\sum_{a}\tilde{\pi}(a|s)\gamma^t A_\pi(s,a)$$

$$\eta(\tilde{\pi})=\eta(\pi)+\sum_{s}\rho_{\tilde{\pi}}(s)\sum_{a}\tilde{\pi}(a|s)A^\pi(s,a)$$

- 注意红圈里面的，这时状态s的分布由新的策略产生，即状态依赖于新策略。
- 但它是没法采样的.....

TRPO的第一个技巧

- 第一个技巧：
- 对状态分布进行处理：忽略状态分布的变化，依然采用旧的策略所对应的状态分布。

$$L_{\pi}\left(\tilde{\pi}\right)=\eta\left(\pi\right)+\sum_{s}\rho_{\pi}\left(s\right)\sum_{a}\tilde{\pi}\left(a|s\right)A^{\pi}\left(s,a\right)$$

- 当新旧参数很接近时，用旧的状态分布代替新的状态分布也是合理的。

TRPO的第二个技巧

- 这时动作a是由新的策略产生。可是新的策略$\tilde{\pi}$是带参数θ的，这个参数是未知的，因此无法用来产生动作。
- 引入第二个技巧，是利用重要性采样对动作分布进行的处理：$q(a|s_n) = \pi_{\theta_{old}}(a|s_n)$

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

- 记：$L_\pi(\tilde{\pi}) = \eta(\pi) + E_{s\sim\rho_{\theta_{old}}, a\sim\pi_{\theta_{old}}}[\frac{\tilde{\pi}_\theta(a|s_n)}{\pi_{\theta_{old}}(a|s_n)} A_{\theta_{old}}(s_n, a)]$

- $L_\pi(\tilde{\pi})$与$\eta(\tilde{\pi})$的唯一区别是状态分布的不同，实际上，$L_\pi(\tilde{\pi})$和$\eta(\tilde{\pi})$在策略$\pi_{\theta old}$一阶近似。

有点疑问……

- Schulman的原文中就是这么写的：

We first replace $\sum_s \rho_{\theta_{\text{old}}}(s)[\ldots]$ in the objective by the expectation $\frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}[\ldots]$. Next, we replace the advan-

用 $\frac{1}{1-\gamma}E_{s\ \rho_{\theta_{old}}}[\cdots]$ 代替 $\sum_s \rho_{\theta_{old}}(s)[\cdots]$ ？

- ρ_π(s)是如下定义的：

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

- d^π(s)是如下定义的：

折扣状态分布，定义是$d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s^t = s|\pi)$

TRPO的第二个技巧

- 因此在θ_{old}附近，能改善L的策略也能改善η，

$$L_{\pi_{\theta_{old}}}\left(\pi_{\theta_{old}}\right)=\eta\left(\pi_{\theta_{old}}\right)$$
$$\nabla_{\theta}L_{\pi_{\theta_{old}}}\left(\pi_{\theta}\right)|_{\theta=\theta_{old}}=\nabla_{\theta}\eta\left(\pi_{\theta}\right)|_{\theta=\theta_{old}}$$

- 那么，步长是多少？为此，引入一个非常重要的不等式

$$\eta(\tilde{\pi})\geqslant L_{\pi}(\tilde{\pi})-CD_{KL}^{\max}(\pi,\tilde{\pi})$$
$$where\ C=\frac{2\varepsilon\gamma}{(1-\gamma)^2}$$

$\epsilon = \max_s |E_{a\sim\pi'(a|s)}[A_\pi(s,a)]|$

- 这里的推导简述如下：利用“总变异散度”来描述两个分布的差异。$D_{TV}(p\parallel q)=\frac{1}{2}\sum_i|p_i-q_i|$
- 然后证明了，令：$\alpha=D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}},\pi_{\mathrm{new}})$.
- 则有 $\eta(\pi_{\mathrm{new}})\geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}})-\frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2$

$$where\ \epsilon=\max_{s,a}|A_{\pi}(s,a)|$$

TRPO的第二个技巧

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

$$\pi_{i+1} = \arg\max_{\pi} \left[L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi) \right]$$

where $C = 4\epsilon\gamma/(1-\gamma)^2$

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

- 根据上一页的不等式和本页算法，可以得到一个单调递增的策略序列。

$\eta(\pi_0) \le \eta(\pi_1) \le \eta(\pi_2) \le \ldots.$

$$M_i(\pi) = L_{\pi_i}(\pi) - C D_{KL}^{\max}(\pi_i, \pi)$$

$$\eta(\pi_{i+1}) \ge M_i(\pi_{i+1}), \text{且} \eta(\pi_i) = M_i(\pi_i) \qquad \eta(\pi_{i+1}) - \eta(\pi_i) \ge M_i(\pi_{i+1}) - M(\pi_i)$$

