计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201000163

基于自指导动作选择的近端策略优化算法

申 怡[1] 刘 全[1,2,3,4]
1 苏州大学计算机科学与技术学院 江苏 苏州 215006
2 苏州大学江苏省计算机信息处理技术重点实验室 江苏 苏州 215006
3 吉林大学符号计算与知识工程教育部重点实验室 长春 130012
4 软件新技术与产业化协同创新中心 南京 210000
(20184227052@stu.suda.edu.cn)

摘 要 强化学习领域中策略单调提升的优化算法是目前的一个研究热点,在离散型和连续型控制任务中都具有了良好的性能表现。近端策略优化(Proximal Policy Optimization,PPO)算法是一种经典策略单调提升算法,但PPO作为一种同策略(on-policy)算法,样本利用率较低。针对该问题,提出了一种基于自指导动作选择的近端策略优化算法(Proximal Policy Optimization based on Self-Directed Action Selection,SDAS-PPO)。SDAS-PPO算法不仅根据重要性采样权重对样本经验进行利用,而且增加了一个同步更新的经验池来存放自身的优秀样本经验,并利用该经验池学习到的自指导网络对动作的选择进行指导。SDAS-PPO算法大大提高了样本利用率,并保证训练网络模型时智能体能快速有效地学习。为了验证SDAS-PPO算法的有效性,将SDAS-PPO算法与TRPO算法、PPO算法和PPO-AMBER算法用于连续型控制任务Mujoco仿真平台中进行比较实验。实验结果表明,该方法在绝大多数环境下具有更好的表现。

关键词:强化学习;深度强化学习;策略梯度;近端策略优化;自指导
中图法分类号 TP181

Proximal Policy Optimization Based on Self-directed Action Selection

SHEN Yi[1] and LIU Quan[1,2,3,4]
1 School of Computer and Technology,Soochow University,Suzhou,Jiangsu 215006,China
2 Provincial Key Laboratory for Computer Information Processing Technology,Soochow University,Suzhou,Jiangsu 215006,China
3 Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education,Jilin University,Changchun 130012,China
4 Collaborative Innovation Center of Novel Software Technology and Industrialization,Nanjing 210000,China

Abstract The optimization algorithm of monotonous improvement of strategy in reinforcement learning is a current research hotspot,and it has achieved good performance in both discrete and continuous control tasks. Proximal policy optimization(PPO) algorithm is a classic strategy monotonic promotion algorithm,but it is an on-policy algorithm with low sample utilization. To solve this problem,an algorithm named proximal policy optimization based on self-directed action selection(SDAS-PPO)is proposed. The SDAS-PPO algorithm not only uses the sample experience according to the importance sampling weight,but also adds a synchronously updated experience pool to store its own excellent sample experience,and uses the self-directed network learned from the experience pool to guide the choice of actions. The SDAS-PPO algorithm greatly improves the sample utilization rate and ensures that the intelligent body can learn quickly and effectively when training the network model. In order to verify the effectiveness of the SDAS-PPO algorithm,the SDAS-PPO algorithm and the TRPO algorithm,PPO algorithm and PPO-AMBER algorithm are used in the continuous control task Mujoco simulation platform for comparative experiments. Experimental results show that this method has better performance in most environments.

Keywords Reinforcement learning,Deep reinforcement learning,Policy gradient,Proximal policy optimization,Self-directed

到稿日期:2020-10-28 返修日期:2021-03-11 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61772355,61702055,61502323,61502329);江苏省高等学校自然科学研究重大项目(18KJA520011,17KJA520004);吉林大学符号计算与知识工程教育部重点实验室资助项目(93K172014K04,93K172017K18);苏州市应用基础研究计划工业部分(SYG201422);江苏省高校优势学科建设工程资助项目
This work was supported by the National Natural Science Foundation of China(61772355,61702055,61502323,61502329),Jiangsu Province Natural Science Research University Major Projects(18KJA520011,17KJA520004),Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education,Jilin University(93K172014K04,93K172017K18),Suzhou Industrial Application of Basic Research Program Part(SYG201422) and A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.
通信作者:刘全(quanliu@suda.edu.cn)

1 引言

强化学习[1](Reinforcement Learning,RL)是一种经验决策方法,其目标是使智能体(Agent)与环境交互获得反馈,并为动作选择提供经验,从而学习到一个使 Agent 获得的累积回报奖赏最大化的最优策略。根据网络模型输出的不同,强化学习可以分为以下几类:基于值函数的强化学习方法[2]、基于策略的强化学习方法[3]和结合以上两类方法优势的行动者-评论家[4](Actor-Critic)方法。

基于值函数的强化学习方法的网络模型输出的是每个动作的价值,并根据最高价值来选择动作,适用于处理离散动作空间的强化学习任务,如 Sarsa[5]算法和 Q-Learning[6]算法。在离散动作强化学习任务中,通常遍历所有的动作来计算动作值函数,从而得到最优动作值函数。但对于连续动作空间的任务而言,动作数量是无限的。在这种情况下,基于值函数的强化学习方法难以确定一个固定动作,因此无法处理连续动作空间或大规模状态动作空间的强化学习任务[7]。

2013 年,谷歌 DeepMind 公司的 Mnih 等第一次将强化学习中的 Q-Learning 算法和深度卷积神经网络[8](Convolutional Neural Network,CNN)相结合,提出了深度 Q 网络[9](Deep Q-Network,DQN)结构。在若干次迭代后,其最终在 Atari2600 游戏中的表现超越了人类高水平玩家,使深度强化学习[10](Deep Reinforcement Learning,DRL)的研究成为了领域热点。其本质上属于采用神经网络作为值函数估计器的一类方法,主要优势在于它能利用深度神经网络对状态特征进行自动抽取,避免了人工定义状态特征带来的不准确性,使 Agent 能够在更原始的状态下进行学习。在此基础上,研究者们提出了许多改进的 Q 网络,如深度竞争 Q 网络[11](Dueling Deep Q-Network,Dueling-DQN)、深度双 Q 网络[12](Double Deep Q-Network,DDQN)、深度循环 Q 网络[13](Deep Recurrent Q-Network,DRQN)等。

