软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(11):5191−5204 [doi: 10.13328/j.cnki.jos.006740]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

融合引力搜索的双延迟深度确定策略梯度方法*

徐平安[1], 刘 全[1,2,3,4], 郝少璞[1], 张立华[1]

[1](苏州大学 计算机科学与技术学院, 江苏 苏州 215006)

[2](软件新技术与产业化协同创新中心 (南京), 江苏 南京 210093)

[3](符号计算与知识工程教育部重点实验室 (吉林大学), 吉林 长春 130012)

[4](江苏省计算机信息处理技术重点实验室 (苏州大学), 江苏 苏州 215006)

通信作者: 刘全, E-mail: quanliu@suda.edu.cn

摘 要: 近年来, 深度强化学习在复杂控制任务中取得了令人瞩目的效果, 然而由于超参数的高敏感性和收敛性难以保证等原因, 严重影响了其对现实问题的适用性. 元启发式算法作为一类模拟自然界客观规律的黑盒优化方法, 虽然能够有效避免超参数的敏感性, 但仍存在无法适应待优化参数量规模巨大和样本使用效率低等问题. 针对以上问题, 提出融合引力搜索的双延迟深度确定策略梯度方法 (twin delayed deep deterministic policy gradient based on gravitational search algorithm, GSA-TD3). 该方法融合两类算法的优势: 一是凭借梯度优化的方式更新策略, 获得更高的样本效率和更快的学习速度; 二是将基于万有引力定律的种群更新方法引入到策略搜索过程中, 使其具有更强的探索性和更好的稳定性. 将 GSA-TD3 应用于一系列复杂控制任务中, 实验表明, 与前沿的同类深度强化学习方法相比, GSA-TD3 在性能上具有显著的优势.

关键词: 深度强化学习; 元启发式算法; 引力搜索; 确定策略梯度; 策略搜索

中图法分类号: TP18

中文引用格式: 徐平安, 刘全, 郝少璞, 张立华. 融合引力搜索的双延迟深度确定策略梯度方法. 软件学报, 2023, 34(11): 5191–5204. http://www.jos.org.cn/1000-9825/6740.htm

英文引用格式: Xu PA, Liu Q, Hao SP, Zhang LH. Twin-delayed-based Deep Deterministic Policy Gradient Method Integrating Gravitational Search. Ruan Jian Xue Bao/Journal of Software, 2023, 34(11): 5191–5204 (in Chinese). http://www.jos.org.cn/1000-9825/6740.htm

Twin-delayed-based Deep Deterministic Policy Gradient Method Integrating Gravitational Search

XU Ping-An[1], LIU Quan[1,2,3,4], HAO Shao-Pu[1], ZHANG Li-Hua[1]

[1](School of Computer Science & Technology, Soochow University, Suzhou 215006, China)

[2](Collaborative Innovation Center of Novel Software Technology and Industrialization (Nanjing), Nanjing 210093, China)

[3](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (Jilin University), Changchun 130012, China)

[4](Jiangsu Provincial Key Laboratory for Computer Information Processing Technology (Soochow University), Suzhou 215006, China)

Abstract: In recent years, deep reinforcement learning has achieved impressive results in complex control tasks. However, its applicability to real-world problems has been seriously weakened by the high sensitivity of hyperparameters and the difficulty in guaranteeing convergence. Metaheuristic algorithms, as a class of black-box optimization methods simulating the objective laws of nature, can effectively avoid the sensitivity of hyperparameters. Nevertheless, they are still faced with various problems, such as the inability to adapt to a huge scale of parameters to be optimized and the low efficiency of sample usage. To address the above problems, this study proposes the twin delayed deep deterministic policy gradient based on a gravitational search algorithm (GSA-TD3). The method combines the

* 基金项目: 国家自然科学基金 (61772355, 61702055, 61876217, 62176175); 江苏高校优势学科建设工程
收稿时间: 2021-08-01; 修改时间: 2021-11-28, 2022-03-30; 采用时间: 2022-07-14; jos 在线出版时间: 2023-06-16
CNKI 网络首发时间: 2023-06-19

advantages of the two types of algorithms. Specifically, it updates the policy by gradient optimization for higher sample efficiency and a faster learning speed. Moreover, it applies the population update method based on the law of gravity to the policy search process to make it more exploratory and stable. GSA-TD3 is further applied to a series of complex control tasks, and experiments show that it significantly out performs similar deep reinforcement learning methods at the forefront.

Key words: deep reinforcement learning (DRL); meta-heuristic algorithm; gravitational search; deterministic policy gradient; policy search

近年来, 强化学习 (reinforcement learning, RL)[1]取得了令人瞩目的成果, 训练出的智能体能够在围棋游戏中战胜人类顶尖选手[2], 在复杂的任务中控制机器人, 在 Atari 2600 游戏中达到超越人类玩家的水平. 促进强化学习爆炸式增长的关键在于其运用了基于深度神经网络的函数逼近器, 将强化学习成功地扩展到具有高维状态和动作空间的任务, 使得在巨大的搜索空间中搜索到足够好的策略成为可能, 这种组合方式被称之为深度强化学习 (deep reinforcement learning, DRL)[3,4].

DRL 已成为人工智能和机器学习领域的一个热点, Mnih 等人[5]结合 Q 学习算法和深度学习, 构建多层深度神经网络逼近状态-动作值函数, 引入经验回放机制, 提出了深度 Q 网络 (deep Q-network, DQN) 方法, DQN 将 Atari 2600 视频游戏画面的像素作为输入, 并达到了超越人类玩家的水平. 为了提升训练效率, Mnih 等人[6]采用多核 CPU 替代 GPU, 以异步并行执行多个智能体的方式降低硬件要求和提高稳定性. 在高维连续状态和动作空间中, Lillicrap 等人[7]将确定策略梯度方法 (deterministic policy gradient, DPG) 和 DQN 融合, 提出了深度确定策略梯度方法 (deep deterministic policy gradient, DDPG). Scott 等人[8]提出了双延迟深度确定策略梯度方法 (twin delayed deep deterministic policy gradient, TD3), 在 DDPG 的基础上采用双 Q 值学习并且延迟策略更新的方式限制值函数被过高估计. Schulman 等人[9]提出了置信域策略优化方法 (trust region policy optimization, TRPO), 证明最小化目标损失函数并且选择合适的步长可以保证策略被单调优化, 并引入广义优势估计 (generalized advantage estimation, GAE)[10], 有效地降低方差. Schulman 等人[11]提出了近端策略优化方法 (proximal policy optimization, PPO), 通过剪枝操作对新旧策略概率比进行限制, 避免出现更新步长过大的现象. Tuomas 等人[12]提出了软性行动者-评论家算法 (soft actor-critic, SAC), 将最大熵引入行动者-评论家框架, 使其能够有效地进行样本学习, 并且增强了探索性.

强有力的探索是强化学习成功完成具有挑战性任务的关键之一, 其能够使智能体学习到更好的策略, 避免过早地收敛到局部最优. 设计具有高探索性的策略搜索方法仍然是 DRL 在高维状态和动作空间上的挑战[13], 目前的工作通过变分信息最大化[14]、好奇心驱动[15]、添加噪音[16]等方式增强探索性. 然而, 上述方法要么依赖于改变结构, 要么需要引入针对特定任务严格调优后的超参数. 其次, DRL 通常对其超参数十分敏感[17], 结果往往难以复现, 并且表现出脆弱的收敛性[18]. 所以, 有效探索与降低超参数敏感在强化学习算法中仍然是一个活跃的研究领域.