- TRPO算法其实就是本页算法的近似，它对KL散度增加了一个限制。

TRPO的第二个技巧

- 只需要在下一步的策略中找到一个是当前M_i最大的策略，根据上面的不等式，很容易得到策略的期望回报是单调递增的。
- 因此问题就转化为最大化这个下界。这个下界也称为原目标函数的代理函数。

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

C是个很大的惩罚系数。

- 直接优化这个代理函数会导致步长非常小，
- TRPO的做法是用前后策略的KL散度的约束来代替上面的惩罚项。

$$\max_\theta E_{s\sim\rho_{\theta_{old}}, a\sim\pi_{\theta_{old}}}\left[\frac{\tilde{\pi}_\theta(a|s_n)}{\pi_{\theta_{old}}(a|s_n)} A_{\theta_{old}}(s_n, a)\right]$$

$$subject\ to \quad D_{KL}^{\max}(\theta_{old}, \theta) \le \delta$$

比较小的数

TRPO的第三个技巧

- 因为有无穷多的状态，因此约束条件有无穷多个，问题仍然不可解。
- 第三个技巧：这里用平均KL散度代替最大的KL散度。这样优化问题就变为在：

$$subject\ to\ \ \bar{D}_{KL}^{\rho_{\theta_{old}}}(\theta_{old},\theta) \le \delta$$

- 约束下，最大化目标函数$L_{\theta old}(\theta)$。
- 最终TRPO问题化简为：

新策略概率分布与旧策略概率分布不能相差太大

$$\max_{\theta} E_{s\sim\rho_{\theta_{old}},a\sim\pi_{\theta_{old}}}[\frac{\tilde{\pi}_{\theta}(a|s_n)}{\pi_{\theta_{old}}(a|s_n)}A_{\theta_{old}}(s_n,a)] \quad subject\ to\ \ \bar{D}_{KL}^{\rho_{\theta_{old}}}(\theta_{old},\theta) \le \delta$$

TRPO方法

- 进一步简化TRPO问题：

$$\min_{\theta} -[\nabla_{\theta} L_{\theta_{old}}(\theta)|_{\theta=\theta_{old}} \cdot (\theta - \theta_{old})]$$

$$\text{subject to } \frac{1}{2}(\theta_{old} - \theta)^T A(\theta_{old})(\theta_{old} - \theta) \leq \delta$$

- 其中：

$$L_{\theta_{old}}(\theta)|_{\theta=\theta_{old}} = E_{s\sim\rho_{\theta_{old}}, a\sim\pi_{\theta_{old}}}\left[\frac{\tilde{\pi}_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\theta_{old}}(s,a)\right]$$

$$A = E_{\theta_{old}}[\nabla^2 \log \pi_{\theta_{old}}]$$

- 最后的结果：

$$\theta = \theta_{old} + \beta d$$

$$A(\theta_{old})d = \nabla_{\theta} L_{\theta_{old}}(\theta)|_{\theta=\theta_{old}}$$

$$\beta = \sqrt{\frac{2\delta}{d^T A d}}$$

d是搜索方向，$d \approx A^{-1}g$

TRPO方法：还有一个技巧

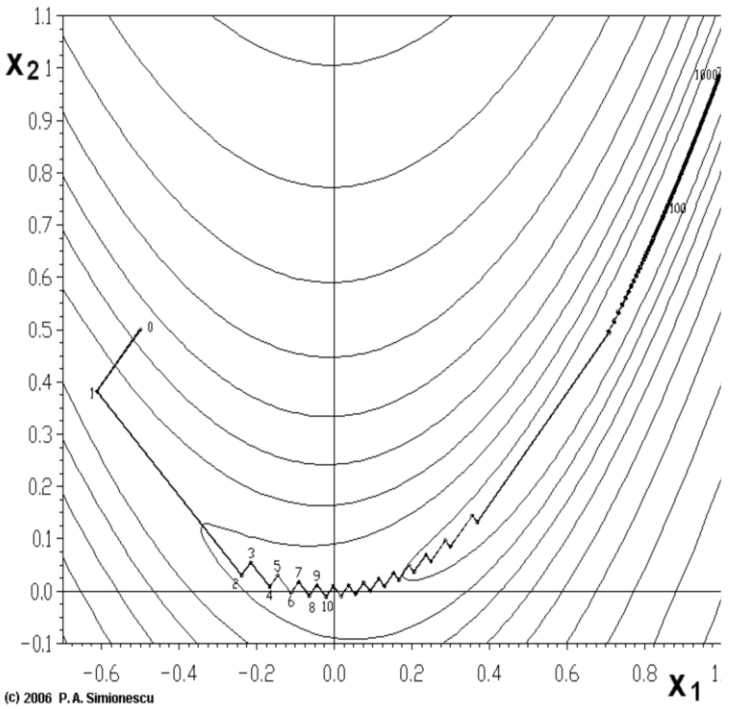
- 为了计算参数更新方向，先对约束中的KL散度作二次近似：
- $\overline{D}_{KL} \approx 1/2(\theta-\theta_{old})^T A(\theta-\theta_{old})$ ，
  - 其中A为Fisher信息矩阵(FIM)。
- 因为像神经网络表示策略时会有上千参数，这里如果要生成和求FIM的逆计算量会非常大，因此这里采用conjugate gradient算法近似求解。
- 接下来TRPO采用line search方法在前面选取的方向上再选取步长保证约束被满足。