基于策略的强化学习方法通过感官分析所处的环境,直接输出下一步要采取的各种动作的概率,然后根据概率采取行动,每个动作都有被选中的概率,能用一个概率分布在连续动作中选取特定动作进行学习,如策略梯度方法(Policy Gradient,PG)。AC(Actor-Critic)算法结合了以上两类方法的优势,行动者(Actor)网络基于概率选择动作,评论家(Critic)网络针对选择的动作给出动作的价值,加快了 PG 方法的学习速度。

对于连续动作空间任务而言,由于动作数目是不可确定的,也就无法给出每个动作的状态动作值,因此 DQN 算法对于连续动作空间任务并不适用。2014 年,Silver 等提出了适用于连续动作空间问题的确定性策略梯度[14](Deterministic Policy Gradient,DPG)算法。Lillicrap 等在 DPG 算法的基础上提出了深度确定性策略梯度[15](Deep Deterministic Policy Gradient,DDPG)算法,通过将深度学习与确定性行动者-评论家算法相结合,来学习大规模连续动作空间任务。Schulman 等提出置信域策略优化[16](Trust Region Policy Optimization,TRPO)算法,证明了该算法在一般随机梯度下改进策略的单调性。随后 OpenAI 公司于 2017 年提出了近端策略优化[17](Proximal Policy Optimization,PPO)算法,使用一阶优化方法进一步简化了 TRPO 算法的计算复杂度。该算法被应用于大规模问题中。

近年来,在 PPO 算法研究方面,Han 等提出了基于自适应多批经验回放的近端策略优化[18](Adaptive Multi-Batch Experience Replay,PPO-AMBER)算法,通过重用重要性采样权重小的旧样本对算法进行优化。由于 PPO-AMBER 算法也是在 PPO 算法的基础上利用样本经验对原算法进行改进,因此在本文第 4 节中将 SDAS-PPO 算法与之进行横向对比实验。Liu 等通过运用动作依赖的斯坦因控制变量来降低方差,以对 PPO 算法进行优化[19]。Ling 等提出了一种 Agent 在稀疏环境下能进行有效探索的方法,即多路径策略优化(Multi-Path Policy Optimization,MPPO)[20]。Pan 等提出的基于模型的探查策略优化(Policy Optimization with Model-based Explorations,POME)[21]方法,在应用于 PPO 算法时也得到了良好的实验效果。Touati 等将非策略差异正则化方法纳入 PPO 算法中[22],以确保更加稳定的策略优化。Li 等将 PPO 算法与分层强化学习相结合,得到了一种有效联合训练分层结构各个级别的基于策略的强化学习方法[23]。

PPO 算法使用截断的代理目标函数对当前策略进行更新,以实现稳定的学习。但由于其代理目标函数的结构会引起截断样本的梯度逐渐消失,导致在连续动作空间任务中 Agent 学习效率低。PPO 算法按照新旧策略的概率分布对目标函数进行加权,即使用重要性采样权重方法对样本经验进行利用,但是动作的选择依然是随机的,因此 Agent 在学习过程中的随机性强,不能进行有针对性的高效学习。因此,为引导 Agent 向优秀经验轨迹学习,提高优秀样本的利用率,最优化累计回报,本文提出了 SDAS-PPO 算法。本文的主要贡献如下:

(1)增加了一个同步更新的经验池,用于存放自身的优秀经验,并在训练过程中对优秀经验轨迹进行同步更新。

(2)指导网络由优秀经验池中的优秀经验训练得到,并对动作的选择进行指导。

(3)在具有连续动作空间的 Mujoco 仿真机器人平台的 6 个环境中,将 SDAS-PPO 算法与 TRPO 算法、PPO 算法、PPO-AMBER 算法进行横向比较实验。实验结果表明,SDAS-PPO 算法能够取得更好的实验效果。

本文第 2 节介绍了策略梯度方法、Actor-Critic 方法以及 PPO 算法;第 3 节阐述了 SDAS-PPO 算法的模型架构以及算法流程;第 4 节通过对比实验验证算法的有效性,并分析了算法的优点以及存在的问题;最后总结全文并展望未来。

2 背景知识

2.1 强化学习

马尔可夫决策过程(Markov Decision Process,MDP)是强化学习中的基础概念,可对强化学习问题建模。带折扣的 MDP 可定义为一个五元组形式 $M=(S,A,P,R,\gamma)$,其中 S 是环境中所有状态的集合,Agent 在 t 时刻所处的环境状态记为 $s_t \in S$;A 是 Agent 可选择动作的集合,Agent 在 t 时刻执行的动作记为 $a_t \in A$;$P:S\times A\times S\rightarrow[0,1]$ 是状态转移概率函

数，Agent 在所处环境状态为 s_t 时，根据策略执行动作 a_t 到达下一状态 s_{t+1} 的概率记为 $s_{t+1}\sim P(s_t,a_t)$；R 是立即奖赏函数，Agent 在 t 时刻状态 s_t 下根据策略执行动作 a_t 获得的立即奖赏记为 $R(s_t)\in R$；$\gamma\in(0,1)$ 是折扣因子，γ 越大表示未来回报对当前情节动作选择的影响越大[24]。强化学习问题的最终目标是获得最大化累积奖赏条件下的最优策略，当 Agent 遵循的策略在所有状态下所得的期望奖赏大于等于其他任何策略所得的期望奖赏时，该策略就称为最优策略，记为 $h^*(s,a)=\operatorname{argmax} Q^h(s,a)$。其中，$Q^h(s,a)=E_h(R_t\mid s_t=s,a_t=a)$ 为状态动作值函数，表示 Agent 遵循策略 h，在状态 s_t 下执行动作 a_t 后至情节结束所得的期望奖赏。类似地，状态值函数 $V^h(s)=E_h(R_t\mid s_t=s)$ 表示 Agent 遵循策略 h 从状态 s 到情节结束所得的期望奖赏。在连续状态动作空间的强化学习问题上，难以将状态动作抽象表示出来并对其进行计算，策略梯度方法的提出能较好地解决上述连续动作空间的相应问题。

