基于KL散度的策略优化

李建国[1] 赵海涛[1] 孙韶媛[2]
(华东理工大学信息科学与工程学院 上海 200237)[1] (东华大学信息科学与技术学院 上海 201620)[2]

摘 要 强化学习(Reinforcement Learning,RL)在复杂的优化和控制问题中具有广泛的应用前景。针对传统的策略梯度方法在处理高维的连续动作空间环境时无法有效学习复杂策略,导致收敛速度慢甚至无法收敛的问题,提出了一种在线学习的基于KL散度的策略优化算法(KL-divergence-based Policy Optimization,KLPO)。在Actor-Critic方法的基础上,通过引入KL散度构造惩罚项,将"新""旧"策略间的散度结合到损失函数中,以对Actor部分的策略更新进行优化;并进一步利用KL散度控制算法更新学习步长,以确保策略每次在由KL散度定义的合理范围内以最大学习步长进行更新。分别在经典的倒立摆仿真环境和公开的连续动作空间的机器人运动环境中对所提算法进行了测试。实验结果表明,KLPO算法能够更好地学习复杂的策略,收敛速度快,并且可获取更高的回报。

关键词 强化学习,KL散度,策略优化,连续动作空间
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.06.032

KL-divergence-based Policy Optimization

LI Jian-guo[1] ZHAO Hai-tao[1] SUN Shao-yuan[2]
(School of Information Science and Engineering,East China University of Science and Technology,Shanghai 200237,China)[1]
(School of Information Science and Technology,Donghua University,Shanghai 201620,China)[2]

Abstract Reinforcement learning has wide application prospects in dealing with the problem of complex optimization and control. Since traditional policy gradient method cannot learn the complex policy effectively in addressing with the environment with high-dimensional and continuous action space,that causes slow convergence rate or even non-convergence,this paper proposed an online KL-divergence-based policy optimization algorithm to solve this issue. Based on Actor-Critic algorithm,the KL-divergence is introduced to construct a penalty which adds the distance between "new" and "old" into policy loss function to optimization the policy update of Actor. Furthermore,the learning step is controlled by KL-divergence to ensure the policy update with maximum learning step within security region. On the experiment of Pendulum and Humanoid,simulation results show that KLPO algorithm can learn complex strategies better,converge faster and get higher returns.

Keywords Reinforcement learning,KL-divergence,Policy optimization,Continuous action space

1 引言

强化学习(RL)是机器学习的重要分支之一,不同于监督学习和无监督学习,它通过试错(Trail-and-error)的方式与环境进行交互、学习[1-2],强化从环境中所获得的回报。强化学习具有自主学习和在线学习的特点,在近几年受到了越来越多的关注[3-5]。

现今大部分的强化学习算法可以分为三大类[6]:1)无导数(Derivative-free)优化方法,将策略看作一个黑盒,无视其他信息,只以期望回报最大为目标来优化策略;2)策略迭代方法,通过迭代计算价值函数使策略收敛到最优;3)策略梯度方法,通过估计采样轨迹的期望回报的梯度来更新策略。

一般的无导数方法,比如交叉熵算法(Cross-Entropy Method,CEM),相比于其他算法更易于理解和实现,并且在大多数问题上都能取得更好的效果,但在交互环境复杂时表现欠佳[7]。

Wukkuams于1992年提出的REINFORCE算法是策略梯度(Policy Gradient,PG)方法的雏形。这种基于策略(Policy-based)的方法与基于值(Value-based)的策略迭代方法相比,不仅避免了值函数误差导致的策略退化,而且更加适用于解决连续动作空间问题。但是,其仍存在数据效率低和方差大的问题,导致训练困难。

到稿日期:2018-04-23 返修日期:2018-08-16 本文受国家自然科学基金(61375007),上海市科委基础研究项目(15JC1400600)资助。
李建国(1992-),男,硕士生,主要研究方向为模式识别、强化学习,E-mail:y30160642@mail.ecust.edu.cn;**赵海涛**(1974-),男,博士,教授,主要研究方向为模式识别、人工智能,E-mail:haitaozhao@ecust.edu.cn(通信作者);**孙韶媛**(1974-),女,博士,教授,主要研究方向为图像处理、计算机视觉等。

针对这些问题，Dale 在原始策略梯度方法中引入基于值的方法对值函数进行估计，提出了 Actor-Critic(AC)方法[8]，同时通过在估计的值函数的基础上减去基线(Baseline)来减小方差。但是，AC 算法依然存在收敛速度慢或者过早收敛的问题。在收敛速度慢的问题上，Silver 提出了确定性策略理论，并结合深度网络提出了深度确定性策略梯度算法[9](Deep Deterministic Policy Gradient，DDPG)，整个确定性策略的学习框架采用了 AC 算法，通过减少需要采样的数据来提高算法效率。但是，该方法依然存在无法将策略探索和改进集成到一个策略中的问题。

