强化学习原理

TRPO 和 PPO 论文笔记

姓名：石若川 学号：2111381 专业：智能科学与技术

1 Trust Region Policy Optimization(TRPO)

1.1 引言

TRPO 于 2015 年由 John Schulman 等人提出，是一种基于策略梯度方法的算法。一般的策略梯度算法存在以下缺点：

- 很难在整个优化过程选择一个时间步长，特别是由于状态和回报在改变统计特性
- 策略经常会过早地收敛到一个次优的几乎确定的策略

当学习步长不合适时，策略梯度方法很可能会崩溃。传统强化学习算法很难保证单调收敛，而 TRPO 却给出了一个能保证单调收敛的策略改善方法。

考虑一个带折扣的无限 MDP 过程 $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$。其中 ρ_0 为初始状态 s_0 的分布。定义 π 为在状态-动作下的随机策略，$\pi \in [0,1]$。折扣回报函数 $\eta(\pi)$ 为：

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t\right)\right]$$

其中，$s_0 \sim \rho_0\left(s_0\right), a_t \sim \pi\left(a_t \mid s_t\right), s_{t+1} \sim P\left(s_{t+1} \mid s_t, a_t\right)$。

值函数、动作值函数和优势值函数的定义如下：

$$\begin{aligned}
Q_\pi(s_t, a_t) &= \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right], \\
V_\pi(s_t) &= \mathbb{E}_{a_t, s_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right], \\
A_\pi(s, a) &= Q_\pi(s, a) - V_\pi(s)
\end{aligned}$$

Sham Kakade 于 2002 年提出了以下替换回报函数：

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

该替代回报函数将新的策略 $\tilde{\pi}$ 的回报函数拆分为旧的策略 π 的回报函数和新旧策略的回报差。若能保证新旧策略的回报差始终大于等于 0，则可以保证新的策略始终不比旧的策略差，进而保证策略单调

收敛。式1的证明如下：

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

将式1展开，可得：

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

使用确定性策略 $\tilde{\pi}(s)=\arg\max_a A_\pi(s,a)$ ，如果至少有一个状态-动作对的优势函数为正数，且对该状态有非零的访问概率，则还没有收敛到最优策略；否则，说明该策略已经达到最优。

在函数近似中，由于估计和近似误差，通常不可避免地会有一些状态的预期优势为负，即 $\sum_a \tilde{\pi}(a\mid s)A_\pi(s,a)<0$。$\rho_{\tilde{\pi}}(s)$ 的复杂性使得式2很难优化。因此，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 $\rho_\pi(s)$，原来的代价函数变为：

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

对任意模型参数 θ_0 有：

$$L_{\pi_{\theta_0}}(\pi_{\theta_0})=\eta(\pi_{\theta_0}),\nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)\big|_{\theta=\theta_0}=\nabla_\theta\eta(\pi_\theta)\big|_{\theta=\theta_0}$$

因此可以说明在小步长下，$L_\pi(\tilde{\pi})$ 与 $\eta(\tilde{\pi})$ 变化趋势一致，但我们无法确定这个小步长的取值范围 (信任区域)。

为解决这一问题，引入了 Kakade&Langford 的结论。新策略 π_{new} 可以写成包括旧策略 π_{old} 在内的混合策略：

$$\pi_{\text{new}}\left(a\mid s\right)=(1-\alpha)\pi_{\text{old}}\left(a\mid s\right)+\alpha\pi'(a\mid s) \tag{4}$$

其中 $\pi'=\operatorname{argmax}_\pi L_{\pi_{\text{old}}}(\pi)$。Kakade&Langford 证明了该新策略的回报函数有下界：

$$\eta\left(\pi_{\text{new}}\right)\geq L_{\pi_{\text{old}}}\left(\pi_{\text{new}}\right)-\frac{2\epsilon\gamma}{\left(1-\gamma\right)^2}\alpha^2,\quad \text{where } \epsilon=\max_s\left|\mathbb{E}_{a\sim\pi'(a|s)}\left[A_\pi(s,a)\right]\right| \tag{5}$$