2.2　策略梯度

策略梯度方法是解决强化学习问题的一个重要方法。基于值函数的强化学习方法是先直接通过迭代计算值函数，再根据值函数来改善策略与之不同的是，基于策略的强化学习方法通过直接对策略进行参数化来计算策略可能更新的方向，以优化策略。由于每次更新的幅度不大，因此参数化策略的方法更容易收敛。

策略梯度方法通过带有参数 θ 的随机策略网络 $\pi_\theta(s_t,a_t)$ 来对策略进行参数化，在训练过程中通过优化梯度来更新策略，最终使目标函数达到最优。目标函数可以定义为关于策略的期望回报，如式(1)所示：

$$J(\theta)=E_\pi\Big[\sum_{t=0}^{\infty}\gamma^t r_t\Big]=\sum_{\tau}P(\tau;\theta)R(\tau)\tag{1}$$

其中，$P(\tau;\theta)$ 表示序列 τ 出现的概率。

通过梯度下降的方法对目标函数进行迭代更新，即求目标函数关于 θ 的导数并更新其参数，如式(2)所示：

$$\theta_{t+1}=\theta_t+\alpha\nabla_\theta J(\theta)\tag{2}$$

其中，α 为网络参数更新的步长，随着迭代的不断更新其最终得到收敛。

2.3　行动者-评论家方法

Actor-Critic 方法将值函数方法和策略梯度方法的优点相融合，用于选择动作的策略结构被称为 Actor，用于评估策略的部分被称为 Critic。Critic 对 Actor 执行动作后更新的策略参数进行评价，评价以时间差分的形式返回；Actor 根据 Critic 的反馈进行策略参数的更新。

在 AC 算法中，动作的选择不再通过最大化动作值函数来选取，而是直接由 Actor 根据参数来获取。因此，AC 算法适用于动作值函数难以被抽象表示出来的连续状态动作空间最优策略的求解。其中 TD 误差[25]是 Critic 评价 Actor 的依据，如果 TD 误差大于零，说明 Actor 所采取的动作有利于寻找最优策略，该动作被选择的概率会增大；反之，如果 TD 误差小于零，说明 Actor 所采取的动作会减少期望奖赏，该动作被选择的概率会减小。TD 误差的更新公式如式(3)所示：

$$\delta_t=\sum_{i=0}^{n-1}\gamma^i r_{t+i}+\gamma^n V(s_{t+n})-V(s_t)\tag{3}$$

在 Agent 学习的过程中可以获得动作的选择概率，并根据奖赏的情况随时进行调整。这种随机概率的方式能直观地对动作选择进行指导，在解决实际的强化学习问题中具有很大的优势。

2.4　近端策略优化算法

PPO 是在 TRPO 的基础上进行的改进。TRPO 算法的每次迭代都尝试从当前的策略中选择一个合适的步长，使新策略得到的累计回报单调递增。其目标函数如式(4)所示：

$$\begin{aligned}J_\theta(\widetilde{\theta})&=\mathbb{E}_{s_t\sim\rho_{\pi_\theta},a_t\sim\pi_{\widetilde{\theta}}}[A_{\pi_\theta}(s_t,a_t)]\\&=\mathbb{E}_{s_t\sim\rho_{\pi_\theta},a_t\sim\pi_\theta}[k_t(\widetilde{\theta})A_{\pi_\theta}(s_t,a_t)]\end{aligned}$$

受限于：

$$\overline{D}_{KL}^{\rho_{\pi_\theta}}(\theta,\widetilde{\theta})\leqslant\delta\tag{4}$$

其中，$A_{\pi_\theta}(s_t,a_t)=Q_{\pi_\theta}(s_t,a_t)-V_{\pi_\theta}(s_t)$ 是优势函数，$k_t(\widetilde{\theta})=\dfrac{\pi_{\widetilde{\theta}}(a_t\mid s_t)}{\pi_\theta(a_t\mid s_t)}$ 是重要性采样权重，$\pi_{\widetilde{\theta}}(a_t\mid s_t)$ 表示新策略的概率分布，$\pi_\theta(a_t\mid s_t)$ 表示旧策略的概率分布。TRPO 算法对模型的每一轮优化都划定了置信区域，为了保证优化的稳定性，每次优化都不超过这个范围，而这个范围的限制是由 KL 散度控制的，要求新策略与旧策略的概率分布不能相差太大，以保持稳定的增长。

为了控制策略的更新幅度，PPO 算法采用了截断的代理目标函数。该算法将 $k_t(\widetilde{\theta})$ 即新旧策略的比值限制在一个区域中，通过控制区域的大小来限制更新的步幅。相比 TRPO 中使用 KL 散度进行限制，PPO 中 $k_t(\widetilde{\theta})$ 的限制更加简单也更容易实现。PPO 算法的目标函数如式(5)所示：

$$\begin{aligned}J^{CLIP}(\widetilde{\theta})=\mathbb{E}_{s_t\sim\rho_{\pi_\theta},a_t\sim\pi_\theta}[&\min(k_t(\widetilde{\theta}),clip(k_t(\widetilde{\theta}),1-\epsilon,\\&1+\epsilon))*\hat{A}_\theta(s,a)]\end{aligned}\tag{5}$$

其中，$clip(k_t(\widetilde{\theta}),1-\epsilon,1+\epsilon)$ 将重要性采样权重 $k_t(\widetilde{\theta})$ 约束在 $(1-\epsilon,1+\epsilon)$ 范围内，ϵ 为超参数。min 函数的作用是对原始项和截断项取最小，因此当策略更新的偏移超出预定区间时，截断项就会起到限制的作用。