作为强化学习的替代方案, Salimans 等人[19]采用黑盒优化的进化算法 (evolution strategy, ES) 可靠地训练出策略网络. 进化算法是一类典型的元启发式算法 (meta-heuristic method), 元启发式算法是一类为克服难以使用数值方法解决实际问题而开发的优化方法[20,21], 这类方法基于种群, 通过随机算子增强探索性, 从而降低了陷入局部最优点的可能性, 且由于其实现容易、数学运算简单等优点而被广泛应用于工程问题[22]. 元启发式算法通常被分为 4 类: 受自然界启发的进化算法, 通过随机产生并逐渐演化的迭代过程达到最佳结果, 典型的方法是遗传算法 (genetic algorithm, GA)[23]和协方差自适应进化策略 (covariance matrix adaptation evolution strategy, CMA-ES)[24]; 基于物理规律的优化算法, 使用物理规则来更新策略, 如引力搜索算法 (gravitational search algorithm, GSA)[25]和黑洞算法 (black hole algorithm, BH)[26]等; 模仿动物社会行为来更新解决方案, 如著名的粒子群算法 (particle swarm optimization, PSO)[27]和灰狼优化算法 (grey wolf optimizer, GWO)[28]等; 模拟人类行为获得最佳结果, 如烟花算法 (firework algorithm)[29]等. 然而, 元启发式算法通常需要完整的轨迹才可以学习, 具有很高的采样复杂度, 并且往往难以解决待优化策略参数规模巨大的高维问题, 最直观的原因是元启发式算法没有使用梯度方法, 梯度方法的优势在于可以对每一个维度的参数进行优化, 而元启发式算法的更新过程是对所有维度参数进行的模糊优化, 并且随机算子在增加探索性的同时也使得在高维问题上很难搜索到参数的最优解.

根据上述两大类方法的优缺点, 本文给出一种融合 DRL 和元启发式算法的策略搜索框架, 并将 GSA 和 TD3 代入到框架中, 提出了融合引力搜索的双延迟深度确定策略梯度方法 (twin delayed deep deterministic policy gradient based on gravitational search algorithm, GSA-TD3), 该方法结合梯度方法和元启发式方法对策略进行优化, 通过元启发式方法中的随机算子增强探索性, 降低陷入局部最优的可能, 同时依赖梯度方法优化高维状态和动作空间的策略参数.

本文的贡献主要包括 3 个方面: (1) 提出融合 DRL 和元启发式算法的策略搜索框架, 通过结合两类方法的优点, 有效地提升策略搜索的效率; (2) 设计评价器算法, 以记录智能体完成多次情节平均累积奖赏的形式, 将强化学习问题转化为元启发式算法的优化目标, 从而将两大类方法衔接在一起; (3) 在 4 个基准实验中, 将 GSA-TD3 算法与经典的 DRL 方法和同类型的前沿算法等进行对比, 验证了所提算法的优越性.

1 相关工作

将基于梯度目标优化方法与进化方法相结合是 DRL 与元启发式算法相结合的最新研究热点[30], 这些方法采用以种群为基础的策略从强化学习过程中继承主要技能[31], 然后采用进化方法进行优化. Khadka 等人[32]提出进化强化学习 (evolutionary reinforcement learning, ERL), ERL 利用选择性突变和基因交叉, 使种群中较弱的策略在探索过程中从强化学习过程训练的策略继承技能, 这种结合方式是样本效率和可扩展性的平衡. 在 ERL 基础上, Khadka 等人[33]提出协同进化强化学习 (collaborative evolutionary reinforcement learning, CERL), 在 CERL 中, 不同的策略在不同范围内探索对应任务的解空间, 并使用资源管理器对策略进行训练资源分配. Bodnar 等人[34]提出一种近端蒸馏进化强化学习 (proximal distilled evolutionary reinforcement learning, PDERL), 将进化方法与强化学习进行分层整合. 通过结合深度神经进化和 DRL. Pourchot 等人[35]提出了结合交叉熵方法的异策略深度强化学习. 然而进化方法因突变过程中具有很强的随机性, 在应对不同的任务时, 无法保证其有效性, Suri 等人[36]提出了基于进化的软性行动者-评论家框架 (evolution-based soft actor-critic, ESAC), 利用软性行动者选择和遗传交叉实现了后代之间的显性转移, 同时自动调整突变进化中的超参数. Hallawa 等人[37]提出了进化驱动强化学习 (evolutionary-driven reinforcement learning, Evo-RL), 将强化学习算法嵌入到一个进化周期中, 使其能够适应无回报状态环境的学习, 更适合于信息不完整的现实世界问题. Chen 等人[38]提出了一种有效的多策略软行动者-评论家算法来协同学习多个策略. 本文提出的方法与上述方法不同之处在于, 以种群为基础的策略不再从强化学习过程中继承技能, 而是将两类方法拼接在一起工作, 改变原先独立工作后继承技能的机制, 在一次迭代过程中, 种群中的每一个行动者在元启发式更新后, 依次采取梯度优化.

2 背景知识

2.1 强化学习

强化学习模型通常被形式化为马尔可夫决策过程 (Markov decision process, MDP), 它描述了智能体在有限的离散时间步内与环境交互的完整信息. 在每个时间步 t, 智能体观察到状态 s_t, 根据策略 π, 获得动作 a_t. 智能体执行动作 a_t 后, 会从环境获得奖赏 r_t, 并且转移到下一个状态 s_{t+1}, 重复该过程, 直至到达终止状态为止. 定义累积奖赏为从时间步 t 开始, 伴随折扣因子 $\gamma \in (0, 1]$ 直至终止状态的奖赏之和, 即 $\sum_{k=1}^{\infty} \gamma^k r_{t+k}$. 状态-动作价值函数 $Q^{\pi}(s,a)$ 描述了从状态 s 开始, 在策略 π 下采取行动 a 的预期收益. 强化学习算法的目标是最大化累积奖赏的期望, 获得最优策略 π_*.

2.2 引力搜索算法

GSA 是基于牛顿万有引力定律的群智能优化算法, 其以种群中的粒子为基本对象, 将粒子所处的位置对应于搜索空间中的一组解, 使粒子遵循万有引力定律在搜索空间中运动, 当粒子移动到最优位置时, 即达到搜索到最优解的目标.

考虑在一个 n 维搜索空间中, 随机产生 N 个粒子, 定义第 i 个粒子的位置为:

$$X_i = \left(x_i^1, \ldots, x_i^d, \ldots, x_i^n\right) \quad \text{for } i = 1, 2, \ldots, N \tag{1}$$

其中, x_i^d 表示第 i 个粒子在 d 维空间上所处的位置.