该结论限制了策略必须为式4的混合策略。因此作者将其拓展到了一般随机策略。

1.2 一般随机策略

作者将策略梯度的步长 α 修改为策略 π 和 $\tilde{\pi}$ 之间距离的一种度量 (Total Variation Divergence，总变量散度 D_{TV}^{max})。其中，$D_{TV}(p\|q) = \frac{1}{2}\sum_i |p_i - q_i|$, 定义了离散概率分布 p, q 之间的一种距离。定义：

$$D_{\mathrm{TV}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{TV}(\pi(\cdot|s) \parallel \tilde{\pi}(\cdot|s)) \tag{6}$$

作者对式5中 ϵ 进行修改，用最大状态动作优势 $\max_{s,a} |A_\pi(s,a)|$ 替代状态期望优势。

令 $\alpha = D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}})$，将下界修改为：

$$\eta(\pi_{\mathrm{new}}) \geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2, \quad where\ \epsilon = \max_{s,a} |A_\pi(s,a)| \tag{7}$$

总变量散度和 KL 散度之间存在关系：$D_{TV}(p\|q)^2 \leq D_{\mathrm{KL}}(p\|q)$。

令 $D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{\mathrm{KL}}(\pi(\cdot \mid s) \| \tilde{\pi}(\cdot \mid s))$ 将式7写为：

$$\eta(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}), \quad where\ C = \frac{4\epsilon\gamma}{(1-\gamma)^2} \tag{8}$$

由此，可以通过 minorization-maximization(MM) 算法进行迭代优化。算法伪代码如下：

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

令 $M_i = L_{\pi_\xi}(\pi) - C D_{KL}^{\max}(\pi_i, \pi)$ 为下界函数。由于 $\pi_i + 1 = \mathrm{argmax}_\pi M_i(\pi)$, 则 $M_i(\pi_{i+1}) = \max_\pi M_i(\pi)$，即可通过迭代下界函数来使 $\eta(\tilde{\pi})$ 单调上升。

数学推导如下：

$$\eta(\pi_{i+1}) \geq M_i(\pi_{i+1})$$

$$\eta(\pi_i) = M_i(\pi_i)$$

$$therefore, \quad \eta(\pi_{i+1}) - \eta(\pi_i) \geq M_i(\pi_{i+1}) - M(\pi_i)$$

因此，通过在每轮迭代中最大化 M_i, 我们保证了 η 是非递减的。

1.3 参数化策略的优化

将策略参数化后，利用 θ 替代 π_θ，例如 $\eta(\theta) := \eta(\pi_\theta), L_\theta(\tilde{\theta}) := L_{\pi_\theta}(\pi_{\tilde{\theta}})$，以及$D_{\mathrm{KL}}(\theta\|\hat{\theta}) := D_{\mathrm{KL}}(\pi_\theta\|\pi_{\tilde{\theta}})$。根据式8，策略的更新方法为：

$$\theta_{new} = \arg\max_\theta \left[L_{\theta_{\text{old}}} \quad (\theta) - C D_{KL}^{\max} \quad (\theta_{\text{old}} \quad , \theta) \right]$$

该方法受惩罚系数 C 的影响，更新步长会很小，导致更新很慢。一个解决方法是将 KL 散度作为惩罚项的极值问题，转化为 KL 散度作为约束条件的优化问题，即：

$$\arg\max_\theta L_{\theta_{\text{old}}}(\theta) \quad \text{subject to } D_{\mathrm{KL}}^{\max}(\theta_{\text{old}}, \theta) \le \delta \tag{9}$$

根据式9，KL 散度在状态空间的每一点上都有约束。在实际求解过程中，求解如此多的约束是不实际的。因此，作者采用了启发式逼近法，使用 KL 散度的均值：

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

由此，得出了以下策略更新方法：

$$\arg\max_\theta L_{\theta_{\text{old}}}(\theta) \quad \text{subject to } \bar{D}_{\mathrm{KL}}^{\rho_{\theta_{old}}}(\theta_{\text{old}}, \theta) \le \delta \tag{10}$$

1.4 基于样本的目标和约束

将 $L_{\theta_{\text{old}}}(\theta)$ 的表达式代入，可得：

$$\arg\max_\theta \sum_s \rho_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a \mid s) A_{\theta_{\text{old}}}(s,a) \quad \text{subject to } \bar{D}_{\mathrm{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \le \delta \tag{11}$$