PPO 算法还运用了优势函数估计方法和增加额外熵奖励的优化方法来进一步提升其性能。使用泛化优势估计(Generalized Advantage Estimation，GAE)构造优势函数能够降低方差，使算法不会产生较大的波动。GAE 的计算式如式(6)所示：

$$\hat{A}_t=\delta_t+(\gamma\lambda)\delta_{t+1}+\cdots+(\gamma\lambda)^{T-t+1}\delta_{T-1}\tag{6}$$

其中，$\delta_t=r_t+\gamma V(s_{t+1})-V(s_t)$。

将 PPO 算法应用在策略(Actor)和值函数(Critic)共享参数的网络结构上时，除了截断回报之外，目标函数还加上了关于值函数估计的误差项以及策略模型的熵正则项，用于鼓励探索。因此，优化后的目标函数如式(7)所示：

$$\begin{aligned}J(\widetilde{\theta})=\mathbb{E}_{s_t\sim\rho_{\pi_\theta},a_t\sim\pi_\theta}[&J^{CLIP}(\widetilde{\theta})-c_1(V_\theta(s)-V_{\text{target}})^2+c_2H\\&(s,\pi_\theta)]\end{aligned}\tag{7}$$

其中，c_1 和 c_2 为两个常数超参数；$(V_\theta(s)-V_{\text{target}})^2$ 是状态值函数的均方误差，误差越小越好；$H(s,\pi_\theta)$ 表示策略 π_θ 的熵值，熵越大越好。

PPO 算法是一种 on-policy 算法。在对目标函数的梯度进行策略更新时，每一次计算梯度都需要使用当前最新的策

略模型进行交互采样来得到相应的样本序列，以此来进行梯度的计算。更新模型之后，使用过的样本经验将只能通过重要性采样权重中的旧策略概率分布来得到利用，因此，优秀样本经验的使用率非常低。

3 基于自指导动作选择的近端策略优化算法

本节分析了自指导动作选择方法以及 SDAS-PPO 算法的结构和思想，然后描述了算法的流程并进行了分析。

3.1 自指导动作选择

PPO 算法利用旧策略中的样本经验来更新当前策略，主要通过重要性采样权重来实现，但这种样本经验的应用存在局限性，不能有效学习到旧策略中的优秀样本经验。在 PPO 算法的训练过程中，Agent 自身学习到的一些高回报情节轨迹并没有被有效利用。因此，为利用这些经验，SDAS-PPO 算法中定义了一个优秀经验池 B'，用于存储高回报的情节轨迹，并定义了一个指导网络 $G(ss_t|\theta^g)$，根据优秀经验池中的情节进行同步更新。

在训练过程中，情节轨迹以 $(s_t, a_t, r_t, s_{t+1}, v_t)$ 的形式存储到经验池 B 中。而优秀经验池 B' 与之不同的是，将经验以 $(s_t, a_t, s_{t+1}, a_{t+1}, \cdots, s_T)$ 的形式存储，T 为情节结束的标志。优秀经验池中的经验并非对每个时间步都进行了存储，而是在每个情节结束时判断该情节的累计回报是否大于优秀经验池中情节回报的平均值。如果大于该平均值，则该情节被判定为优秀情节，并将该情节经验放入优秀经验池中，否则舍去。判断情节是否为优秀情节的标准如式(8)所示：

$$G_n = r_1 + r_2 + \cdots + r_T \geqslant \bar{G}_{m,n} \tag{8}$$

其中：

$$\bar{G}_{m,n} = \frac{G_1 + G_2 + \cdots + G_m}{m} \tag{9}$$

式(9)表示在优秀经验池 B' 中的 m 个情节的回报均值，n 表示第 n 个情节，G_n 表示第 n 个情节的回报。

由式(8)可知，当前情节的累计回报大于或等于经验池中情节的平均回报时被判断为优秀经验，会被放入优秀经验池。当优秀经验池中的情节存储到第 m 个，并且第 $m+1$ 个情节的累计回报大于前 m 个情节的回报均值时，会将第 1 个情节的轨迹舍去，将第 $m+1$ 个情节放入经验池，体现在式(9)中的计算回报均值时分子变为 $G_2 + G_3 + \cdots + G_m + G_{m+1}$。因为优秀经验池 B' 的大小是固定的，所以在将情节轨迹放入的同时就更新了经验池中的优秀情节，保证了指导网络在指导动作选择时选择的都是最近最优的样本轨迹。

指导网络 $G(ss_t|\theta^g)$ 是在优秀经验池 B' 中通过监督学习学习得到的，根据优秀经验池中样本轨迹的自我更新而同步升级。指导网络在训练时通过最小化损失函数来更新，其损失函数如式(10)所示：

$$L^{\mathrm{GUIDE}}(\theta^g) = \frac{1}{N}\sum_{m=1}^{N}\left[(G(ss_m|\theta^g) - a_m)^2\right] \tag{10}$$

指导网络是由 Agent 自身学习到的优秀样本经验训练出来的，没有外界的指导，因此是一种自指导的方式。动作的选择实际上就是 Agent 从当前策略的概率分布中根据一定的概率选择出动作的过程。加上指导网络的动作指导之后，当前策略的概率分布如式(11)所示：

$$\pi_{\tilde{\theta}}(a_t|s_t) + \tau(G(ss_t|\theta^g) - \pi_{\tilde{\theta}}(a_t|s_t)) \tag{11}$$

其中，τ 为指导参数，约束在(0,1)之间，本文第 4.2 节在 HalfCheetah-v2 环境中进行了指导参数的对比实验。

经过指导网络对动作选择的指导，当前策略的概率分布也随之改变，相应的重要性采样权重如式(12)所示：

$$c_t(\tilde{\theta}) = \frac{\pi_\theta(a_t|s_t) + \tau(G(ss_t|\theta^g) - \pi_\theta(a_t|s_t))}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)} \tag{12}$$