插播：自然梯度

- 自然梯度：在保持$\|\Delta W\|^2$不变（即参数变化测度为固定值）的前提下，寻找一个最佳的方向，使得J(W+ΔW)最小（或最大）。
- 直观上，它代表着带有参数下似然的梯度的变化率。
- Amari等人利用“黎曼流形”的有关理论，证明在“黎曼流形”中的最佳方向（最速下降或上升）不是“负”常规梯度方向，而是“负”黎曼梯度方向。这个方向（自然梯度）可表示为$G^{-1}(\theta)\nabla_\theta J$。
- 其中的G为Fisher information matrix(FIM)，其形式为$G = E_\theta[\nabla_\theta \log P_\theta(X) \nabla_\theta \log P_\theta(X)^T]$ 。
- 它使得策略在相对于参数不太敏感的地方，步长大；而在敏感的地方，步长小。

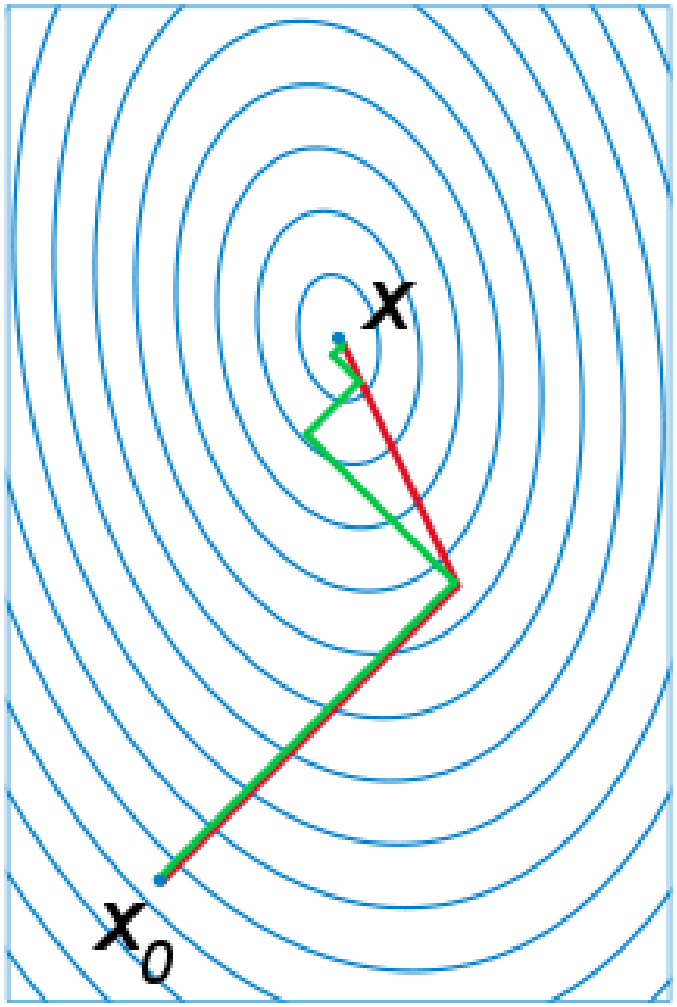
插播： conjugate gradient

- 在梯度下降法中，当次迭代的梯度方向也可能是和上次迭代梯度方向垂直，但和再之前的梯度方向就不垂直了，所以会有“之”形路线。
- 共轭梯度法（conjugate gradient）是介于最速下降法与牛顿法之间的一个方法，它仅需利用一阶导数信息，而且克服了最速下降法收敛慢的缺点，又避免了牛顿法需要存储和计算海森矩阵并求逆的缺点。



插播： conjugate gradient

- 共轭梯度法是一个典型的共轭方向法，它的每一个搜索方向是互相共轭的（每次消除了一个维度上的误差），而这些搜索方向d仅仅是负梯度方向与上一次迭代的搜索方向的组合。
- 共轭梯度法的优点是所需存储量小，具收敛速度快，稳定性高，而且不需要任何外部参数。