本文在 Actor-Critic 算法框架下，通过结合 KL 散度，提出了随机性的基于 KL 散度的策略优化方法(KLPO)。KL 散度是一种用来衡量两种概率分布间差别的度量。由于智能体生成的策略形式是动作空间的状态分布，因此 KL 散度可成为一种度量“新”“旧”策略间“距离”的自然方式。通过将 KL 散度结合到策略更新的损失函数中，构成一个“惩罚项”，即将“新”“旧”策略间的距离结合到损失函数中，以确保每次策略更新后，“新”“旧”策略的差距不是非常大，由此定义了一个新的损失函数来确保算法的稳定性。

大部分强化学习算法都采用固定步长更新参数，无法保证收敛速度，并且容易产生震荡，延长了收敛时间，降低了算法性能。因此进一步利用 KL 散度来控制算法更新的学习步长，实现训练过程中学习步长的自适应调整，使参数在由 KL 散度定义的一个合理范围内以最大学习步长进行更新，加快了算法收敛速度，提高了算法的鲁棒性。

在经典的倒立摆实验和公开的连续动作空间的机器人仿真实验中进行测试，KLPO 算法均能够快速有效地学习到策略并获得更高的奖励回报[1)]。

2　强化学习与 Actor-Critic

强化学习的问题即是一系列的决策问题，以最大化累积的奖励回报总和为目标。智能体(Agent)自主性地探索环境和感知环境，根据当前的环境状态采取动作并从环境中获得回报，在交互过程中通过不断学习来优化策略，以此得到最优策略[10]。智能体与环境交互的过程如图 1 所示。

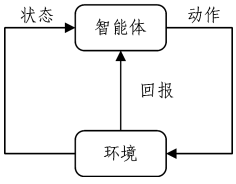

图 1　强化学习框架

Fig. 1　Framework of reinforcement learning

由于当前状态下的决策与之前的决策无关，因此强化学习问题可以看成是马尔科夫决策过程(Markov Decision Process，MDP)。通常将强化学习定义为元组(S,D,A,P,r,γ)，其中 S 为有限状态$\{s_1,s_2,\cdots,s_n\}$合集，D 为初始状态分布，A 为可选动作$\{a_1,a_2,\cdots,a_m\}$合集，$P:S\times A\times S\to R$ 为状态转换分布。$r:S\to R$ 为回馈函数，γ 为折扣因子[11]。若每回合进行 H 次交互，则过程记录为轨迹 $\iota:\{s_0,a_0,r_0,\cdots,s_H,a_H,r_H\}$。

2.1　策略梯度

策略梯度是一种直接逼近策略，将策略参数化为 π_θ，利用线性或非线性(神经网络)对策略进行表示[12]，寻找最优的参数 θ 来实现强化学习的目标：累积回报的期望 $E[\sum_{t=0}^{H}R(s_t|\pi_\theta)]$最大。通过直接对策略进行迭代计算，更新参数值，直到累积回报的期望最大，此时的参数所对应的策略即为最优策略[13]。随机策略梯度的计算公式为：

$$\nabla_\theta J(\pi_\theta)=E_{s,\rho^\pi,a,\pi_\theta}[\nabla_\theta\log\pi_\theta(a|s)Q^\pi(a|s)] \tag{1}$$

其中，$\pi_\theta(a|s)$表示智能体在状态 s 下基于策略 π_θ 采取动作 a 的概率；$Q(s_t,a_t)$为动作值函数，表示智能体在状态 s_t 采取动作 a_t 所获得的回报。

在利用当前策略 π_θ 采样 m 条轨迹后，可以利用这 m 条轨迹的平均经验对策略梯度进行逼近：

$$\nabla_\theta U(\theta)\approx G=\frac{1}{m}\sum_{i=1}^{m}\nabla_\theta\log P(\iota;\theta)R(\iota) \tag{2}$$

其中，$R(\iota)=\sum_{t=0}^{H}R(s_t,a_t)$表示轨迹 ι 的回报；$P(\iota;\theta)$表示轨迹 ι 出现的概率。策略梯度的更新采用的是最速下降法，其更新方式为：

$$\theta_{\text{new}}=\theta_{\text{old}}+\alpha\nabla_\theta U(\theta) \tag{3}$$