首先利用 $\frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}[\ldots]$ 替换 $\sum_s \rho_{\theta_{\text{old}}}(s)[\ldots]$。接着，将优势函数 A_{old} 替换为 Q_{old} 。由于 $A = Q - V$ ，且状态价值函数 V 在状态已知时不变，因此在求极值问题中该替换不影响结果。最后，用 q 表示采样分布，此时单个状态 s_n 对损失函数的贡献为：

$$\sum_a \pi_\theta(a \mid s_n) A_{\theta_{\text{old}}}(s_n, a) = \mathbb{E}_{a\sim q}\left[\frac{\pi_\theta(a \mid s_n)}{q(a \mid s_n)} A_{\theta_{\text{old}}}(s_n, a) \right]$$

最终优化问题转换为：

$$\begin{aligned} &\arg\max_\theta \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}, a\sim q\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{\text{old}}}(s,a) \right] \\ &\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}[D_{\mathrm{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s)\|\pi_\theta(\cdot \mid s))] \le \delta \end{aligned} \tag{12}$$

对式中的目标函数进行一阶近似，对约束条件进行二阶近似：

$$\mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{\text{old}}}(s,a) \right] \approx g^T(\theta - \theta_{old})$$

$$\mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\mathrm{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot \mid s)\|\pi_\theta(\cdot \mid s)\right)\right] \approx \frac{1}{2}(\theta - \theta_{old})^T \boldsymbol{F} (\theta - \theta_{old})$$

其中，g 表示目标函数的梯度

$$g = \nabla_\theta \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]\Big|_{\theta=\theta_{\text{old}}}$$

$\boldsymbol{F}$ 表示策略之间平均 KL 散度的 Hessian 矩阵，即 θ_{old} 的 Fisher 信息矩阵

$$\boldsymbol{F} = H\left[\mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot\mid s)\|\pi_\theta(\cdot\mid s))]\right]$$

式12转换为：

$$\theta_{new} = \arg\max_\theta g^T(\theta-\theta_{old})$$
$$\text{subject to } \frac{1}{2}(\theta-\theta_{old})^T\boldsymbol{F}(\theta-\theta_{old}) \le \delta$$

利用 KKT 条件可以得到迭代公式：

$$\theta_{new} = \theta_{old} + \sqrt{\frac{2\delta}{g^T\boldsymbol{F}^{-1}g}}\boldsymbol{F}^{-1}g$$

1.5 共轭梯度算法

一般来说，用神经网络表示的策略函数的参数数量巨大，计算和存储矩阵 $\boldsymbol{F}$ 的逆矩阵会耗费大量的内存资源和时间。TRPO 通过共轭梯度法回避了这个问题，它的核心思想是直接计算 $x = \boldsymbol{F}^{-1}g, x$ 为参数更新方向。假设满足 KL 距离约束的参数更新时的最大步长为 β，根据 KL 距离约束条件，有 $\frac{1}{2}(\beta x)^T\boldsymbol{F}(\beta x) = \delta,$，得到 $\beta = \sqrt{\frac{2\delta}{x^T\boldsymbol{F}x}}$。因此，此时参数更新方式为

$$\theta_{new} = \theta_{old} + \sqrt{\frac{2\delta}{x^T\boldsymbol{F}x}}x$$

1.6 线性搜索

由于 TRPO 算法在求解优化问题时，分别对目标函数和约束条件进行一阶近似和二阶近似，并非精准求解，因此，θ_{new} 可能未必比 θ_{old} 好，或未必能满足 KL 散度限制。因此，TRPO 在每次迭代的最后进行一次线性搜索，以确保找到满足条件。具体来说，就是找到一个最小的非负整数 i，使得按照

$$\theta_{new} = \theta_{old} + \alpha^i\sqrt{\frac{2\delta}{x^T\boldsymbol{F}x}}x$$

求出的 θ_{new} 依然满足 KL 散度限制，并且提升目标函数 L_{θ_k}，其中 $\alpha\in(0,1)$ 是一个决定线性搜索步长的超参数。

1.7 实验结果

作者在 MuJoCo 模拟器上 Swimmer、Walker 和 Hopper 三个任务上进行了机器人运动的实验，利用 Single TRPO、Vine TRPO 与 CEM、CMA、Natural Gradient 等方法进行对比，结果如图1.1所示。实验结果表明，TRPO 解决了所有问题并获得了最佳解决方案。