粒子的适应度值代表粒子当前所处位置在搜索空间中的优劣程度, 适应度值越大, 证明该粒子越靠近所求函数的最优值. 粒子的质量是根据适应度值大小进行计算的, 两者成正比, 定义在时刻 t, 粒子的质量为:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{2}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{3}$$

其中, $fit_i(t)$ 表示粒子 i 在时刻 t 的适应度值, 定义在时刻 t, 最优和最差的适应度值为:

$$best(t) = \max_{j \in \{1,\ldots,N\}} fit_j(t) \tag{4}$$

$$worst(t) = \min_{j \in \{1,\ldots,N\}} fit_j(t) \tag{5}$$

在万有引力定律中, 粒子之间因为引力相互吸引, 两个粒子之间的引力与它们的质量成正比, 与它们之间的距离 R 成反比. 在时刻 t, 定义粒子 i 在 d 维上受到粒子 j 的引力为:

$$F_{ij}^d(t) = G(t) \frac{M_i(t) \times M_j(t)}{R_{ij}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t)\right) \tag{6}$$

其中, $G(t)$ 是第 t 次迭代时的引力常数, 具体更新方式如公式 (7) 所示, 通常设置常数 G_0 为 100, α 为 20, $t_{\max}$ 为迭代的最大次数. $R_{ij}(t)$ 是在 t 时刻粒子 i 和粒子 j 之间的欧式距离, 其具体计算方式如公式 (8) 所示. ε 是一个极小的常数.

$$G(t) = G_0 \mathrm{e}^{-\alpha \frac{t}{t_{\max}}} \tag{7}$$

$$R_{ij}(t) = \|X_i(t), X_j(t)\| \tag{8}$$

粒子 i 在第 d 维所受的总引力是其他粒子施加给粒子 i 引力的第 d 维分量的随机加权和, 随机算子 $rand_j$ 服从区间 [0, 1] 上的均匀分布:

$$F_i^d(t) = \sum_{j=1, j \neq i}^{N} rand_j F_{ij}^d(t) \tag{9}$$

任何粒子的当前速度等于它之前的速度和速度变化量的总和, 其加速度等于作用在粒子上的力除以其自身质量. 对于粒子 i, 给出在 d 维上的加速度更新公式:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{10}$$

加速度是速度的增量, 因此, 对于每一次迭代过程, 第 d 维加速度与速度的和便是粒子的新速度, 根据运动规律, 粒子以新速度移动后到达新的位置:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{11}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{12}$$

其中, 随机算子 $rand_i$ 属于区间 [0, 1].

2.3 双延迟深度确定策略梯度

值函数被高估是强化学习算法普遍存在的一个问题, 以离散动作环境的 Q 值学习 (Q-learning) 算法为例, 其值函数的估计值更新依赖于贪心的目标 $y = r + \gamma \max_{a'} Q(s', a')$, 而 Q 值的估计存在误差 ω, 伴随误差的 Q 值在执行完最大化操作后会明显大于真实的最大 Q 值, 即 $\mathbb{E}_{\omega}[\max_{a'}(Q(s', a') + \omega)] \geqslant \max_{a'} Q(s', a')$. 估计误差是使用值函数逼近引起的, 所以值函数在更新的时候会出现高估偏差, 并且这种偏差会随着贝尔曼方程传递下去.

策略梯度方法将期望收益最大化的目标重新定义为最大化性能指标 $J(\phi)$, 其中 ϕ 表示策略的所有参数. 一种比较常用的确定策略梯度方法是 DDPG, 其是一种能够处理高维连续状态动作空间任务的无模型算法. 然而

在确定策略梯度方法中, 以最大化 Q 值函数的方式优化策略, 依旧会为值函数的估计引入偏差, 从而导致高估问题出现. DDPG 会因价值函数在训练过程中出现高估问题而影响性能, 如果在训练过程中价值函数不断被高估, 策略更新将受到严重的负面影响. TD3 在 DDPG 基础上提出 3 种技术进行改进, 能够有效防止价值函数被高估等问题.

第 1 种技术是裁剪的双 Q 值学习, 使用两个独立的 Q 值函数替代原先的一个 Q 值函数. 两个价值网络通过分离动作的选择, 获得两个不一致的 Q 值估计值, 选取较小的估计值作为更新目标, 从而缓解 Q 值函数在更新的过程中被过高估计:

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', a') \tag{13}$$

第 2 种技术是延迟更新, 保持策略和目标值函数更新的频率低于价值函数更新的频率. 如果没有固定的目标值函数, 就会导致每次更新引入新的残余误差, 并开始累积此误差. 这就要求改进措施在价值函数的更新过程中要尽可能保持固定的目标值函数. 最直接有效的方式是使价值函数的更新频率是策略和目标值函数更新频率的整数倍, 即策略和目标值函数的更新次数明显少于价值函数的更新次数, 从而保证了价值函数在多次更新中使用的是固定的目标值函数. 不太频繁的策略和目标值函数更新可以获得低方差的价值估计, 这使得价值网络在被用于更新策略网络之前变得更加稳定, 以达到减少累积误差的目的.

第 3 种技术是目标策略平滑, 从类似的 Q 值估计来进行引导更新. 当智能体利用同策略进行探索时, 很容易因为没有尝试足够广泛的行动, 使得学习过程有效性偏低. 虽然 TD3 是异策略算法, 但确定策略也存在难以保证充分探索的问题, 有效的方法是在环境中采样时增加有效的噪声. TD3 通过向目标策略中添加少量随机噪声来近似类似状态动作值的期望:

$$\begin{cases} y = r + Q_{\theta'}\left(s', \pi_{\phi'}(s') + \mu\right) \\ \mu \sim clip(N(0,\sigma), -c, c) \end{cases} \tag{14}$$

其中, μ 是随机噪声, 服从高斯分布; $clip(N(0,\sigma), -c, c)$ 是截断函数, 将服从高斯分布的随机噪声 μ 截断在区间 $[-c, c]$ 内.

TD3 是在 DDPG 基础上的改进, 其 Q 值更新计算方法类似于 DDPG. 行为策略在探索环境过程中引入高斯噪声, 执行完每一个动作之后, 包含当前状态、动作、下一个状态和奖赏的四元组 (s_t, a_t, r_t, s_{t+1}) 会被记录到经验缓冲池 $\mathcal{D}$ 中. 评论家通过最小化损失函数更新, B 表示从经验缓冲池中取出的样本数量:

$$L = B^{-1} \sum_i \left(y - Q_{\theta_i}(s, a)\right)^2 \tag{15}$$

为了达到延迟更新减少累积误差的目的, 在特定的 Q 值更新迭代次数之后, 以最大化第 1 个评论家的方式更新行动者网络.

$$\nabla_\phi J(\phi) = B^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s) \tag{16}$$

同时更新目标评论家网络和目标行动者网络:

$$\begin{cases} \theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i' \\ \phi' \leftarrow \tau\phi + (1-\tau)\phi' \end{cases} \tag{17}$$

3 GSA-TD3

如图 1 所示, 本文提出的算法 GSA-TD3 融合了 GSA 和 TD3, 以种群更新和梯度更新相结合的方式搜索最佳策略, 其主要的创新点在于: (1) 将强化学习过程中的策略作为种群中最基本的粒子, 策略的所有参数与粒子的位置相对应; (2) 评价器的任务是使当前策略与环境进行交互, 并将多条情节的累积奖赏均值作为当前策略的适应度值. 在每一次迭代过程中, 种群中的每一个策略都会被评价器评估, 并返回其适应度值, 用于计算质量、加速度和速度并更新策略参数, 然后评论家网络根据缓冲池中取出的经验进行梯度更新, 同时种群中的策略也进行梯度更新.