其中，α 为学习效率，通常来说学习效率越大，收敛速度越快，但容易产生震荡 。

2.2　Actor-Critic 方法

从策略梯度的直观解释可以看到，轨迹回报 $R(\iota)$就像是一个评价器(Critic)，该评价器的评价参数更新后，判断轨迹出现的概率应该变大还是变小，即 Critic 基于 Actor 的行为评判该行为的得分，Actor 根据 Critic 的评分修改行为的概率[14]。Actor-Critic 算法的框架如图 2 所示。

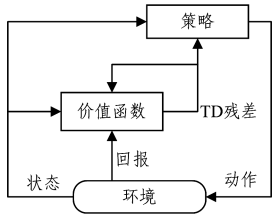

图 2　Actor-Critic 算法的结构

Fig. 2　Architecture of Actor-Critic algorithm

策略的参数调整幅度由估计回报 $R(\iota)$进行评价，因此在保持策略梯度大小不变的情况下，策略梯度可以进一步写为：

$$G=E[\sum_{t=0}^{\infty}\varphi_t\nabla_\theta\log\pi_\theta(a_t|s_t)] \tag{4}$$

其中，$\pi_\theta(a|s)$为 Actor，φ_t 为 Critic，因此式(4)就是一个广义的 Actor-Critic 框架。Actor 为策略函数，经常用神经网络来表示，因此被称为策略网络。Critic 为评价函数，对于大部分问题，φ_t 也常用神经网络进行逼近，因此称其为评价网络。当 φ_t 取优势函数(Advantage Function)$A_\pi(s,a)$时，不仅可以

1) https://github.com/wiatfor/KLPO

解决当 φ_t 直接取轨迹的累积回报时所造成的计算所得策略梯度方差过大的问题，还可以避免当 φ_t 为 TD 残差[15](Temporal Difference Error)时所造成的策略梯度存在大偏差的问题。此时，式(4)被称为优势表演家-评判者算法[16](Advantage Actor-Critic，A2C)。动作值函数、状态值函数、优势函数的定义分别为：

$$Q_\pi(s_t,a_t)=E_{s_{t+1},a_{t+1},\cdots}[\sum_{i=0}^{\infty}\gamma^i r(s_{t+i})] \tag{5}$$

$$V_\pi(s_t)=E_{a_t,s_{t+1},\cdots}[\sum_{i=0}^{\infty}\gamma^i r(s_{t+i})] \tag{6}$$

$$A_\pi(s,a)=Q_\pi(s,a)-V_\pi(s) \tag{7}$$

其中，$a_t\leftarrow\pi(a_t\mid s_t)$，$s_{t+1}\leftarrow P(s_{t+1}\mid s_t,a_t)$，$\gamma\in(0,1)$为折扣因子[17]。

3 结合 KL 散度的策略优化

KL 散度的表达形式为：

$$D_{\mathrm{KL}}(p\parallel q)=\sum_{a\in A}p(a)[\log p(a)-\log q(a)]$$

它是描述两个概率分布 p 和 q 的差异的一种方法[18]。在信息论中，$D_{\mathrm{KL}}(p\parallel q)$表示当用概率分布 p 来拟合 q 时产生的信息损耗，其中 p 表示真实分布，q 表示 p 的拟合分布。本文在 A2C 框架下结合 KL 散度对 Actor 部分的策略更新进行优化。

3.1 Actor 部分的策略优化

在强化学习中，优化的目标函数为期望回报(以及最大熵正则化)，表达形式为：

$$J(\theta;\tau)=\sum_{a\in A}\pi_\theta(a\mid s)r(a)-\tau H(\pi_\theta(a\mid s)) \tag{8}$$

其中，$r(a)$为回报函数；τ 决定正则化程度；$H(\pi_\theta)$是分布 π_θ 的熵，$H(\pi_\theta)=-\sum_{a\in A}\pi_\theta\log(\pi_\theta)$。

一般情况下使用梯度下降法来优化目标函数，但是由于其梯度的方差太大，导致优化 $J(\theta;\tau)$非常困难，因此定义一个在输出空间的分布：

$$\pi_\theta^*(a)=\frac{1}{Z}\exp(r(a)/\tau) \tag{9}$$

其中，$Z=\sum_{a\in A}\exp(r(a)/\tau)$。通过引出 $\pi_\theta(a)$与 $\pi_\theta^*(a)$分布间的 KL 散度来重新定义式(8)为：

$$D_{\mathrm{KL}}(\pi_\theta^*(a\mid s)\parallel\pi_\theta(a\mid s))=\frac{1}{\tau}J(\theta;\tau)+\mathrm{constant} \tag{10}$$