因此，SDAS-PPO 算法中更新 Actor 网络的目标函数如式(13)所示：

$$J^{\mathrm{SDAS}}(\tilde{\theta}) = \mathbb{E}_{s_t\sim\rho_{\pi_{\tilde{\theta}}}, a_t\sim\pi_\theta}\left[\min(c_t(\tilde{\theta}), clip(c_t(\tilde{\theta}), 1-\epsilon, 1+\epsilon)) * \hat{A}_t(s_t, a_t)\right] \tag{13}$$

SDAS-PPO 算法沿用了 PPO 算法在目标函数中加入截断项的思想，式(13)中的 $clip(c_t(\tilde{\theta}), 1-\epsilon, 1+\epsilon)$ 是将经过指导网络指导动作选择后的重要性采样权重 $c_t(\tilde{\theta})$ 约束在 $(1-\epsilon, 1+\epsilon)$ 范围内，ϵ 为限制参数。

Critic 网络更新时，采用了最小化限制损失函数的方式，将当前状态值函数与上一时刻的状态值函数的差值限制在 $(-\epsilon, \epsilon)$ 之间，由此得出限制后的状态值函数如式(14)所示：

$$V_{clip}(\omega) = V_{t-1}(\omega) + clip(V_t(\omega) - V_{t-1}(\omega), -\epsilon, \epsilon) \tag{14}$$

再将限制后的状态值函数计算出来的均方误差与当前状态值函数所计算出的均方误差进行比较以取最小值，有效降低了 Critic 网络的损失，使 SDAS-PPO 算法的学习效果得到显著的提升。

3.2 算法结构

SDAS-PPO 算法使用了 Actor-Critic 方法的框架。在训练过程中，Actor 网络更新的目标函数中状态动作值函数是由 Critic 网络评估策略后提供的。在每一个时间步 t，Actor 网络根据当前策略以及指导网络的指导选择并执行动作 a_t，产生样本经验 $e_i = (s_t, a_t, r_t, s_{t+1}, v_t)$ 后，将其存储到经验池中。在每个情节结束之后，通过判断该情节是否有资格放入优秀经验池，将优秀经验轨迹 $g_i = (s_t, a_t, s_{t+1}, a_{t+1}, \cdots, s_T)$ 存储到优秀经验池中，并对指导网络进行同步更新。SDAS-PPO 算法的模型架构如图 1 所示。

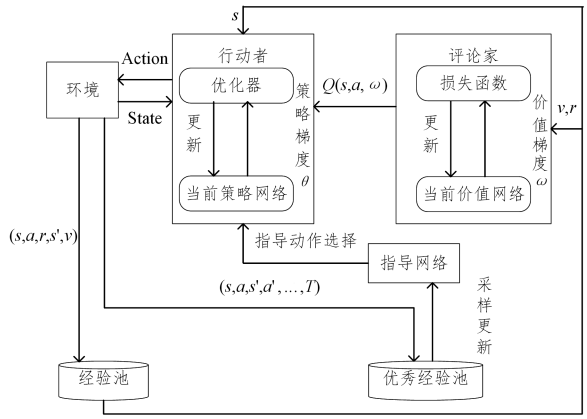

图 1 SDAS-PPO 算法的模型架构

Fig. 1 Model architecture of SDAS-PPO algorithm

3.3 算法描述

为了提高优秀样本经验的利用率，加快策略优化的速度，基于自指导动作选择的近端策略优化算法通过增加一个自身

优秀经验池来存储优秀经验轨迹的方式，训练了指导网络，以对动作选择进行指导。算法的流程如算法 1 所示。

算法 1　基于自指导动作选择的近端策略优化算法

输入：最大情节数 E，最大时间步 T，学习率 γ，GAE 参数 λ，限制参数 ϵ，指导参数 τ，优秀经验池中情节数 m

输出：策略网络参数 θ

1. 初始化 actor 网络参数 θ，critic 网络参数 ω，指导网络参数 θ^g，初始化经验池 B，优秀经验池 B′，最大情节数 E，最大时间步 T，计算初始化 $\bar{G}_{m,n}$；
2. for episode←1 to E do
3.　情节回报 G=0，空的情节轨迹 Trace，获取初始状态 s_t
4.　for t←1 to T do
5.　获得经过指导网络指导的当前策略概率分布：
$\pi_{\tilde{\theta}}(a_t|s_t)+\tau(G(s\ s_t|\theta^g)-\pi_{\tilde{\theta}}(a_t|s_t))$
6.　选择并执行动作 a_t，获得立即奖赏 r_t 和下一状态 s_{t+1}
7.　计算累计回报 $G=G+r_t$
8.　将经验样本 $e_i=(s_t,a_t,r_t,s_{t+1},v_t)$ 存入经验缓冲池中
9.　将经验轨迹 $g_i=(s_t,a_t,s_{t+1},a_{t+1},\cdots,s_T)$ 存入 Trace
10.　从经验池 B 中随机抽取 N 个经验样本
11.　根据 critic 网络的值函数来计算优势函数：
$\hat{A}_t(s_t,a_t)=Q_t(s_t,a_t,\omega)-V_t(\omega)$
12.　计算重要性采样权重：
$$c_t(\tilde{\theta})=\frac{\pi_{\tilde{\theta}}(a_t|s_t)+\tau(G(s\ s_t|\theta^g)-\pi_{\tilde{\theta}}(a_t|s_t))}{\pi_\theta(a_t|s_t)}$$
13.　通过最大化 SDAS-PPO 的目标函数更新 actor 网络：
$$J^{SDAS}(\tilde{\theta})\approx\frac{1}{N}\sum_{t=1}^{N}[\min(c_t(\tilde{\theta}),clip(c_t(\tilde{\theta}),1-\epsilon,1+\epsilon))*\hat{A}_t(s_t,a_t)]$$
14.　计算限制值函数：
$V_{clip}(\omega)=V_{t-1}(\omega)+clip(V_t(\omega)-V_{t-1}(\omega),-\epsilon,\epsilon)$
15.　通过最小化限制损失函数来更新 critic 网络：
$$L^{SDAS}(\omega)=\frac{1}{N}\sum_{t=1}^{N}[\min((V_{clip}(\omega)-G)^2,(V_t(\omega)-G)^2)]$$
16.　通过最小化损失函数来更新指导网络：
$$L^{GUIDE}(\theta^g)=\frac{1}{N}\sum_{m=1}^{N}[(G(ss_m|\theta^g)-a_m)^2]$$
17.　更新网络参数 $\theta\leftarrow\tilde{\theta}$
18.　end for
19.　如果 $G\geqslant\bar{G}_{m,n}$，则将轨迹 Trace 放入经验池 B′中，并重新计算 $\bar{G}_{m,n}$
20. end for