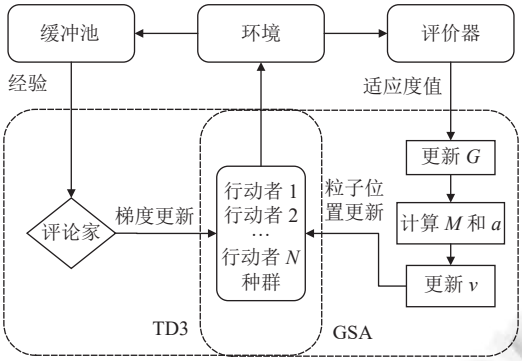


图 1 GSA-TD3 结构

融入元启发式算法的直接目的是提高强化学习算法的探索性, 避免陷入局部最优, 从而提升算法的性能. 然而, 超强的探索性只是增加了获得优秀动作经验的可能性, 并不能保证所有收集的经验数据都具有学习价值. 在实际应用中, 过高的探索性不仅没有提升算法的性能, 反而对其收敛进程产生影响, 所以优秀的强化学习算法需要兼顾探索和利用的平衡. 与 DRL 框架的结合使得所选择的元启发式算法不仅需要较强的探索性, 还需要种群中的粒子存在紧密联系, 即单个粒子的更新受到种群中越多其他粒子的影响越好, 从而在算法训练的后期平衡粒子因较强的探索而引入的不确定性, 以达到提升算法收敛性能的目的. 相较于经典的元启发式算法例如 GA 和 PSO, 选择 GSA 的原因在于, 虽然 GA 和 PSO 相比于 GSA 虽然具备更强的探索性, 但是在 GA 中, 新一代粒子的更新过程仅使用到了上一代两个粒子的交叉变异信息, 同样地在 PSO 中, 也仅使用了上一代两个最优粒子去计算当前粒子的更新方向. 上述两种算法的更新方式都是采用了部分粒子信息, 而忽略了全局的影响, 在算法训练的后期, 易产生较大的波动, 难以达到收敛状态. 在 GSA 中, 粒子的更新依赖于种群中除自身外所有粒子施加的引力信息, 相比于 GA 和 PSO, GSA 中粒子的更新受到全局信息的影响更大, 更易趋于平衡状态.

GSA-TD3 的主要优势在于整合了多个策略的优势, 较优的策略会给予其他策略正向反馈. 在 TD3 中, 单个策略仅进行梯度更新, 而在 GSA-TD3 算法结构中, 多个策略不仅同时进行梯度更新, 还会根据适应度值在万有引力定律下进行启发式更新. 不同的策略, 在进行梯度更新后, 会表现出对任务不同程度的解决能力, 所有的策略在万有引力的作用下相互影响, 从而加速了策略收敛的进程. 同时, GSA-TD3 继承了 GSA 中引入的随机算子, 使得策略在元启发式更新的过程中具有更强的探索性, 提高了搜索到最优策略的可能性.

在 GSA-TD3 算法结构中, 评价器是评判策略解决特定任务优劣的唯一标准, 其将策略与环境多次交互后的累积奖赏均值作为评价指标对策略做出评价, 伪代码如算法 1 所示. 在伪代码中, 评价器算法不仅需要与环境交互计算出当前策略的适应度值, 还需要将与环境交互的经验数据保存在经验缓冲池中并记录与环境交互的总时间步.

算法 1. 评价器算法.

输入: 策略 π, 经验缓冲池 $\mathcal{D}$, 情节数 ξ;
输出: 策略 π 的适应度值 *fitness*, 与环境交互的总时间步数 *T*.

evaluate(π, $\mathcal{D}$, ξ)
1. Let *fitness* = 0, *T* = 0.
2. **for** $i = 0{:}\xi$ **do**
3. Reset environment and get initial state s_0.
4. **while** environment is not done **do**
5. Select action $a_t = \pi(s_t) + noise$.

6. Execute action a_t.
7. Observe reward r_t and new state s_{t+1}.
8. $fitness = fitness + r_t$, $T = T + 1$.
9. Fill $\mathcal{D}$ with collected experience.
10. $s_t = s_{t+1}$.
11. **end while**
12. **end for**
13. **return** $fitness/\xi$, T.

评价器算法输入需要进行评价的策略π、经验缓冲池$\mathcal{D}$和评价该策略所需的情节数ξ, 输出对应于输入策略的适应度值和当前策略π与环境交互的总时间步 T. 评价器的主要任务是记录当前策略在与环境交互时所获得的奖赏, 并计算ξ条情节所获得的平均奖赏. 首先重置多条情节的总适应度值和总时间步, 当一个情节结束, 环境都会重置并且初始化一个状态s_0. 如算法 1 第 4–11 行所示, 当环境中的状态没有到达终止状态时, 智能体会根据被评价的策略π获取动作a_t, 执行后会获得奖赏r_t和下一个状态s_{t+1}, 总的适应度值以加上奖赏r_t的方式更新, 将收集到的经验存入经验缓冲池中, 下一个状态变为当前状态, 重复上述过程直至到达终止状态为止. 当完成ξ条情节后, 返回当前策略的适应度值和总时间步, 如算法 1 第 13 行所示.

算法 2 介绍了 GSA-TD3 的整个过程, 其是对图 1 的工作过程的详细描述. GSA-TD3 继承于 GSA 和 TD3, 为了融合两种算法的优势, 做出两处改进: (1) 将 GSA 原有的引力常数更新公式更改为:

$$G(t) = G_0\beta^t,\ t \leqslant t_{\max} \tag{18}$$

其中, 常数β的取值范围是 (0, 1], $t_{\max}$为元启发式更新部分迭代的最大次数. 在原始的 GSA 算法中, 引力常数虽然随迭代次数的增大而减小, 但是在算法趋于收敛的阶段, 引力常数仍是一个较大的数值, 不利于算法达到收敛状态. 而修改后的引力常数$G(t)$随迭代次数的增长逐渐减小的幅度增加, 降低了在训练后期的算法的探索性, 是为了达到探索与收敛的平衡. (2) 将 TD3 中的延迟更新取消, 取消的目的是使得种群中的策略在梯度更新后以最新的评估适应度值参与到元启发式更新, 加速策略的训练进程.

在 GSA-TD3 中, 种群规模N代表种群中含有的策略个数, 与 TD3 初始化设置唯一不同之处在于, TD3 中仅需要初始化一个行为策略和目标策略, 而此处需要初始化 N 个, 如算法 2 第 1–2 行所示. 因为智能体与环境的交互过程被集成到评价器算法中, 所以无法和 TD3 一样与环境每交互一步进行一次梯度更新, 评价器算法返回的总时间步数是当前策略与环境交互的总步数, 同时也是评论家网络和策略网络所需要梯度更新的次数, 算法中设置 *train_steps* 的目的即在于记录该更新次数, 如算法 2 第 6–9 行所示. 在得到每个策略的适应度值后, 依据万有引力定律对策略进行更新, 如算法 2 第 12–14 行所示. 最后, 从经验缓冲池中取出 B 条经验, 完成对评论家网络和策略网络的梯度更新, 并更新目标评论家网络和目标策略网络, 如算法 2 第 15–22 行所示.

算法 2. GSA-TD3.

输入: 最大环境交互步数 *max_steps*, 种群规模 N, 引力常数G_0, β, $t_{\max}$, 折扣率 γ, 评论家网络学习率lr_{critic}, 策略网络学习率lr_{actor}, 训练所用批量数 B, τ.