当 $\pi_\theta^*(a\mid s)=\pi_\theta(a\mid s)$时，$D_{\mathrm{KL}}(\pi_\theta^*\parallel\pi_\theta)$和 $J(\theta;\tau)$最小。在没有熵正则化，即 $\tau=0$ 时，π_θ^* 是一个 δ 分布：

$$\delta(a\mid a^*)=\begin{cases}1, & a=a^*\\ 0, & a\neq a^*\end{cases} \tag{11}$$

当 $\tau\to0$，$\delta(a\mid a^*)$等同于 π_θ^*。通过容许 τ 为非零，并在最大似然估计方向上优化 KL 散度，定义了一个 KL 散度惩罚项(KL Penalty)：

$$J_{\text{KL-P}}(\theta;\tau)=-\lambda D_{\mathrm{KL}}(\pi_\tau^*(a)\parallel\pi_\theta(a))-H(\pi_\tau^*) \tag{12}$$

由于神经网络的参数空间为黎曼空间，目标函数的最速下降方向是在参数变化测度为固定值下最小化目标函数的参数变化方向，因此该方向：$G^{-1}(\theta)\nabla_\theta J$，其中 G 为黎曼度量张量。在统计问题上，参数空间的黎曼结构为 Fisher Information Matrix(FIM)，其形式为：

$$F=E_\theta[\nabla_\theta\log P_\theta(X)\nabla_\theta\log P_\theta(X)^{\mathrm{T}}] \tag{13}$$

对于参数 θ^* 及其领域的一点 θ，$D_{\mathrm{KL}}(\theta^*,\theta)=0$，当 $\theta=\theta^*$ 时 KL 散度达到最小，因此 KL 散度相对于参数 θ 的一阶导数为零，用泰勒公式展开到二阶则只剩下二次项，该二次项中的 Hessian 矩阵即为 FIM。

如果要参数更新前后其表征分布间的 KL 距离为定值，则可以用 FIM 替代上面的度量张量，参数更新的最速下降方向就由传统梯度 $\nabla_\theta J$ 变为 $F_\theta^{-1}\nabla_\theta J$，即 FIM 的逆乘以标准梯度。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。为了计算参数更新方向，对惩罚项中的 KL 散度作二次近似：

$$D_{\mathrm{KL}}\approx\frac{1}{2}(\theta-\theta_{\mathrm{old}})^{\mathrm{T}}F(\theta-\theta_{\mathrm{old}}) \tag{14}$$

其中，F 为 FIM，代表相应参数下似然梯度的变化率。

本文将 KL 散度惩罚项与 A2C 中的 Actor 更新相结合，把惩罚项作为一个约束转换到损失函数中来控制策略的更新幅度，从而提出了基于 KL 散度的策略优化算法(KLPO)，重新定义了策略更新的损失函数：

$$J_{\mathrm{KLPO}}(\theta)=\log(\pi_\theta(a\mid s))\times A_\pi(s,a)-H(\pi_{\theta_{\mathrm{old}}})-\lambda D_{\mathrm{KL}}(\pi_{\theta_{\mathrm{old}}}(a\mid s)\parallel\pi_\theta(a\mid s))-C \tag{15}$$

其中，λ 为 D_{KL} 的自适应权重因子，其随着策略的更新而调整。

$$\lambda=\begin{cases}\max(1/35,\lambda/1.5), & D_{\mathrm{KL}}<KL\_target/2\\ \min(35,1.5\times\lambda), & D_{\mathrm{KL}}>2\times KL\_target\end{cases} \tag{16}$$

$$C=[\max(0,D_{\mathrm{KL}}-KL\_target)]^2 \tag{17}$$

学习步长 lr 由“新”“旧”策略间的距离进行自适应调整。

$$lr=\begin{cases}2\times lr, & D_{\mathrm{KL}}<KL\_target/2\\ lr/2, & D_{\mathrm{KL}}>2\times KL\_target\end{cases} \tag{18}$$

KL_target 为设定的超参数。在进行策略更新时，通过 KL_target 来控制是否终止此次更新，当 $D_{\mathrm{KL}}>4\times KL\_target$ 时提前结束更新，并直接利用此时的策略进入下一回合的测试，与环境进行交互。

3.2 Critic 更新

学习过程是在线策略(On-policy)：Critic 必须了解和评价 Actor 当前采用的策略。Critic 作为评价函数，基于 Actor 所选择的动作评判该动作的得分，通常用神经网络来逼近，其参数用 ω 表示。Critic 更新的损失函数采用 TD 残差，并使用 Adam 优化器通过梯度下降来更新 Critic 网络参数。值函数的网络更新为：

$$\begin{cases}\delta\leftarrow V^\pi(s_{t+1},\omega)-V^\pi(s_t,\omega)\\ \omega\leftarrow\omega+\beta\delta\ \nabla_\omega V^\pi(s_{t+1},\omega)\end{cases} \tag{19}$$