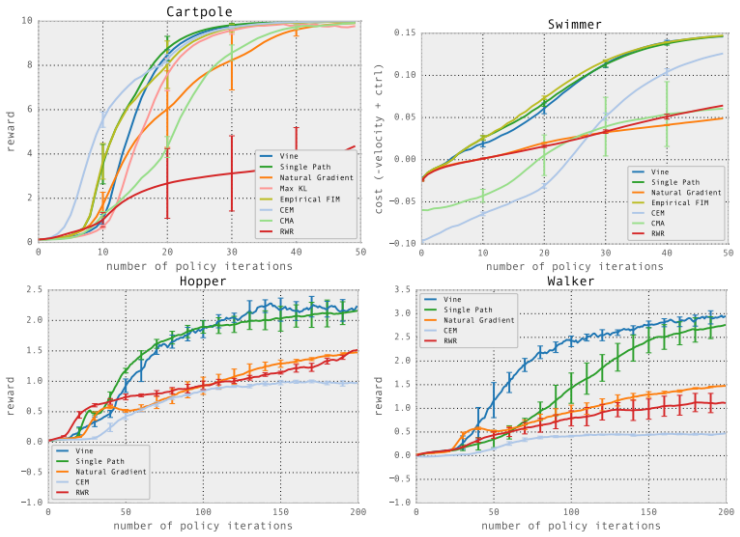

图 1.1: MuJoCo 学习结果

为了评估 TRPO 在具有复杂观测结果的任务中的表现，作者使用原始图像输入，训练了玩 Atari 游戏的策略。虽然 TRPO 仅在部分些游戏中胜过先前的方法，但它始终获得了较好的分数。与先前的方法不同，TRPO 并非专门为此任务而设计。将相同的策略搜索方法应用于诸如机器人运动和基于图像的游戏玩法等不同方法，证明了 TRPO 的普遍性。

	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

图 1.2: Atari 学习结果

2 Proximal Policy Optimization(PPO)

PPO 算法同样由 John Schulman 提出，和 TRPO 不同的是：TRPO 对目标函数采取一阶近似，约束条件采取二阶近似，然而 PPO 采用了一系列的一阶方法（Clip）。在效果相近的同时，PPO 在算法上更加简单。

2.1 TRPO 的惩罚形式

根据式12，将 KL 散度作为在目标函数中的惩罚项，公式如下：

$$\arg\max_{\theta} \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right] \tag{13}$$

然而，因为权重 β 难以选择和调整的问题，所以 TRPO 并没有采取惩罚的方式设置目标函数。

2.2 PPO 形式 1：裁剪替代目标函数

定义 $r_t(\theta)=\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$，易得 $r(\theta_{\text{old}})=1$。TRPO 的目标函数可以写为：

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

当没有约束时，该目标函数会使得策略被过大更新。因此需要对 $r_t(\theta)$ 过于偏离 1 的情况进行惩罚。

作者提出的方法是将 $r_t(\theta)$ 限制在 $[1-\epsilon,1+\epsilon]$ 范围内，其中 ϵ 为超参数，表示限制范围。PPO 的目标函数写为：

$$L^{CLIP}(\theta)=\hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t,\text{clip}(r_t(\theta),1-\epsilon,1+\epsilon)\hat{A}_t)\Big] \tag{14}$$

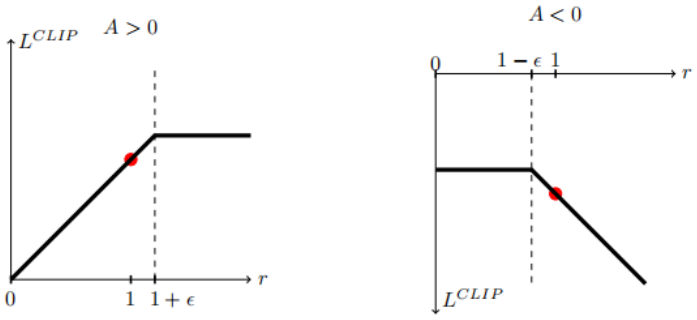

图 2.3: L^{CLIP}

2.3 PPO 形式 2：自适应 KL 惩罚系数

除了裁剪的方式外，作者基于式13提出了自适应 KL 惩罚系数的方法。经过实验验证，作者发现 KL 惩罚的性能弱于裁剪的方法。自适应 KL 惩罚系数的策略更新方法如下：