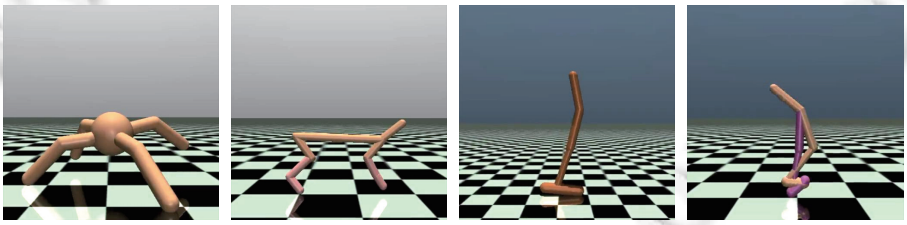
1. Initialize critic networks and actor networks with random parameters.
2. Initialize target critic networks and target actor networks with the same parameters as above.
3. Initialize replay buffer $\mathcal{D}$.
4. Let $total\_steps = 0$, $t = 1$.
5. **while** $total\_steps < max\_steps$ and $t \leqslant t_{\max}$ **do**
6. Let $train\_steps = 0$.
7. **for** $i = 1: N$ **do**

fitness[*i*], *actor_steps* = *evaluate*(actor[*i*], $\mathcal{D}$, ξ).
train_steps = *train_steps* + *actor_steps*.
end for
total_steps = *total_steps* + *train_steps*, *t* = *t* + 1.
Update gravitational constant *G*, *best* and *worst* of the population.
Calculate *M* and *a* for each policy in population.
Upadte velocity and position.
for *step* = 1: *train_steps* **do**
Sample mini-batch of *B* transitions (s,a,r,s') from $\mathcal{D}$.
Train critic networks.
for *i* = 1: *N* **do**
Train *i*th actor network in population.
end for
Update parameters of target critic networks and target actor networks.
end for
end while

4 实 验

4.1 实验环境介绍

OpenAI Gym[39]是面向强化学习开发和算法对比的开源工具包, 其中包含众多的基准测试, 如 Atali 2600 游戏, 并且向第三方任务开放了通用接口, 为人工智能开发者和研究者提供了便利的环境和具有挑战性的任务. 如图 2 所示, 本文采用 OpenAI 基于 Mujoco 物理引擎在 Gym 中开发的一系列连续控制任务作为实验环境.



(a) Ant-v2 (b) HalfCheetah-v2 (c) Hopper-v2 (d) Walker2d-v2

图 2 Mujoco 物理引擎环境

为了有效评估算法的性能, 选取基于 Mujoco 物理引擎模拟的连续控制任务 Ant-v2、HalfCheetah-v2、Hopper-v2 和 Walker2d-v2 进行实验, 其简要介绍列于表 1. 本文所有实验均采用配置为 Intel Xeon E5-2680 v4 CPU、2 块 NVIDIA Tesla P40 GPU 和 128 GB 内存的服务器作为硬件环境.

表 1 实验任务简要介绍

任务	状态维度	动作维度	任务目标
Ant-v2	111	8	训练一个四足的智能体学会行走
HalfCheetah-v2	17	6	训练一个两足的智能体学会行走
Hopper-v2	11	3	训练一个单足的智能体向前跳跃
Walker2d-v2	17	6	训练二维双足智能体尽可能快地向前走

4.2 实验设置

对于每一个任务, 所有参与对比的算法均会以 5 个不同的随机种子独立运行, 随机种子的取值是 2020 至 2024 之间的整数. 当一个算法独立运行时, 其训练的策略网络会在相同的任务副本上被周期性的评测, 评测方式为使用当前策略网络在该任务副本上完成多个情节, 记录多个情节的累积奖赏均值作为该策略网络在当前时间步的性能指标. 在之后的算法性能对比图中, 实线代表该算法 5 次独立运行所训练的策略在当前时间步评测后得到的性能均值, 阴影部分为 5 次独立训练的性能值的波动, 阴影部分越大, 表示训练该策略的算法稳定性越差.

本文提出的 GSA-TD3 使用深度学习框架 PyTorch 在 TD3 作者提供的代码基础上实现, 采用有两个隐层 (第 1 层有 400 个神经元, 第 2 层有 300 个神经元) 的线性神经网络作为评论家网络和行动者网络, 评论家网络和行动者网络分别使用 ReLU 函数和 tanh 函数作为激活函数, 并使用优化器 Adam 以梯度下降的方式更新神经网络参数, GSA-TD3 其他的超参数设置如表 2 所示. 在评测 GSA-TD3 算法时, 选取当前迭代次数下, 种群中最优的策略, 即适应度值最大的策略进行评测. 为了更加公平有效地对比不同算法的性能, 本文涉及的其他算法均使用与 GSA-TD3 相一致的行动者和评论家网络结构, 其超参数尽可能地采用与 GSA-TD3 相同的设置.

表 2　GSA-TD3 超参数

超参数	取值	参数描述
N	10	种群规模
G_0	100	引力常数初始值
β	0.99	引力更新常数
$t_{\max}$	500	启发式更新最大迭代次数
γ	0.99	折扣率
lr_{critic}	1E–3	评论家网络学习率
lr_{actor}	2E–3	策略网络学习率
B	256	训练所用批量数
τ	5E–3	软更新常数

4.3 实验结果与分析

首先, 选择经典的深度强化学习算法 TD3、SAC 和 PPO 与本文提出的 GSA-TD3 进行性能对比, 其学习曲线如图 3 所示. GSA-TD3 相较于经典的深度强化学习, 其优势来自算法集成元启发式算法的更新步骤, 使得在策略搜索的过程中具有更强的探索性且不易陷入局部最优的陷阱. 作为 TD3 的进阶版本, 在所有进行对比的任务上, GSA-TD3 相较于 TD3 均有不同程度的提升, 并且相对于 PPO, GSA-TD3 也有非常好的表现. 在任务 HalfCheetah-v2 中, GSA-TD3 略逊于 SAC, 这是因为 HalfCheetah-v2 与其他任务相比, 搜索空间更为复杂, 需要极强的探索性, 才可以获得不错的效果. 与 TD3 采用确定策略不同的是, SAC 采用随机策略, 确定策略根据状态值输出确定的动作, 而随机策略则是根据状态值输出动作的概率分布, 通过采样的方式获得动作, 这就使得采用随机策略获得的动作相比于确定策略, 具有随机性, 即采用随机策略的 SAC 相较于采用确定策略的 TD3, 拥有更强的探索性. 此外, SAC 将策略的熵引入损失函数, 在策略更新过程中最大化性能指标的同时最大化策略熵, 使得根据该策略输出的动作概率分布更加均匀, 从另一个角度提升了探索性, 所以 SAC 更适合需要高探索的任务. GSA-TD3 在 TD3 的基础上引入引力搜索增强探索性, 但是过度的探索性会导致智能体收集到较多无用的经验数据, 经过训练后会降低算法的稳定性, 为了保证算法的稳定性, GSA-TD3 在训练过程中不断减小 G 值, 通过缓慢降低探索性的方式提升稳定性, 达到探索性与稳定性的平衡, 故在任务 HalfCheetah-v2 中, GSA-TD3 因保持稳定性的缘故, 其性能略逊于 SAC. 在任务 Walker2d-v2 中, GSA-TD3 相比于 SAC, 曲线更为平稳, 即拥有更好的稳定性, 且在除 HalfCheetah-v2 的其他任务中, 由于保持探索性与稳定性处于平衡状态, GSA-TD3 的性能均优于 SAC.

表 3 给出了 GSA-TD3 和上述对比算法在 5 次独立训练终止时, 策略在对应任务上所获得的累积奖赏的平均值、标准差和中位数, 并且将每个任务中表现最好 (以平均值作为评判标准) 的算法数据加粗标记. GSA-TD3 在

Ant-v2、Hopper-v2 和 Walker2d-v2 任务上表现最优, 在 HalfCheetah-v2 上, 其最终性能仅次于 SAC. 同时, 相比较于这 3 个算法, GSA-TD3 的标准差相对较低, 体现出较为优异的稳定性.