3.3 KLPO 算法流程

KLPO 算法的流程如算法 1 所示。

算法 1 KLPO 算法

初始化环境状态

初始化参数

Repeat(对于每一回合)

Repeat(对于每一回合的每一步) # 与环境交互

(a)在状态 s_t 下根据策略 π_θ 输出的分布随机采样动作 a_t

(b)执行动作 a_t，收集回馈信息$\{r_t,s_{t+1}\}$

计算优势函数 $A_\pi(s,a)$和 $\log(\pi_\theta(a|s))$

将当前策略 π_θ 赋值予 $\pi_{\theta_{old}}$

Repeat(M) ＃M 为 Actor 每次更新的迭代次数

$J_{REPO}(\theta)=\log(\pi_\theta(a|s))\times A_\pi(s,a)-\lambda D_{KL}(\pi_{\theta_{old}}(a|s)\parallel\pi_\theta(a|s))-H(\pi_{\theta_{old}})-[\max(0,D_{KL}-KL\_target)]^2$

使用 Adam 优化器更新参数 θ

If $D_{KL}>4\times KL\_target$ 则

终止循环

Repeat(N) ＃N 为 Critic 每次更新的迭代次数

$L(\omega)=-\sum_{t=1}^{T}(r_t+\gamma V^\pi(s_{t+1},\omega)-V^\pi(s_t,\omega))$

使用 Adam 优化器更新参数 ω

If $D_{KL}<KL\_target/2$ 则

$lr\leftarrow 2\times lr$

If $D_{KL}>2\times KL\_target$ 则

$lr\leftarrow lr/2$

输出最终策略 π_θ

4　仿真实验和结果分析

为验证 KLPO 算法的有效性，在 Open AI Gym 测试平台上对其进行实验，并在相同环境下分别采用 Advantage Actor-Critic(A2C)算法、交叉熵算法(CEM)和 DDPG 作为对比进行仿真实验。

4.1　实验环境及参数设置

本文选取了 2 个状态空间维度不同的实验环境，分别是状态空间维度为 3 的倒立摆仿真环境，以及状态空间维度为 376 的仿真机器人运动环境。状态空间维度的大小间接地反映了环境的复杂程度。采取的动作空间都为连续集。

经典的倒立摆仿真环境如图 3 所示。该实验内容是控制操控钟摆的扭矩，使钟摆从自然的下垂状态直立起来并保持平衡。扭矩用来限制智能体直接摆动钟摆，操控的关节个数为 1。在 Mujoco(Multi Joint Dynamics with Contact)仿真环境中进行机器人运动实验来训练机器人行走，选取的仿真机器人模型为 Humanoid-v1，如图 4 所示。机器人的状态是它们的广义位置和速度，由各个关节的力矩进行控制；其状态空间为 376 维，操控关节为 17 个。高维度以及由于接触而产生的非光滑性，使得这项实验非常具有挑战性。该实验的目标是控制各个关节的扭矩使机器人模型达到目标状态，在保持不摔倒的情况下沿着 X 轴方向向前移动。当机器人偏离轨道中心时，对其加上一个小的惩罚力矩，如果在每个时间步长上机器人模型没有摔倒，则再加上一个小的奖励回报[19-20]。在一个回合内移动的距离越远，则获得的回报越高；如果机器人摔倒，则回合结束。

图 3　倒立摆

Fig. 3　Pendulum

图 4　Humanoid-v1

Fig. 4　Humanoid-v1

在算法模型的训练过程中，如果将值网络和策略网络的学习效率设置得过大，则会导致最终学到的策略波动太大，无法收敛；如果设置得过小，则会增加训练时间，并影响实验结果。折扣因子 $\gamma(\gamma<1)$反映了智能体对未来所获取回报的重视程度，γ 越大表明越重视将来所获取的回报。通过多次实验来测试调整这些超参数，以获取最合适的数值，使最终的目标结果最大化。超参数的具体设置如表 1 和表 2 所列。

表 1　倒立摆实验的超参数设定

Table 1　Hyper parameters setting of Pendulum experiment

实验次数 n	时间步长 len	值网络学习效率 α	策略网络学习效率 β	折扣因子 γ	更新步长 b
500	200	0.001	0.002	0.95	36

表 2　Humanoid-v1 实验的超参数设定

Table 2　Hyper parameters setting of Humanoid-v1 experiment

实验次数 n	值网络学习效率 α	策略网络学习效率 β	折扣因子 γ	更新回合 b
100 000	0.009	0.01	0.99	20

为了保证策略每次更新都有足够的训练数据，在倒立摆实验中，每回合运行 36 个时间步长后进行算法更新；在 Humanoid-v1 实验中，每采集 20 回合的数据后进行算法更新。