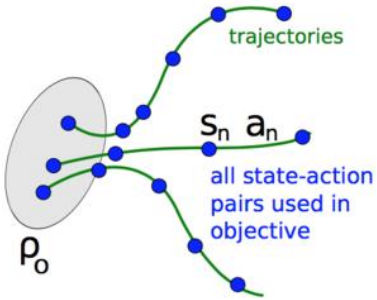
插播：line search

- 线搜索(line search)是一种迭代的求得某个函数的最值的方法。
- 对于每次迭代，线搜索会计算得到搜索的方向p_k以及沿这个方向移动的步长α_k。
- 大多数的线搜索方法都会要求p_k是下降方向(descent direction)，亦即需要满足以下条件：$p_k^T \nabla f_k < 0$，这样就能够保证函数$f(x)$沿着这个方向是下降的。一般来说，搜索方向是$p_k = -B_k^{-1} \nabla f_k$。
- 其中B_k是一个对称非奇异矩阵。在最速下降法中，B_k是单位矩阵I，在牛顿方法(Newton)中B_k则是海森(Hessian)矩阵$\nabla^2 f(x_k)$，在Quasi-Newton方法中通过迭代求得Hessian矩阵的近似矩阵。

TRPO方法：估计Q值

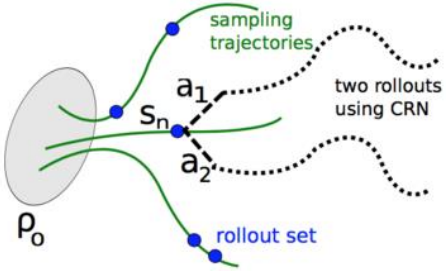
- 左，single path：通过模拟策略生成一组轨迹，并将所有状态-动作对（s_n，a_n）合并到目标中。
- 右，vine的说明：先生成一组“主干”轨迹，然后从到达状态的子集生成“分支” rollouts。对于每个状态s_n，执行多个动作（这里是a_1和a_2），并在每个动作之后执行rollout，使用公共随机数来减小方差。

Single Path



Can be run in the real world

Vines



Requires simulator, obtains extra variance reduction

TRPO方法的总结

- 为了解优化问题，TRPO：
  - 先线性近似目标函数，

$$L_{\pi}(\tilde{\pi}) = \eta(\pi) + E_{s\sim\rho_{\theta_{old}},a\sim\pi_{\theta_{old}}}[\frac{\tilde{\pi}_{\theta}(a|s_n)}{\pi_{\theta_{old}}(a|s_n)}A_{\theta_{old}}(s_n,a)]$$

  - 再二阶近似约束条件，

$$subject\ to\ \ \bar{D}_{KL}^{\rho_{\theta_{old}}}(\theta_{old},\theta) \le \delta$$

$$\text{subject to } \frac{1}{2}(\theta_{old}-\theta)^T A(\theta_{old})(\theta_{old}-\theta) \le \delta$$

  - 最后通过conjugate gradient算法和line search求解。

PPO方法

- PPO算法是对TRPO算法的改进，更易于实现，且数据效率更高。PPO尝试通过一阶优化的方法来解。
- TRPO方法中通过使用约束而非惩罚项来保证策略更新的稳定性，主要原因是作为惩罚项的话会引入权重因子，而这个参数难以调节。
- 与TRPO中用约束来限制策略更新幅度不同，PPO中采用了惩罚项，或者说正则项的做法，并提出了基于clipped probability ratio的目标函数。

PPO方法

- TRPO方法中最后的结论如下：

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

- 考虑用引入惩罚项使之成为非约束条件下的优化问题。

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

- TRPO使用硬约束而不是惩罚，因为很难选择在不同的问题中或甚至在单个问题中表现良好的单个beta值，其中特性随着学习过程而变化。
- 因此，为了实现模拟TRPO单调改进的一阶算法的目标，还需要进一步的修改。

PPO方法

$$\max_{\theta} E_{s\sim\rho_{\theta_{old}},a\sim\pi_{\theta_{old}}}[\frac{\tilde{\pi}_{\theta}(a|s_n)}{\pi_{\theta_{old}}(a|s_n)}A_{\theta_{old}}(s_n,a)]$$

- 首先定义probability ratio，代表两个策略概率的比率

$$r_t(\theta)=\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

- TRPO的代理目标函数就可表示为：

$$L^{CPI}(\theta)=\hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t]$$

- 从TRPO的约束条件中可以看出，算法对每一轮的优化都划定了置信区域，为保证优化的稳定性，每一步优化都不能超过这个范围。
- 直观的看，约束条件要求新的策略概率分布不能与旧的概率分布相差太大，可以认为，TRPO的约束条件相当于$r_t(\theta)$接近于1。