算法 1 中，第 5 步为对动作选择增加指导的过程，第 6－9 步为产生样本经验以及将经验存储到经验池的过程，第 10－13 步通过最大化 SDAS-PPO 算法的目标函数来更新 Actor 网络，第 14－16 步通过最小化限制损失函数的方法来更新 Critic 网络，第 19 步为判断该情节是否为优秀情节，并将通过审核的情节经验放入优秀经验池的过程。

4　实验及结果分析

4.1　实验环境与参数设置

为评估 SDAS-PPO 算法在具体强化学习任务中的表现，本文在 OpenAI 公司开源的 Gym 平台[26]上的 Mujoco 环境[27]中对算法进行实验验证。Mujoco 是一个物理模拟器，包含了多种具有连续动作空间的强化学习任务，用于对强化学习方法、深度强化学习方法以及模仿学习方法进行测试。

本文首先探究了指导参数对算法性能的影响，然后将 SDAS-PPO 与 PPO-AMBER，TRPO 和 PPO 这 3 种算法进行了对比。实验的环境选取 Mujoco 环境下具有代表性的 6 个任务进行对比实验，即 HalfCheetah-v2，Ant-v2，Swimmer-v2，Reacher-v2，Humanoid-v2 和 HumanoidStandup-v2。表 1 列出了不同任务输入的状态维度和输出的动作维度。

表 1　Mujoco 平台上 6 个任务的状态维度和动作维度

Table 1　State dimension and action dimension of six tasks on the Mujoco platform

任务名称	状态维度	动作维度
HalfCheetah-v2	17	6
Ant-v2	111	8
Swimmer-v2	8	2
Reacher-v2	11	2
Humanoid-v2	376	17
HumanoidStandup-v2	376	17

为保证对比实验的公平性和有效性，本文实验所使用的 TRPO 和 PPO 算法来自 OpenAI baselines[28]深度强化学习算法集，所使用的 PPO-AMBER 算法的网络结构和参数设置与文献[18]中的设置保持一致。在 SDAS-PPO 算法中，Actor 网络和 Critic 网络结构的设置与 PPO 算法一样。经验池的大小为 100 000，优秀经验池的大小为 20 000，优秀经验池是经验池大小的 1/5。每批次选取样本的数量为 32，每个情节的最大时间步数为 2048，超过 2048 步时情节就重新开始。每个 Mujoco 环境下训练的总时间步数为 200 万，但在 Reacher 环境下训练的时间步数到达 100 万时，就收敛到一个稳定的策略，因此 Reacher 环境训练的总时间步数为 100 万。SDAS-PPO 算法中，限制参数 $\epsilon=0.2$，指导参数 $\tau=1\times10^{-5}$，学习率 $\gamma=0.99$，GAE 参数 $\lambda=0.95$，优秀经验池中情节数 $m=50$。

4.2　指导参数对比实验

为了探究 SDAS-PPO 算法中指导参数 τ 对算法性能的影响，使用不同 τ 的值在 HalfCheetah 环境中进行对比实验，实验结果如图 2 所示。

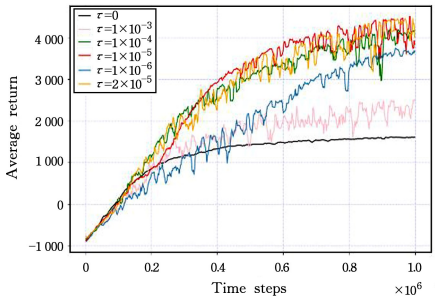

图 2　指导参数 τ 的对比实验结果图

Fig. 2　Comparative experimental results of guidance parameter τ

从图 2 中可以看出，当 $\tau=1\times10^{-5}$ 时实验效果明显优于其他 τ 值，因此在第 4.3 节的对比实验中选取指导参数 $\tau=1\times10^{-5}$。虽然当 $\tau=1\times10^{-4}$ 和 $\tau=2\times10^{-5}$ 时、训练步数在

30 万之前的平均回报优于当 $\tau=1\times10^{-5}$ 时的平均回报，但是在 30 万步之后直至 100 万步时，$\tau=1\times10^{-5}$ 能让算法稳定且不断向上优化，直至收敛到一个较好的策略。当 $\tau=0$ 时，算法中动作的选择不受指导网络的指导，由自身随机进行选择，因此训练得到的奖励回报较低。

4.3 性能对比实验

为了验证本文提出的 SDAS-PPO 算法的有效性，本文评估了 SDAS-PPO，PPO，PPO-AMBER 和 TRPO 算法在 HalfCheetah-v2，Ant-v2，Swimmer-v2，Reacher-v2，Humanoid-v2 和 HumanoidStandup-v2 上的性能表现，每个环境使用了 5 组不同的随机种子进行训练，最终结果由所有随机种子取平均得到，曲线图中横坐标为训练的时间步数，纵坐标是当前训练时间步的平均回报。