由于环境的复杂度不同，为了保证智能体能够有效地学习到最优策略，在倒立摆实验中使智能体与环境进行 500 回合的交互，每回合运行 200 个时间步长结束；在 Humanoid-v1 实验中进行 10 万回合的交互。策略神经网络的输出是在给定状态下动作的状态分布，然后智能体从动作状态分布中随机采样，并将其作为执行动作应用到环境中。为了在学习的开始阶段使算法更倾向于“探索”(Exploration)，将策略中的各个动作初始化为服从均匀分布，在每个状态下每个动作被选取的概率相同，此时策略的熵也最大。随着回合数的增加，算法的更新次数增多，在特定状态下能够获得更高回报的动作的概率得到提升，这时算法更倾向于“利用”(Exploitation)，策略的熵也不断减小。

4.2　实验结果与分析

在倒立摆实验中，采用 A2C 算法进行对比实验，描述实验结果的指标为每回合执行完 200 个时间步长后的累积回报，其结果如图 5 所示。

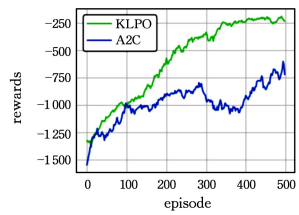

图 5　倒立摆实验

Fig. 5　Pendulum experiment

从图 5 可以看出，在倒立摆实验中，KLPO 优于 A2C 算法，在最后可以取得高于 A2C 算法 300 的总回报优势；且 KLPO 策略比 A2C 算法更加稳定。

在 Humanoid-v1 实验中，采用 A2C，CEM，DDPG 作为对比算法。在描述实验结果的指标中，最重要的是每个回合所

获得的累积回报，其次为机器人模型每个回合运行的平均时间步长，结果分别如图 6 和图 7 所示。

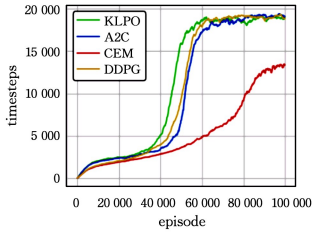

图 6 运行时间步长

Fig. 6 Running time-steps

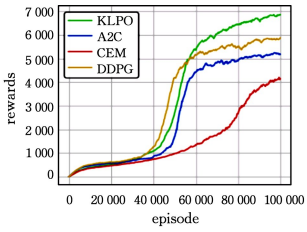

图 7 累积总回报

Fig. 7 Episode total rewards

Humanoid-v1 实验中，KLPO 同样取得了优异的效果。图 6 给出了每回合运行的平均时间步长，即机器人模型在每个回合保持不摔倒的情况下所执行的动作次数。虽然 A2C，DDPG 的效果与 KLPO 效果近似，但是能够保持不摔倒并执行相同的动作次数，并不代表其能够学习到有效的步伐和行走更远的距离来获得更高的总回报。

图 7 给出了每回合所获得的累积总回报。可以明显地看出，KLPO 能够使机器人模型更好地学习到正确、有效的行走步伐，从而获得远高于其他算法的奖励回报。在状态空间维度只有 3 的倒立摆实验中，智能体与环境进行 400 回合的交互学习后，KLPO 即达到了收敛状态，表明了智能体已经学习到了用最少的动作将钟摆直立起来，以此来获取更高的回报；而在状态空间维度为 376 的 Humanoid-v1 实验中，智能体与环境进行 95 000 回合的交互学习后 KLPO 才达到收敛状态，这反映了环境的状态空间维度的提升，增加了未知环境状态的数量，加大了环境的复杂程度，导致 KLPO 的学习速度和收敛速度减缓。

为了保证有足够的数据进行训练，每采集 20 回合的训练样本后进行一次算法更新。图 8 和图 9 分别给出了进行 20 个回合所取得的最大折扣回报与平均折扣回报，该结果也侧面证实了 KLPO 算法的优势性。CEM 是无导数算法，因此其样本复杂度与参数的个数不匹配，并且在较复杂的问题上表现不佳。KLPO 仅通过使用少量的先验知识、一般策略和简单的奖励函数，即可使 Humanoid-v1 模型学习到所有的步伐；而大多数学习运动模型的先验方法，通常需要手工构建对模型的平衡和步伐有明确表示的策略。