PPO方法

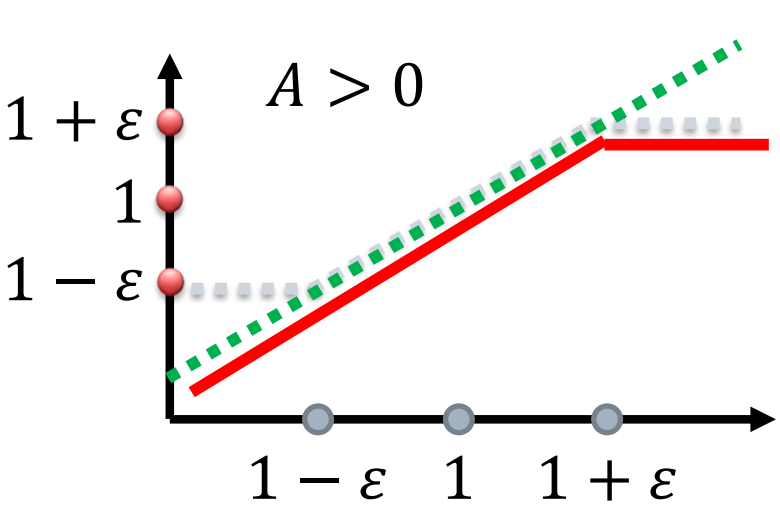
- 但这样的目标下策略更新有可能会很大。因此，clipping目标函数中加入了clip项:

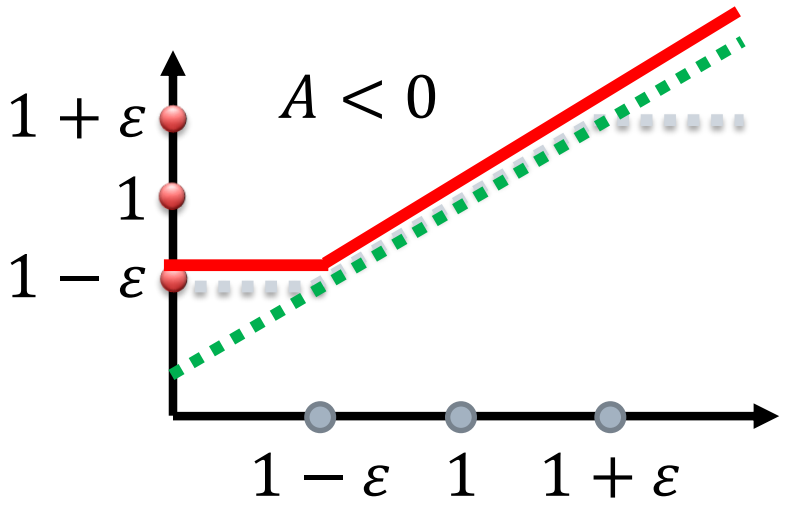
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

  - 其中ϵ为超参数，论文中说可以取0.2。
- 为确保能达到TRPO中的置信区域的效果，算法用两方面来保证：
  - clip项促使得r_t不偏离[1−ϵ,1+ϵ]所定义的区间
  - 算法会对clipped和unclipped目标进行最小化操作，当策略更新的偏移超出预定区间时，这个clip项就会产生影响。

PPO方法

- 由于是取最小值，
- 当A>0时， r_t的上限$(1+\epsilon)$会起作用；
  - 若$r_t>(1+\epsilon)$，则这时$(1+\epsilon)A<r_tA$
- 而当A<0时， r_t的下限$(1-\epsilon)$会起作用。
  - 若$r_t<(1-\epsilon)$，则这时$(1-\epsilon)A<r_tA$





PPO方法

- 另外一种替代（补充）方案（Adaptive KL Penalty）：对于KL散度进行惩罚。
- 目标函数是采用了自适应的KL惩罚项系数，它的目标函数为：

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

- 当策略更新前后KL散度小于预定值时，惩罚项系数β减小到原来的一半；当大于预定值时，系数β增加一倍。

- Compute $d = \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$
  - If $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$
  - If $d > d_{\text{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$

论文中说：其实验效果不及上面clipping目标函数。

PPO方法

- 如果使用神经网络，且它为策略和值函数共享了网络参数的话，则损失函数中应当结合策略优化目标和值函数误差项：

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

- 其中的c_1和c_2为系数，S和L_t^{VF}分别为策略熵奖励和值函数平方误差项。$(V_\theta(s_t) - V_t^{\text{targ}})^2$
- 整个PPO算法每次迭代中，先通过执行当前策略估计优势函数，然后通过优化代理函数更新策略参数。