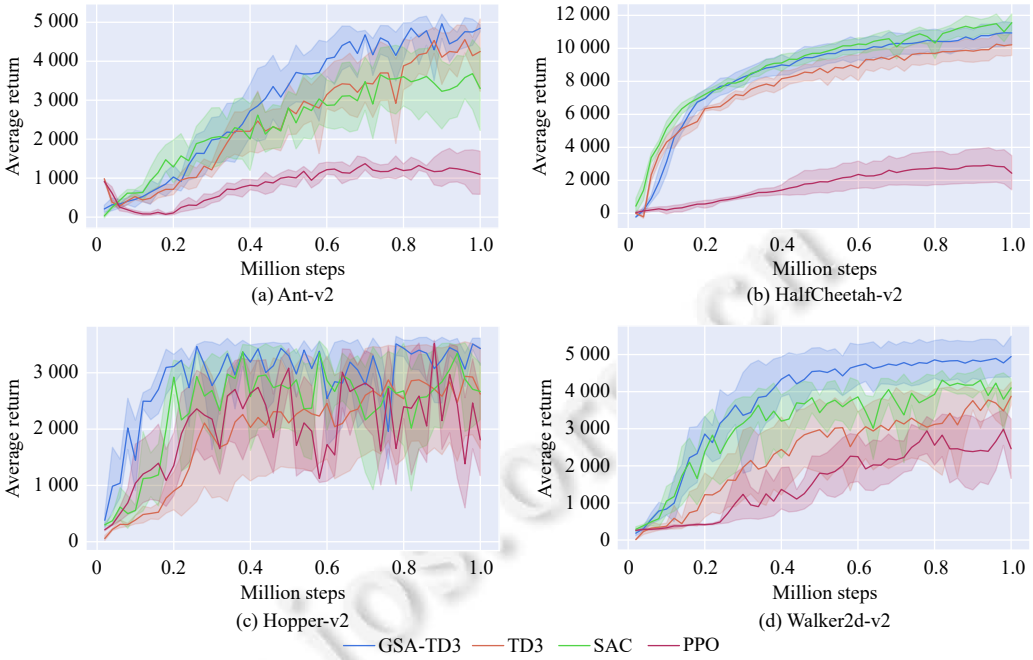


图 3 GSA-TD3 和经典深度强化学习算法学习曲线

表 3 GSA-TD3 和经典深度强化学习算法最终性能

任务	GSA-TD3			TD3			SAC			PPO		
	平均值	标准差	中位数	平均值	标准差	中位数	平均值	标准差	中位数	平均值	标准差	中位数
Ant-v2	**4847**	**102**	**4885**	4250	1011	4661	3301	1171	3732	1362	489	1297
HalfCheetah-v2	10932	810	11409	10215	751	10625	**11553**	**574**	**11542**	2431	1206	1644
Hopper-v2	**3430**	**299**	**3532**	2622	1028	3023	2669	595	2615	1810	816	1234
Walker2d-v2	**4946**	**663**	**4498**	3875	380	3728	4102	276	4141	2462	922	2164

其次, 选择两种相关联的前沿算法 CERL 和 ESAC 与本文提出的 GSA-TD3 进行性能对比. CERL 的全称是协作进化强化学习, 其主要思想是通过有概率的交叉变异过程对策略的参数进行扰动产生新一代的策略, 并保护优势策略, 放大优势策略的作用, 将计算资源倾向于较优的策略. ESAC 的主要思想是从每一代变异的种群中选取部分优胜者, 即适应度值较高的行动者, 将优胜者进行交叉变异更新后和 SAC 训练的行动者加入种群中, 从而达到优化策略的目的.

GSA-TD3 与 CERL 和 ESAC 的性能对比如图 4 所示, 在任务 Ant-v2 中, CERL 在经过 1 百万时间步的训练后, 策略仍处于很低的水平, 这是因为 Ant-v2 环境状态维度高, 导致策略网络的参数量巨大, 而 CERL 的交叉变异过程随机性高且没有自适应地控制参数交叉变异的强度, 对于参数量巨大的策略, 交叉变异过程所做的探索常常是无用的. 与 CERL 不同的是, ESAC 仅选取部分行动者进行交叉变异, 虽然降低了探索性, 但是在一定程度上缓解了交叉变异过程无用探索的影响, 从图 4 中可以看出, ESAC 明显优于 CERL.

CERL 和 ESAC 在交叉变异过程中没有控制参数交叉变异的强度, 从而放大了随机算子在算法中的影响, 这在很多基于进化策略的策略搜索算法中都存在. 因此在本文提出的算法 GSA-TD3 中, 修改了引力常数的计算方式, 引力常数呈逐渐递减的趋势, 降低了随机算子的影响, 使得算法在训练策略的过程中表现出前期迅速后期趋于稳定的特点. 同时 CERL 和 ESAC 也过多关注了具有优势的策略, 赋予过多的计算资源, 使得其他较差的策略难以在迭代过程中完成从劣到优的转化. 在所有评测的任务上, GSA-TD3 相较于 CERL 和 ESAC 性能表现更为优异.

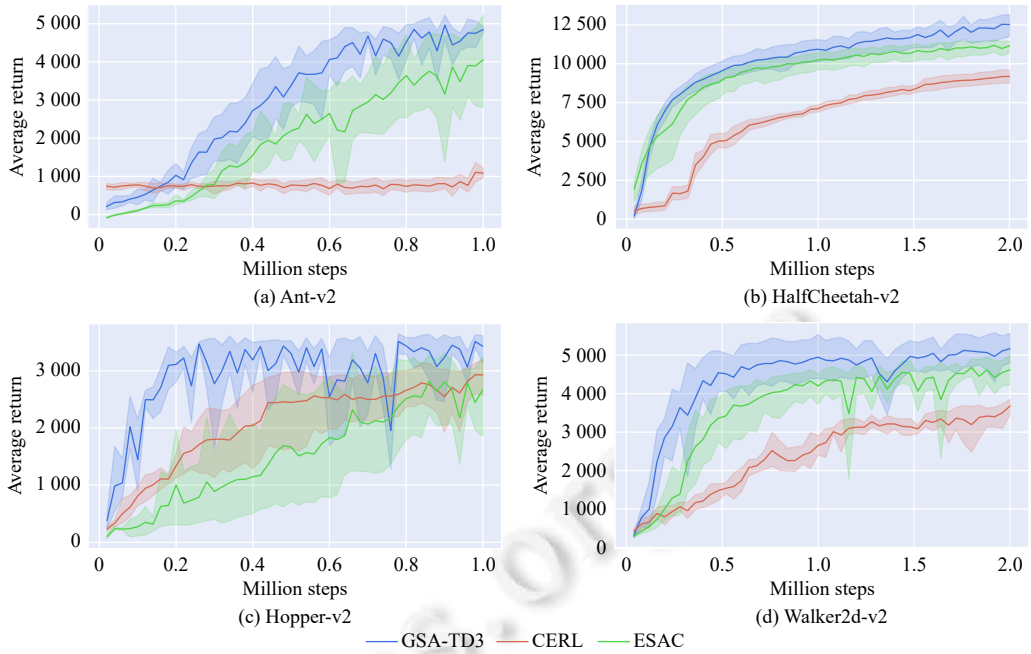


图 4 GSA-TD3、CERL 和 ESAC 学习曲线

表 4 给出了 GSA-TD3、CERL 和 ESAC 在 5 次独立训练终止时, 策略在对应任务上所获得的累积奖赏的平均值、标准差和中位数, 并且将每个任务中表现最好 (以平均值作为评判标准) 的算法数据加粗标记. 从表 4 中可以看出, GSA-TD3 相对于 CERL 和 ESAC 具有巨大的优势.