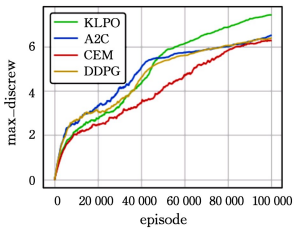

图 8 最大折扣回报

Fig. 8 Maximum discount reword

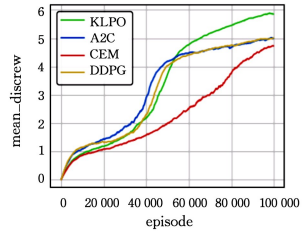

图 9 平均折扣回报

Fig. 9 Mean discount reword

图 10 给出了 KLPO 训练过程中自适应学习步长的变化趋势。学习步长根据策略的收敛速度和“新”“旧”策略网络间的“差距”不断地调整，确保神经网络参数每次以最大学习步长进行更新。图 11 给出了“新”“旧”策略网络的 KL 散度，KLPO 在相对合理的范围内进行最大幅度的更新，加快了算法的收敛速度。

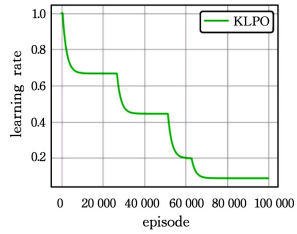

图 10 KLPO 的学习效率

Fig. 10 KLPO learning rate

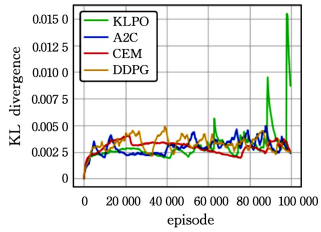

图 11 KL 散度

Fig. 11 KL divergence

图 12 给出了策略网络熵的变化趋势。熵能判断动作选择决策的不确定性，随着算法的收敛，策略网络不断向最优策略接近，最终在给定的一个机器人模型状态下，策略网络输出的动作行为状态分布中最优动作所占的比重远远高于其他动作所占的比重。因此，随着算法的不断更新，最优动作的概率增加。

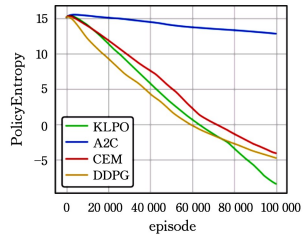

图 12 策略网络的熵

Fig. 12 Entropy of policy network

图 13 给出了策略网络的损失函数值。策略网络部分使用 Adam 优化器不断更新迭代损失函数的参数，以减小损失函数的值。从图 13 中可以看出，KLPO 在训练后可以更好地学习到策略来优化损失函数，使其达到最小。表 3 列出了算法在 Humaoid-v1 实验中进行 10 万回合交互和更新后各参数指标的数值。其中，*Rew* 为累积总回报；*M-dr* 为平均折扣回报；*P-loss* 为策略的损失值；*P-entr* 为策略网络的熵；*V-loss* 为值网络的损失值。

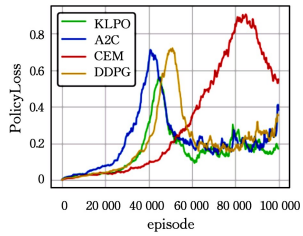

图 13 策略网络损失值

Fig. 13 Loss of policy network

表 3 Humanoid-v1 实验中参数指标的数值

Table 3 Parameter values in Humanoid-v1 experiment

algorithm	*Rew*	*M-dr*	*P-loss*	*P-entr*	*V-loss*
A2C	5283	5.12	0.390	12.73	0.027
CEM	4195	4.79	0.536	−4.311	—
DDPG	5908	5.06	0.388	−4.949	0.029
KLPO	**6890**	**5.81**	**0.182**	**−8.755**	**0.014**

通过以上对比仿真实验以及实验结果的分析可知，KLPO 在连续动作空间、高维度和奖励稀疏的复杂环境下能取得优异的效果。KLPO 通过利用损失函数控制“新”“旧”策略间的“距离”以及自适应调节步长来加快策略的学习速度，从而提高了学习的鲁棒性。

结束语　本文基于 Actor-Critic 方法处理连续动作空间的优势[21]，将其与 KL 散度相结合，提出了 KLPO 算法。通过引入 KL 散度构造惩罚项，并将其结合到 Actor 更新的损失函数中，对 Actor 部分的策略更新进行优化；进一步利用 KL 散度控制策略网络的学习步长，从而提升算法的学习效率，其 Critic 部分采用传统的在线选择时间差分学习[22]。KLPO 算法将“新”“旧”策略间的距离体现在策略的损失函数中，以引导策略的更新，并通过控制策略的更新步长来确保算法每次都能够在一个稳定的范围内以最大步长进行更新，以此提升算法的稳定性，加快收敛速度。