PPO方法

- PPO还可以结合其他方法对来对目标公式中的优势函数进行逼近，例如：N步回报价值估计。

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

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

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

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方法

DPPO论文总结的

Algorithm 1 Proximal Policy Optimization (adapted from [8])

for $i \in \{1, \cdots, N\}$ **do**
  Run policy π_θ for T timesteps, collecting $\{s_t, a_t, r_t\}$
  Estimate advantages $\hat{A}_t = \sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t)$
  $\pi_{\text{old}} \leftarrow \pi_\theta$
  for $j \in \{1, \cdots, M\}$ **do**
    $J_{PPO}(\theta) = \sum_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)} \hat{A}_t - \lambda \text{KL}[\pi_{old}|\pi_\theta]$
    Update θ by a gradient method w.r.t. $J_{PPO}(\theta)$
  end for
  for $j \in \{1, \cdots, B\}$ **do**
    $L_{BL}(\phi) = -\sum_{t=1}^{T} (\sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t))^2$
    Update ϕ by a gradient method w.r.t. $L_{BL}(\phi)$
  end for
  if $\text{KL}[\pi_{old}|\pi_\theta] > \beta_{high} \text{KL}_{target}$ **then**
    $\lambda \leftarrow \alpha\lambda$
  else if $\text{KL}[\pi_{old}|\pi_\theta] < \beta_{low} \text{KL}_{target}$ **then**
    $\lambda \leftarrow \lambda/\alpha$
  end if
end for