表 4 GSA-TD3、CERL 和 ESAC 最终性能

任务	GSA-TD3			CERL			ESAC		
	平均值	标准差	中位数	平均值	标准差	中位数	平均值	标准差	中位数
Ant-v2	**4847**	**102**	**4885**	669	425	864	4056	1399	4777
HalfCheetah-v2	**12526**	**859**	**12774**	9182	486	8980	11151	468	11288
Hopper-v2	**3430**	**299**	**3532**	2928	376	3073	2668	804	2891
Walker2d-v2	**5167**	**465**	**4930**	3421	352	3429	4618	439	4841

GSA-TD3 是基于融合 DRL 与元启发式算法作为新框架的一种尝试, 本文选取另一种较为流行的元启发式算法 GWO 在 GSA-TD3 算法框架中替代 GSA 构成基于灰狼优化的双延迟深度确定策略梯度方法 (twin delayed deep deterministic policy gradient based on grey wolf optimizer, GWO-TD3), 并与 GSA-TD3 进行性能对比, 其结果如图 5 所示. 虽然 GWO-TD3 能够有效地训练策略, 但是与 GSA-TD3 算法仍有较大的差距.

GWO 是模拟狼群捕猎的群智能优化算法, 在元启发式搜索的每一次迭代过程中, 选取种群中最优的 3 个粒子, 根据它们所处的位置和需要更新的粒子所处的位置进行更新, 其优点是计算量小, 使得搜索进程加快, 缺点是仅考虑 3 个最优的粒子而忽略大部分粒子的作用, 从而使得在搜索的过程中有可能陷入局部最优的陷阱. 推广到 GWO-TD3, 在 Ant-v2 和 Walker2d-v2 的环境中, 其在开始阶段策略训练速度相比于 GSA-TD3 快, 但是在所有的任务上, GWO-TD3 因陷入局部最优, 在多轮训练后, 其性能均落后于 GSA-TD3.

表 5 给出了 GSA-TD3 和 GWO-TD3 在 5 次独立训练终止时, 策略在对应任务上所获得的累积奖赏的平均值、标准差和中位数, 并且将每个任务中表现最好 (以平均值作为评判标准) 的算法数据加粗标记. 从表 5 中可以看出, GSA-TD3 相对于 GWO-TD3 具有明显的优势.

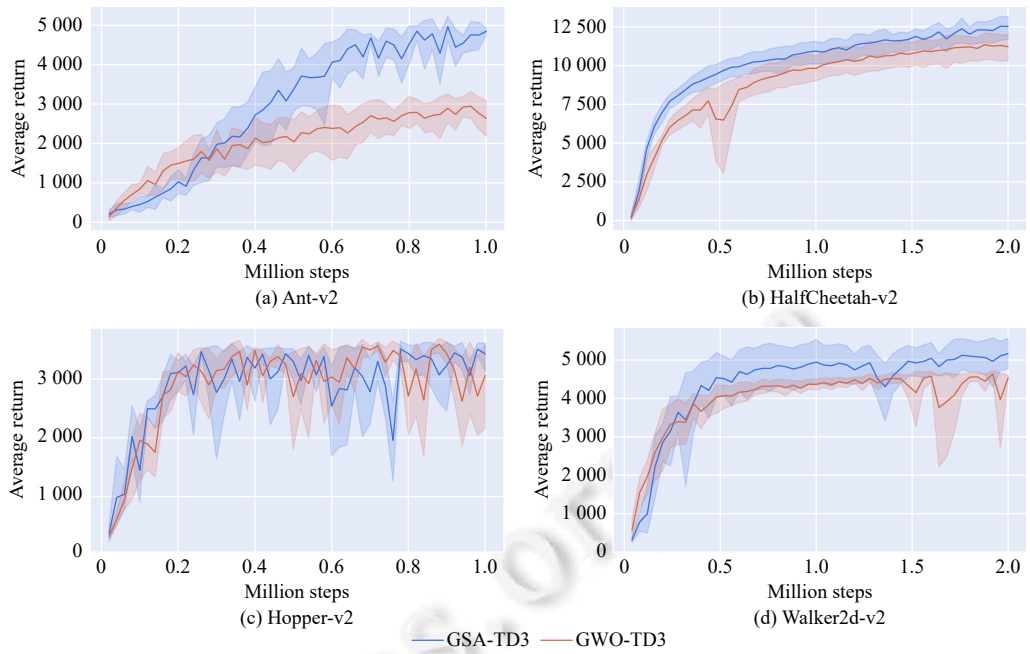


图 5 GSA-TD3 和 GWO-TD3 学习曲线

表 5 GSA-TD3 和 GWO-TD3 最终性能

任务	GSA-TD3			GWO-TD3		
	平均值	标准差	中位数	平均值	标准差	中位数
Ant-v2	**4847**	**102**	**4885**	2638	505	2685
HalfCheetah-v2	**12526**	**859**	**12774**	11210	992	11391
Hopper-v2	**3430**	**299**	**3532**	3062	913	3484
Walker2d-v2	**5167**	**465**	**4930**	4536	80	4575

5 总 结

本文提出了一种结合 DRL 和元启发式算法的策略搜索算法 GSA-TD3. 该算法以融合梯度目标优化方法和元启发式方法的方式对策略进行优化, 将强化学习过程中的策略作为种群中的粒子, 并且把当前策略与环境交互的多条情节的累积奖赏均值作为当前策略的适应度值. 在每一次迭代周期中, 智能体完成与环境交互后对策略依次执行元启发式更新和梯度更新. 该算法在实现更快的学习速度和高样本效率的同时, 能够获得更好的探索性和稳定性. 本文选取 4 个经典的连续控制任务验证算法的有效性, 实验结果表明, 本文提出的算法具有优异的效果.

本文采取确定策略的原因是当固定初始状态时, 每一个情节的累积回报是确定的, 能够准确地反映出当前策略的适应度值. 相比于确定策略, 随机策略有更强的探索性, 并且可以将探索和改进集成到同一个策略中. 下一步研究重点是将元启发式算法融入基于随机策略的深度强化学习中, 重点在于改进评价器算法, 使其能够准确地计算出随机策略的适应度值.

References:

[1] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. 2nd ed., Cambridge: MIT Press, 2018.

[2] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen YT, Lillicrap T, Hui F, Sifre L, van den driessche G, Graepel T, Hassabis D. Mastering the game of go without human knowledge. Nature, 2017, 550(7676):

354–359. [doi: 10.1038/nature24270]

[3] Liu Q, Zhai JW, Zhang ZZ, Zhong S, Zhou Q, Zhang P, Xu J. A survey on deep reinforcement learning. Chinese Journal of Computers, 2018, 41(1): 1–27 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2018.00001]

[4] Liu JW, Gao F, Luo XL. Survey of deep reinforcement learning based on value function and policy gradient. Chinese Journal of Computers, 2019, 42(6): 1406–1438 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2019.01406]

[5] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533. [doi: 10.1038/nature14236]

[6] Mnih V, Badia AP, Mirza M, Graves A, Harley T, Lillicrap TP, Silver D, Kavukcuoglu K. Asynchronous methods for deep reinforcement learning. In: Proc. of the 33rd Int'l Conf. on Machine Learning. New York: JMLR, 2016. 1928–1937.

[7] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. In: Proc. of the 4th Int'l Conf. on Learning Representations. San Juan: CoRR, 2016. [doi: 10.48550/arXiv.1509.02971]

[8] Fujimoto S, van Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: PMLR, 2018. 1582–1591.

[9] Schulman J, Levine S, Moritz P, Jordan M, Abbeel P. Trust region policy optimization. In: Proc. of the 32nd Int'l Conf. on Machine Learning. Lille: JMLR, 2015. 1889–1897.

[10] Schulman J, Moritz P, Levine S, Jordan MI, Abbeel P. High-dimensional continuous control using generalized advantage estimation. In: Proc. of the 4th Int'l Conf. on Learning Representations. San Juan, 2016. [doi: 10.48550/arXiv.1506.02438]

[11] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv:1707.06347, 2017.