本文研究工作主要集中于 Actor 部分的优化，并在实验中取得了优异的效果，但没有对 Actor-Critic 方法的另一个核心 Critic 进行深入研究。Critic 评估的状态估计值的好坏将直接影响 Actor 的策略更新。下一步的工作主要是改进 Critic 部分，让 Critic 更好地评价、指引 Actor 的更新，以此来进一步提升算法的性能。

参 考 文 献

[1] MOUSAVI S S, SCHUKAT M, HOWLEY E. Deep Reinforcement Learning: An Overview[C]// Sai Intelligent Systems Conference. Cham: Springer, 2016: 426-440.

[2] WU J, HE H, PENG J, et al. Continuous reinforcement learning of energy management with deep Q network for a power split hybrid electric bus[J]. Applied Energy, 2018, 222: 799-811.

[3] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-532.

[4] SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of Go with deep neural networks and tree search. [J]. Nature, 2016, 529(7587): 484-489.

[5] WANG Z, SCHAUL T, HESSEL M, et al. Dueling network architectures for deep reinforcement learning[C]// Proceedings of International Conference on Machine Learning. PMLR, 2016: 1995-2003.

[6] DUAN Y, CHEN X, HOUTHOOFT R, et al. Benchmarking deep reinforcement learning for continuous control[C]// International Conference on International Conference on Machine Learning. JMLR. org, 2016: 1329-1338.

[7] SONG R, LEWIS F L, WEI Q. Off-Policy Integral Reinforcement Learning Method to Solve Nonlinear Continuous-Time Multiplayer Nonzero-Sum Games [J]. IEEE Transactions on Neural Networks & Learning Systems, 2016, 28(3): 704.

[8] GU S, HOLLY E, LILLICRAP T, et al. Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates[C]// International Conference on Robotics and Automation. New York: IEEE Press, 2017: 3389-3396.

[9] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[J]. Computer Science, 2015, 8(6): A187.

[10] PANNE M V D, PANNE M V D, PANNE M V D, et al. DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning[J]. Acm Transactions on Graphics, 2017, 36(4): 41.

[11] YUAN C L, RADULESCU A, DANIEL R, et al. Dynamic Interaction between Reinforcement Learning and Attention in Multidimensional Environments[J]. Neuron, 2017, 93(2): 451-463.

[12] ZHAO D, WANG B, LIU D. A supervised Actor-Critic approach for adaptive cruise control[J]. Soft Computing, 2013, 17(11): 2089-2099.

[13] THOMAS P S, BRUNSKILL E. Data-efficient off-policy policy evaluation for reinforcement learning[C]// International Conference on Machine Learning. JMLR. org, 2016: 2139-2148.

[14] CHEN X G, GAO Y, FAN S G, et al. Kernel-Based Continous-Action Actor-Critic Learning[J]. Patten Recognition and Artificial Intelligence, 2014, 27(2): 103-110. (in Chinese)
陈兴国，高阳，范顺国，等. 基于核方法的连续动作 Actor-Critic 学习[J]. 模式识别与人工智能，2014，27(2)：103-110.

[15] VAMVOUDAKIS K G, LEWIS F L. Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem[J]. Automatica, 2010, 46(5): 878-888.

[16] LEVINE S, FINN C, DARRELL T, et al. End-to-end training of deep visuomotor policies[J]. Journal of Machine Learning Research, 2015, 17(1): 1334-1373.

[17] JOEL D, NIV Y, RUPPIN E. Actor-critic models of the basal ganglia: new anatomical and computational perspectives [J]. Neural Networks, 2002, 15(4): 535-547.

[18] FILIPPI S, CAPPÉ O, GARIVIER A. Optimism in reinforcement learning and Kullback-Leibler divergence[C]// Communication, Control, and Computing. IEEE, 2011: 115-122.

[19] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[J]. Computer Science, 2015, 8(6): A187.

[20] SCHULMAN J, LEVINE S, MORITZ P, et al. Trust Region Policy Optimization[C]// Proceedings of International Conference on Machine Learning. PMLR, 2015: 1889-1897.

[21] YUAN C L, RADULESCU A, DANIEL R, et al. Dynamic Interaction between Reinforcement Learning and Attention in Multidimensional Environments[J]. Neuron, 2017, 93(2): 451-463.

[22] CHEN X, YANG G, WANG R. Online Selective Kernel-Based Temporal Difference Learning[J]. IEEE Transactions on Neural Networks & Learning Systems, 2013, 24(12): 1944-1950.