Actor：PPO的目标函数

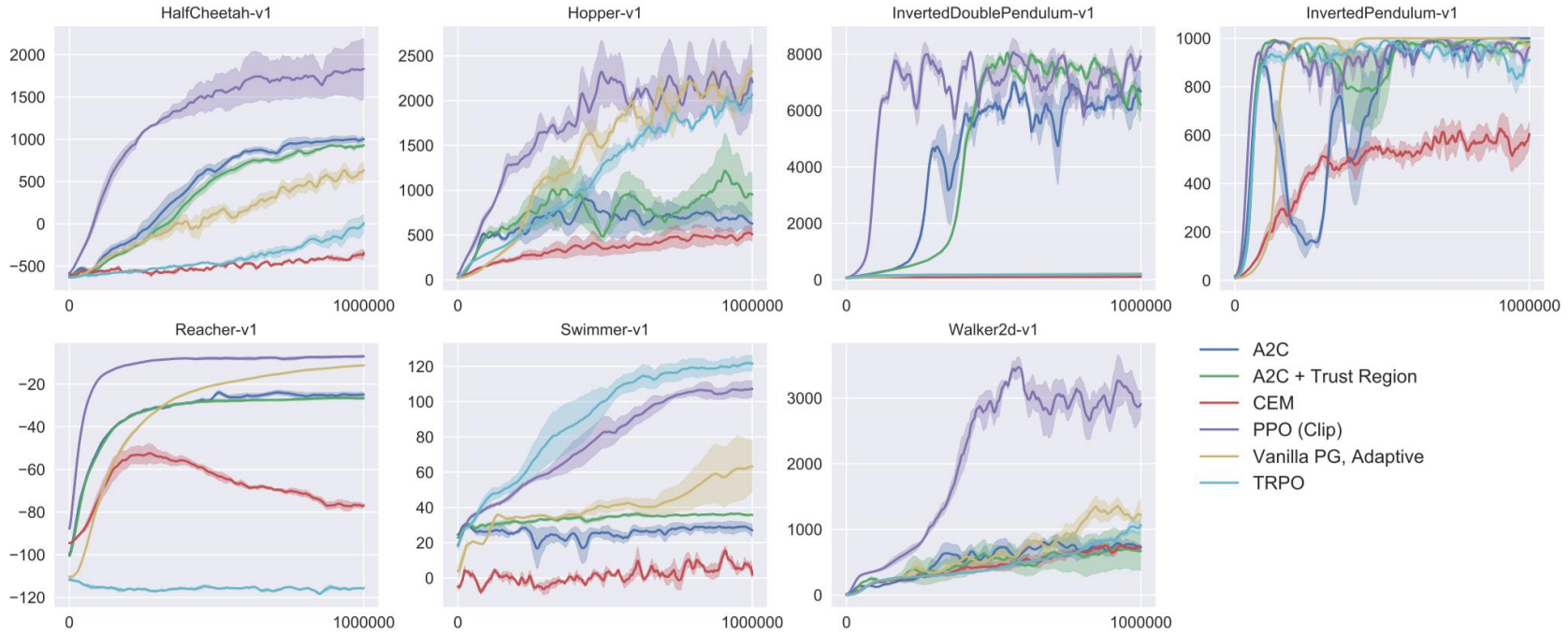
Critic：评价$V_\phi(s_t)$是否准确的baseline损失函数

Adaptive KL Penalty

PPO方法

https://www.sohu.com/a/160700396_473283

- 按照论文上的说法，在连续控制环境下，PPO几乎在各种游戏中都超过了以往的方法。
- 因为易于使用和表现良好，PPO已经成为OpenAI默认的强化学习算法。



DPPO方法

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n})$$

- DeepMind在论文《Emergence of Locomotion Behaviours in Rich Environments》中提出了分布式的PPO算法（DPPO）。
- DPPO的一个重点就是如何只利用简单的回报函数，通过丰富多变的环境来学习到稳定的行为。为了达到这个目的，需要改善强化学习算法的伸缩性。
- 在DPPO算法中，数据的收集和梯度的计算被分布到多个worker中，思想类似于A3C算法。
- 原始的PPO算法通过完整的回报和估计策略优势。
- 为了便于使用batch update的RNN，这里使用了K-step returns来估计策略优势。

$$\hat{A}_t = \sum_{i=1}^{K} \gamma^{i-1} r_{t+i} \quad + \quad \gamma^{K-1} V_\phi(s_{t+K}) - V_\phi(s_t).$$

DPPO方法

- DPPO算法分为两部分：chief和worker。
- chief部分从worker收集梯度，收集指定个数后，将它们的均值更新到总的参数。
- worker部分在每次迭代中依次做M步策略参数和B步值函数参数的更新；
  - 对于每个worker，每轮迭代中按当前策略执行T步，然后把它们按K个数据一份分好，对于每K步样本，估计优势函数A；
  - 之后分别通过梯度$\nabla_\theta J_{PPO}$和$\nabla_\phi L_{BL}$更新相应参数。
  - 另外它会根据当前策略和之前策略的KL距离是否超出区域调节目标函数中的系数λ。

DPPO方法

Algorithm 2 Distributed Proximal Policy Optimization (chief)

for $i \in \{1, \cdots, N\}$ **do**
  for $j \in \{1, \cdots, M\}$ **do**
    Wait until at least $W - D$ gradients wrt. θ are available
    average gradients and update global θ
  end for
  for $j \in \{1, \cdots, B\}$ **do**
    Wait until at least $W - D$ gradients wrt. ϕ are available
    average gradients and update global ϕ
  end for
end for

- W is the number of workers;
- D sets a threshold for the number of workers whose gradients must be available to update the parameters.
- M, B is the number of sub-iterations with policy and baseline updates given a batch of datapoints.
- T is the number of data points collected per worker before parameter updates are computed.
- K is the number of time steps for computing K-step returns

DPPO方法

Algorithm 3 Distributed Proximal Policy Optimization (worker)

for $i \in \{1, \cdots, N\}$ **do**
  for $w \in \{1, \cdots T/K\}$ **do**
    Run policy π_θ for K timesteps, collecting $\{s_t, a_t, r_t\}$ for $t \in \{(i-1)K, \ldots, iK-1\}$
    Estimate return $\hat{R}_t = \sum_{t=(i-1)K}^{iK-1} \gamma^{t-(i-1)K} r_t + \gamma^K V_\phi(s_{iK})$
    Estimate advantages $\hat{A}_t = \hat{R}_t - V_\phi(s_t)$
    Store partial trajectory information
  end for
  $\pi_{\text{old}} \leftarrow \pi_\theta$
  for $m \in \{1, \cdots, M\}$ **do**
    $J_{PPO}(\theta) = \sum_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)} \hat{A}_t - \lambda \mathrm{KL}[\pi_{old}|\pi_\theta] - \xi \max(0, \mathrm{KL}[\pi_{old}|\pi_\theta] - 2\mathrm{KL}_{target})^2$
    if $\mathrm{KL}[\pi_{old}|\pi_\theta > 4\mathrm{KL}_{target}$ **then**
      break and continue with next outer iteration $i+1$
    end if
    Compute $\nabla_\theta J_{PPO}$
    send gradient wrt. to θ to chief
    wait until gradient accepted or dropped; update parameters
  end for
  for $b \in \{1, \cdots, B\}$ **do**
    $L_{BL}(\phi) = -\sum_{t=1}^{T} (\hat{R}_t - V_\phi(s_t))^2$
    Compute $\nabla_\phi L_{BL}$
    send gradient wrt. to ϕ to chief
    wait until gradient accepted or dropped; update parameters
  end for
  if $\mathrm{KL}[\pi_{old}|\pi_\theta] > \beta_{high}\mathrm{KL}_{target}$ **then**
    $\lambda \leftarrow \tilde{\alpha}\lambda$
  else if $\mathrm{KL}[\pi_{old}|\pi_\theta] < \beta_{low}\mathrm{KL}_{target}$ **then**
    $\lambda \leftarrow \lambda/\tilde{\alpha}$
  end if
end for