实验结果如图 3 所示。

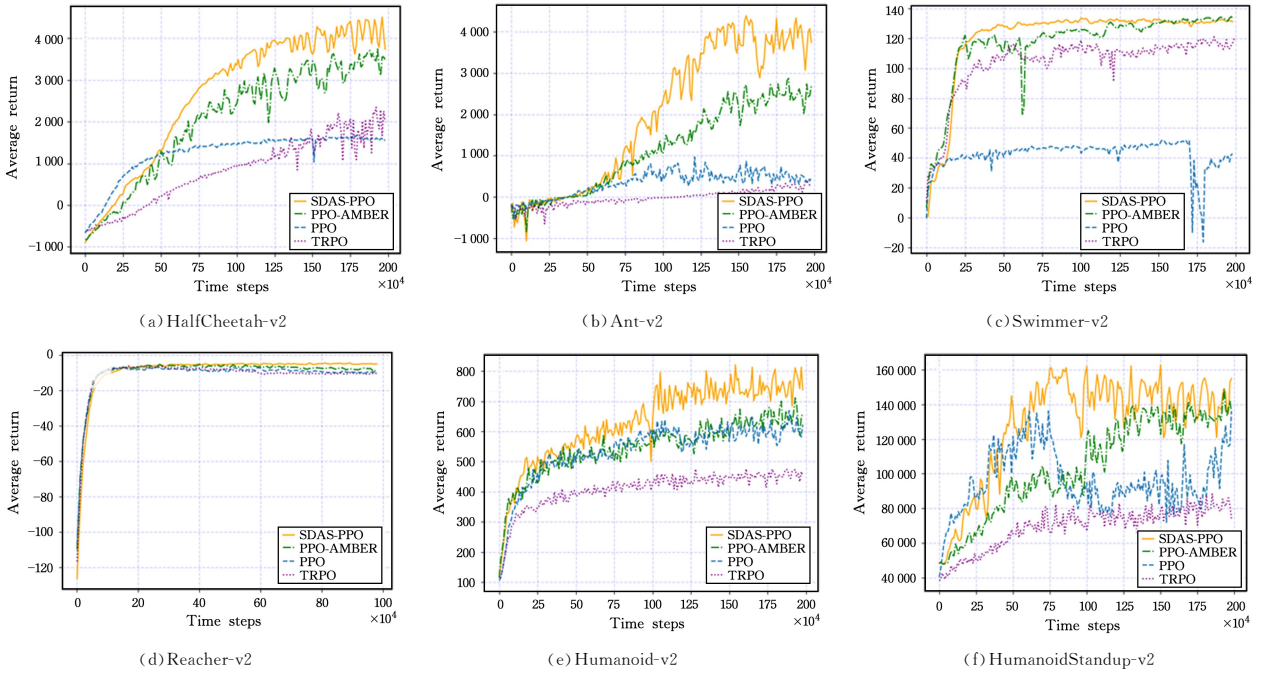

图 3 SDAS-PPO 算法与 PPO，PPO-AMBER，TRPO 算法的对比实验结果

Fig. 3 Comparative experimental results of SDAS-PPO algorithm and PPO, PPO-AMBER, TRPO algorithms

从图 3 可以看出，SDAS-PPO 算法在大多数 Mujoco 环境中的表现都优于 PPO 和 PPO-AMBER，并且收敛速度更快，说明自身优秀经验指导动作选择的方法能够引导 Agent 在训练时不断学习累计奖赏高的策略，最终取得优异的奖赏回报。

在 HalfCheetah 和 HumanoidStandup 实验环境中，SDAS-PPO 算法前期学习的效果相比其他对比算法优势不明显，这是因为训练前期动作的探索较多，优秀的高回报经验轨迹并不多，导致自身经验指导的作用并没有发挥出来。随着时间步与自身优秀经验的增加，指导动作选择的作用才逐渐展现出来，为训练后期带来更好的表现。

此外，在训练过程中，Humanoid，Ant 和 HumanoidStandup 实验环境的波动较大，主要原因在于这 3 个实验环境都是状态维度高的环境。在高纬度连续动作空间的深度强化学习任务中，由于其环境维度较高，状态动作空间是无限的，导致 Agent 探索动作较多，难以获得稳定的策略，因此获取的优秀样本经验较少。优秀样本经验的不足使得其对动作的指导出现偏差，方差较大，因此实验环境的波动性也较大。

Reacher 环境中，Agent 通过控制手臂使其末端伸向指定位置，到达的用时越短，奖赏就越高。由于该任务对动作的选择并不敏感，实验环境中也没有别的干扰因素，因此算法能够快速收敛，且不会产生较大的波动，但本文方法在实验中的效果更优。

算法的好坏不仅体现在训练过程中，训练结束后模型在测试阶段的表现也是算法有效性的一个重要参考标准。为了展示 SDAS-PPO 算法在测试阶段的表现，将训练后的模型在测试集中进行测试。在测试集上选取 5 组随机种子，每个实验环境下测试 20 000 步，获得的最终平均奖赏的结果如表 2 所列。

表 2 6 个 Mujoco 环境测试集的结果

Table 2 Results of 6 Mujoco environmental test sets

实验环境	Half-Cheetah	Ant	Swimmer	Reacher	Humanoid	Humanoid-Standup
SDAS-PPO	3835.68	3188.88	131.29	−4.07	800.16	156991.02
PPO	1615.92	531.94	136.47	−10.42	766.62	125026.71
PPO-AMBER	3580.79	2795.74	32.48	−6.53	619.58	133160.18
TRPO	2414.39	765.79	119.21	−9.90	495.69	82539.52

从表 2 中发现，SDAS-PPO 算法在大部分 Mujoco 实验环境中性能都有提升，说明本文提出的基于自身优秀经验指导的近端策略优化算法在大部分任务中相比其他基于概率的深度强化学习算法有一定的优势，能够更好地引导 Agent 向高回报的经验轨迹学习，得到更优的奖赏回报。上述实验结果还验证了 SDAS-PPO 算法的稳定性，实验结果不会因随机种子的影响而发生改变。