- 利用小批量的 SGD 的方法，优化 KL 惩罚目标

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

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

1.5、2 和 β 的初始值为超参数，算法对其并不敏感。

2.4 算法

作者将 PPO 的目标函数定义为：

$$L_t^{CLIP+VF+S}(\theta)=\hat{\mathbb{E}}_t\Big[L_t^{CLIP}(\theta)-c_1L_t^{VF}(\theta)+c_2S[\pi_\theta](s_t)\Big]$$

其中 c_1,c_2 为超参数，L_t^{VF} 为值函数的损失，S 为熵，用来促进探索。

在 T 个时间步中运行策略（其中 T 远远小于幕的长度），并利用收集到的样本进行更新。这种方法需要一个不超出时间步 T 的优势估计器：

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

在此基础上进行拓展，可以使用广义优势估计的截断版本：

$$\begin{aligned} &\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \\ &\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned}$$

使用固定长度轨迹段的 PPO 算法如下所示。每次迭代，N 个 Actor 中的每个 Actor 都会收集 T 个时间步的数据。然后，在这些 NT 个时间步的数据上构建损失，并在 K 幕内使用小批量 SGD 对其进行优化。

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

2.5 实验结果

作者同样在 MuJoCo 环境和 Atari 游戏中进行了测试。MuJoCo 的测试结果如图1.1所示，作者将 PPO 和 A2C、TRPO、CEM 等方法进行对比，实验发现发现在几乎所有连续控制环境中，PPO 都优于先前的方法。

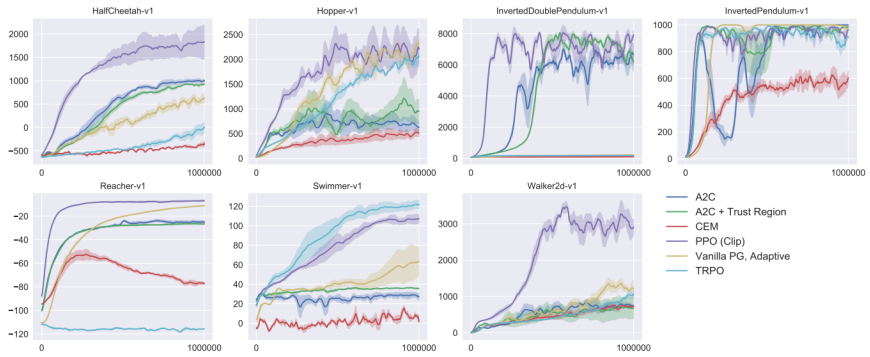

图 2.4: MuJoCo 实验结果

为展示 PPO 在高维连续控制问题上的表现，作者在一个涉及到 3D 仿真人类角色的问题集上进行训练，其中机器人必须奔跑、转向，并可能在被立方体袭击时站起来。这三种任务的学习曲线如图2.5所示。

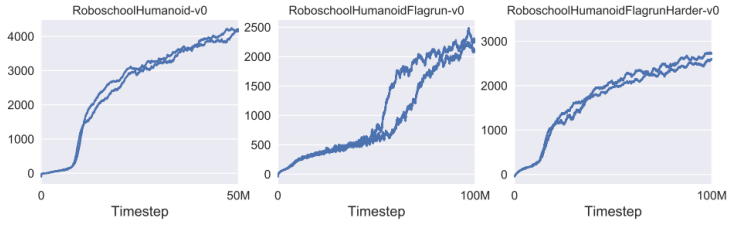

图 2.5: 3D 人类控制任务学习曲线

作者在 Atari 游戏中将 PPO 和 A2C、ACER 进行对比，实验结果如图2.6所示。

	A2C	ACER	PPO	Tie
(1) avg. episode reward over all of training	1	18	**30**	0
(2) avg. episode reward over last 100 episodes	1	**28**	19	1

图 2.6: 每种算法获胜的游戏数量

3 总结

TRPO 算法通过限制策略参数更新的大小，确保每次迭代后的策略不会变得过于不稳定。论文中展示了一个近似于该方法的实际算法，该算法在信任域约束条件下优化特定目标。TRPO 在一系列具有挑战性的策略学习任务中取得了良好的实验结果，优于先前的方法。

PPO 算法引入了近端策略优化方法，使用多轮随机梯度上升来执行每次策略更新。这些方法具有信任域方法的稳定性和可靠性，但实现起来要简单得多，只需要对基本策略梯度实现做出少量代码更改，适用于更一般的情况，并且具有更好的整体性能。