ACER方法

- TRPO算法中采用了conjugate gradient方法求解，会有很多Fisher-vector乘积运算。因此在大规模问题中计算量十分大。
- 与TRPO方法通过约束条件限制策略更新不同的是，ACER算法中维护average policy network用于表示过往策略的移动平均，并保持策略更新不偏离这个均值。
- ACER将策略网络分解为两部分：分布f，以及生成该分布的统计量$\phi_\theta(x)$的深度神经网络。也就是说，给定$f(\cdot|\phi_\theta(x))$，策略完全由网络$\phi_\theta(x)$来表征。
- ACER的策略梯度记为$\hat{g}_t^{acer}$。

ACER方法

- ACER策略定义的梯度如下：

$$\begin{aligned}\widehat{g}_t^{\text{acer}} &= \bar{\rho}_t \nabla_\theta \log \pi_\theta(a_t|x_t)[Q^{\text{ret}}(x_t,a_t) - V_{\theta_v}(x_t)] \\ &\quad + \mathop{\mathbb{E}}_{a\sim\pi}\left(\left[\frac{\rho_t(a)-c}{\rho_t(a)}\right]_+ \nabla_\theta \log \pi_\theta(a|x_t)[Q_{\theta_v}(x_t,a) - V_{\theta_v}(x_t)]\right)\end{aligned}$$

- 其中，Q^{ret}是基于Retrace的Q值，ACER用Q^{ret}来估计Q^{π}。

$$Q^{\text{ret}}(x_t,a_t) = r_t + \gamma\bar{\rho}_{t+1}[Q^{\text{ret}}(x_{t+1},a_{t+1}) - Q(x_{t+1},a_{t+1})] + \gamma V(x_{t+1})$$

$$\bar{\rho}_t = \min\{c,\rho_t\} \text{ with } \rho_t = \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}$$

$$[x]_+ = x \text{ if } x > 0 \text{ and it is zero otherwise}$$

- 当c = ∞，上式恢复为异策略条件下的策略梯度；
- 当c = 0，上式恢复为基于Q值的Actor-Critic更新方法。

ACER方法

- 给定平均策略网络，ACER的信任区域更新包括两个阶段。在第一阶段，利用线性化的KL散度约束求解下列优化问题：

$$\min_z \frac{1}{2}||\hat{g}_t^{\text{acer}} - z||_2^2$$

$$s.t. \quad \nabla_{\phi_\theta} D_{KL}[f(\cdot|\phi_{\theta_a}(x_t))||f(\cdot|\phi_\theta(x_t))]^T z \leq \delta$$

- 该约束为线性的，因此整个问题可转为二次规划问题，有解析解，令：$k = \nabla_{\phi_\theta(x_t)} D_{KL}[f(\cdot|\phi_{\theta_a}(x_t)\|f(\cdot|\phi_\theta(x_t)]$

$$z^* = \hat{g}_t^{\text{acer}} - \max\{0, \frac{k^T \hat{g}_t^{\text{acer}} - \delta}{||k||_2^2}\}k$$

- 这个解表示在约束满足的情况下，则解即为 $\hat{g}_t^{\text{acer}}$ ；如果不满足，则沿k方向减少，从而使当前政策与平均政策网络之间的改变比率减少。

ACER方法

- 在第二阶段中，使用反向传播算法，通过链式法则$\frac{\partial \phi_\theta(x)}{\partial \theta} z^*$更新策略网络参数$\theta$。
- 和TRPO方法中用二次近似约束中的D_{KL}不同，ACER方法将约束写为线性形式。
- 因此可以通过解析解得到目标对于ϕ_θ的梯度，然后通过NN更新ϕ_θ对于θ的梯度。
- 从而避免了Fisher信息矩阵的计算，因此更适合大规模问题。

相关论文一览表

- 《Trust Region Policy Optimization》 Schulman, J., Levine, S., Moritz, P., Jordan, M. I. & Abbeel, P. 2015
- 《Proximal Policy Optimization Algorithms》 Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. 2017
- 《Emergence of Locomotion Behaviours in Rich Environments》 Heess, N. et al. 2017
- 《Sample Efficient Actor-Critic with Experience Replay》 Wang, Z. et al. 2017

参考资料

- https://blog.csdn.net/jinzhuojun/article/details/78007628
- https://zhuanlan.zhihu.com/p/26308073
- https://blog.csdn.net/Pony017/article/details/81146374
- https://blog.csdn.net/weixin_41679411/article/details/82421121

问答互动

在所报课的课程页面，
1、点击“全部问题”显示本课程所有学员提问的问题。
2、点击“提问”即可向该课程的老师和助教提问问题。

联系我们

小象学院：互联网新技术在线教育领航者

– 微信公众号：**小象学院**