结束语 本文针对连续动作空间下的近端策略优化算法中样本经验利用率不高的问题，提出了基于自指导动作选择

的近端策略优化算法。该算法通过增加经验池存储自身高回报的经验轨迹，训练得到一个指导网络，并对动作的选择进行指导，提高了优秀样本经验的利用率，引导 Agent 向高回报的情节学习，提高了算法的学习效率。为了验证该算法的稳定性和有效性，本文在 6 个 Mujoco 环境下进行对比实验，实验结果表明，当指导参数 $\tau=10\times^{-5}$ 时，SDAS-PPO 能够更有效地学习到一个好的策略。

近端策略优化算法中，还存在训练过程中 Agent 学习步伐迈得过小导致的更新缓慢问题和构造截断目标函数导致的高方差问题，未来的一些工作可以在本文的基础上对上述两个问题进行进一步的研究。

参 考 文 献

[1] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction[M]. Cambridge, MA: MIT Press, 1998: 6-22.

[2] PARR R, LI L, TAYLOR G, et al. An Analysis of Linear Models, Linear Value-Function Approximation, and Feature Selection for Reinforcement Learning[C]// International Conference on Machine Learning. 2008.

[3] KOHL N, STONE P. Policy gradient reinforcement learning for fast quadrupedal locomotion[C]// IEEE International Conference on Robotics & Automation. IEEE, 2004.

[4] BARTO A G, SUTTON R S, ANDERSON C W. Neuronlike adaptive elements that can solve difficult learning control problems[J]. IEEE Transaction on Systems, Man and Cybernetics, 1983, 13(5): 834-846.

[5] SEIJEN H V, HASSELT H V, WHITESON S, et al. A theoretical and empirical analysis of Expected Sarsa[C]// Adaptive Dynamic Programming and Reinforcement Learning, 2009. IEEE, 2009.

[6] KIUMARSI B, LEWIS F L, MODARES H, et al. Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics[J]. Automatica, 2014, 50(4): 1167-1175.

[7] TANGKARATT V, ABDOLMALEKI A, SUGIYAMA M. Guide Actor-Critic for Continuous Control[J]. arXiv: 1705.07606, 2017.

[8] KRIZHEVSKY A, SUTSKEVER I, HINTON G. ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in Neural Information Processing Systems, 2012, 25: 1097-1105.

[9] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[10] LIU Q, ZHAI J W, ZHANG Z Z, et al. A review of deep reinforcement learning[J]. Chinese Journal of Computers, 2018, 41(1): 1-27.

[11] WANG Z, SCHAUL T, HESSEL M, et al. Dueling network architectures for deep reinforcement learning[J]. Proceedings of the 33nd International Conference on Machine Learning. New York, USA, 2016: 692-700.

[12] VAN HASSELT H, GUEZ A, SILVER D. Deep Reinforcement Learning with Double Q-Learning[C]// Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. Phoenix, USA, 2016: 2094-2100.

[13] HAUSKNECHT M, STONE P. Deep recurrent q-learning for partially observable mdps[C]// 2015 AAAI fall symposium series. 2015.

[14] SILVER D, LEVER G, HEESS N, et al. Deterministic policy gradient algorithms[C]// Proc. of the 31st Int. Conf. on Machine Learning. New York: ACM, 2014: 387-395.

[15] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[J]. arXiv: 1509.02971, 2015.

[16] SCHULMAN J, LEVINE S, ABBEEL P, et al. Trust region policy optimization[C]// International Conference on Machine Learning. PMLR, 2015: 1889-1897.

[17] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms[J]. arXiv: 1707.06347, 2017.

[18] HAN S, SUNG Y. Amber: Adaptive multi-batch experience replay for continuous action control[J]. arXiv: 1710.04423, 2017.

[19] LIU H, FENG Y, MAO Y, et al. Sample-efficient policy optimization with stein control variate[J]. arXiv: 1710.11198, 2017.

[20] LING P, CAI Q P, HUANG L B. Multi-Path Policy Optimization[C]// International Conference on Autonomous Agents and Multi Agent Systems. 2020: 1001-1009.

[21] PAN F, CAI Q, ZENG A X, et al. Policy optimization with model-based explorations[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33: 4675-4682.

[22] TOUATI A, ZHANG A, PINEAU J, et al. Stable policy optimization via off-policy divergence regularization[C]// Conference on Uncertainty in Artificial Intelligence. PMLR, 2020: 1328-1337.

[23] LI A, FLORENSA C, CLAVERA I, et al. Sub-policy Adaptation for Hierarchical Reinforcement Learning[C]// International Conference on Learning Representations. 2019.

[24] YOSHIDA N, UCHIBE E, DOYA K. Reinforcement learning with state-dependent discount factor[C]// IEEE Third Joint International Conference on Development & Learning & Epigenetic Robotics. IEEE, 2013.

[25] FU Q M, LIU Q, SUN H K, et al. A second-order TD Error fast Q(λ) algorithm[J]. Pattern Recognition and Artificial Intelligence, 2013(3): 282-292.

[26] BROCKMAN G, CHEUNG V, PETTERSSON L, et al. Openai gym[J]. arXiv: 1606.01540, 2016.

[27] TODOROV E, EREZ T, TASSA Y. MuJoCo: A physics engine for model-based control[C]// 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2012.

[28] DHARIWAL P, HESSE N, MANNING C, et al. OpenAI baselines [OL]. GitHub, 2017. https://github.com/openai/baselines.

SHEN Yi, born in 1995, postgraduate. Her main research interests include deep reinforcement learning and so on.

LIU Quan, born in 1969, Ph.D, professor, is a member of China Computer Federation. His main research interests include deep reinforcement learning and automated reasoning.