[12] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: PMLR, 2018. 1856–1865.

[13] Plappert M, Houthooft R, Dhariwal P, Sidor S, Chen RY, Chen X, Asfour T, Abbeel P, Andrychowicz M. Parameter space noise for exploration. In: Proc. of the 6th Int'l Conf. on Learning Representations. Vancouver: OpenReview.net, 2018.

[14] Houthooft R, Chen X, Duan Y, Schulman J, De Turck F, Abbeel P. VIME: Variational information maximizing exploration. In: Proc. of the 30th Int'l Conf. on Neural Information Processing Systems. Barcelona: Curran Associates Inc., 2016. 1117–1125.

[15] Pathak D, Agrawal P, Efros AA, Darrell T. Curiosity-driven exploration by self-supervised prediction. In: Proc. of the 34th Int'l Conf. on Machine Learning. Sydney: PMLR, 2017. 2778–2787.

[16] Fortunato M, Azar MG, Piot B, Menick J, Hessel M, Osband I, Graves A, Mnih V, Munos R, Hassabis D, Pietquin O, Blundell C, Legg S. Noisy networks for exploration. In: Proc. of the 6th Int'l Conf. on Learning Representations. Vancouver: OpenReview.net, 2018. 1–21.

[17] Henderson P, Islam R, Bachman P, Pineau J, Precup D, Meger D. Deep reinforcement learning that matters. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence, the 30th Innovative Applications of Artificial Intelligence, and the 8th AAAI Symp. on Educational Advances in Artificial Intelligence. New Orleans: AAAI Press, 2018. 3207–3214.

[18] Haarnoja T, Tang HR, Abbeel P, Levine S. Reinforcement learning with deep energy-based policies. In: Proc. of the 34th Int'l Conf. on Machine Learning. Sydney: JMLR, 2017. 1352–1361.

[19] Salimans T, Ho J, Chen X, Sidor S, Sutskever I. Evolution strategies as a scalable alternative to reinforcement learning. arXiv: 1703.03864, 2017.

[20] Dosoglu MK, Guvenc U, Duman S, Sonmez Y, Kahraman HT. Symbiotic organisms search optimization algorithm for economic/ emission dispatch problem in power systems. Neural Computing and Applications, 2018, 29(3): 721–737. [doi: 10.1007/s00521-016-2481-7]

[21] Lara CL, Trespalacios F, Grossmann IE. Global optimization algorithm for capacitated multi-facility continuous location-allocation problems. Journal of Global Optimization, 2018, 71(4): 871–889. [doi: 10.1007/s10898-018-0621-6]

[22] Gao WF, Luo YT, Yuan YF. Overview of intelligent optimization algorithms for solving nonlinear equation systems. Control and Decision, 2021, 36(4): 769–778 (in Chinese with English abstract). [doi: 10.13195/j.kzyjc.2020.0379]

[23] Mitchell M. An Introduction to Genetic Algorithms. Cambridge: MIT Press, 1998.

[24] Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 2001, 9(2): 159–195. [doi: 10.1162/106365601750190398]

[25] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: A gravitational search algorithm. Information Sciences, 2009, 179(13): 2232–2248. [doi: 10.1016/j.ins.2009.03.004]

[26] Hatamlou A. Black hole: A new heuristic optimization approach for data clustering. Information Sciences, 2013, 222: 175–184. [doi: 10.1016/j.ins.2012.08.023]

[27] Kennedy J, Eberhart R. Particle swarm optimization. In: Proc. of the 1995 Int'l Conf. on Neural Networks (ICNN 1995). Perth: IEEE, 1995. 1942–1948. [doi: 10.1109/ICNN.1995.488968]

[28] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Advances in Engineering Software, 2014, 69: 46–61. [doi: 10.1016/j.advengsoft.2013.12.007]

[29] Tan Y, Zhu YC. Fireworks algorithm for optimization. In: Proc. of the 1st Int'l Conf. on Swarm Intelligence. Beijing: Springer, 2010. 355–364. [doi: 10.1007/978-3-642-13495-1_44]

[30] Miconi T, Rawal A, Clune J, Stanley KO. Backpropamine: Training self-modifying neural networks with differentiable neuromodulated plasticity. In: Proc. of the 7th Int'l Conf. on Learning Representations. New Orleans: OpenReview.net, 2020.

[31] Rockefeller G, Khadka S, Tumer K. Multi-level fitness critics for cooperative coevolution. In: Proc. of the 19th Int'l Conf. on Autonomous Agents and Multiagent Systems. Auckland: Int'l Foundation for Autonomous Agents and Multiagent Systems, 2020. 1143–1151.

[32] Khadka S, Tumer K. Evolutionary reinforcement learning. arXiv:1805.07917, 2018.

[33] Khadka S, Majumdar S, Nassar T, Dwiel Z, Tumer E, Miret S, Liu YY, Tumer K. Collaborative evolutionary reinforcement learning. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 3341–3350.

[34] Bodnar C, Day B, Lió P. Proximal distilled evolutionary reinforcement learning. In: Proc. of the 34th AAAI Conf. on Artificial Intelligence, the 32nd Innovative Applications of Artificial Intelligence Conf., the 10th AAAI Symp. on Educational Advances in Artificial Intelligence. New York: AAAI Press, 2020. 3283–3290.

[35] Pourchot A, Sigaud O. CEM-RL: Combining evolutionary and gradient-based methods for policy search. In: Proc. of the 7th Int'l Conf. on Learning Representations. New Orleans: OpenReview.net, 2019. 1–18.

[36] Suri K, Shi XQ, Plataniotis KN, Lawryshyn YA. Maximum mutation reinforcement learning for scalable control. arXiv:2007.13690, 2020.

[37] Hallawa A, Born T, Schmeink A, Dartmann G, Peine A, Martin L, Iacca G, Eiben AE, Ascheid G. Evo-RL: Evolutionary-driven reinforcement learning. In: Proc. of the 2020 Genetic and Evolutionary Computation Conf. Companion. Lille: ACM, 2020. 153–154. [doi: 10.1145/3449726.3459475]

[38] Chen DQ, Wang YZ, Gao W. Combining a gradient-based method and an evolution strategy for multi-objective reinforcement learning. Applied Intelligence, 2020, 50(10): 3301–3317. [doi: 10.1007/s10489-020-01702-7]

[39] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W. OpenAI Gym. arXiv:1606.01540, 2016.

附中文参考文献:

[3] 刘全, 翟建伟, 章宗长, 钟珊, 周倩, 章鹏, 徐进. 深度强化学习综述. 计算机学报, 2018, 41(1): 1–27. [doi: 10.11897/SP.J.1016.2018.00001]

[4] 刘建伟, 高峰, 罗雄麟. 基于值函数和策略梯度的深度强化学习综述. 计算机学报, 2019, 42(6): 1406–1438. [doi: 10.11897/SP.J.1016.2019.01406]

[22] 高卫峰, 罗宇婷, 原杨飞. 求解非线性方程组的智能优化算法综述, 控制与决策, 2021, 36(4): 769–778. [doi: 10.13195/j.kzyjc.2020.0379]

徐平安(1997－), 男, 硕士, 主要研究领域为强化学习, 深度学习, 深度强化学习.

郝少璞(1994－), 男, 硕士, 主要研究领域为强化学习, 深度强化学习, 模仿学习.

刘全(1969－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为智能信息处理, 自动推理, 机器学习.

张立华(1992－), 男, 博士生, CCF 学生会员, 主要研究领域为强化学习, 深度强化学习, 模仿学习.