异策略深度强化学习中的经验回放研究综述

胡子剑 [1] 高晓光 [1] 万开方 [1] 张乐天 [2] 汪强龙 [1] NERETIN Evgeny [3]

摘 要 作为一种不需要事先获得训练数据的机器学习方法, 强化学习 (Reinforcement learning, RL) 在智能体与环境的不断交互过程中寻找最优策略, 是解决序贯决策问题的一种重要方法. 通过与深度学习 (Deep learning, DL) 结合, 深度强化学习 (Deep reinforcement learning, DRL) 同时具备了强大的感知和决策能力, 被广泛应用于多个领域来解决复杂的决策问题. 异策略强化学习通过将交互经验进行存储和回放, 将探索和利用分离开来, 更易寻找到全局最优解. 如何对经验进行合理高效的利用是提升异策略强化学习方法效率的关键. 首先对强化学习的基本理论进行介绍; 随后对同策略和异策略强化学习算法进行简要介绍; 接着介绍经验回放 (Experience replay, ER) 问题的两种主流解决方案, 包括经验利用和经验增广; 最后对相关的研究工作进行总结和展望.

关键词 深度强化学习, 异策略, 经验回放, 人工智能

引用格式 胡子剑, 高晓光, 万开方, 张乐天, 汪强龙, NERETIN Evgeny. 异策略深度强化学习中的经验回放研究综述. 自动化学报, 2023, **49**(11): 2237−2256

DOI 10.16383/j.aas.c220648

Research on Experience Replay of Off-policy Deep Reinforcement Learning: A Review

HU Zi-Jian[1] GAO Xiao-Guang[1] WAN Kai-Fang[1] ZHANG Le-Tian[2] WANG Qiang-Long[1] NERETIN Evgeny[3]

Abstract As a machine learning method that does not need to obtain training data in advance, reinforcement learning (RL) is an important method to solve the sequential decision-making problem by finding the optimal strategy in the continuous interaction between the agent and the environment. Through the combination of deep learning (DL), deep reinforcement learning (DRL) has both powerful perception and decision-making capabilities, and is widely used in many fields to solve complex decision-making problems. Off-policy reinforcement learning separates exploration and utilization by storing and replaying interactive experience, making it easier to find the global optimal solution. How to make reasonable and efficient use of experience is the key to improve the efficiency of off-policy reinforcement learning methods. First, this paper introduces the basic theory of reinforcement learning. Then, the on-policy and off-policy reinforcement learning algorithms are briefly introduced. Next, two mainstream solutions of experience replay (ER) problem are introduced, including experience utilization and experience expansion. Finally, the relevant research work is summarized and prospected.

Key words Deep reinforcement learning (DRL), off-policy, experience replay (ER), artificial intelligence

Citation Hu Zi-Jian, Gao Xiao-Guang, Wan Kai-Fang, Zhang Le-Tian, Wang Qiang-Long, Neretin Evgeny. Research on experience replay of off-policy deep reinforcement learning: A review. *Acta Automatica Sinica*, 2023, **49**(11): 2237−2256

强化学习 (Reinforcement learning, RL) 的来源通常被认为是心理学中的行为主义理论, 即有机体能获得最大利益的习惯性行为是在环境给予的奖励或惩罚的不断刺激下, 逐步形成的对刺激的预期. 直到 20 世纪末, RL 才开始得到研究者们的重视并迅速发展, 并被认为是设计智能体的核心技术之一[1−2].

RL 通过 “试错” (Trail-and-error)[2] 的方式与环境进行交互并获得奖励, 并依据奖励不断调整智能体的行为策略. 这种符合人类的经验性思维与直

收稿日期 2022-08-18 录用日期 2023-01-21
Manuscript received August 18, 2022; accepted January 21, 2023
国家自然科学基金 (62003267, 61573285), 中央高校基本科研业务费专项资金 (G2022KY0602), 电磁空间作战与应用重点实验室 (2022ZX0090), 西安市科技计划项目 —— 关键核心技术攻关工程项目计划 (21RGZN0016), 陕西省重点研发计划项目 (2023-GHZD-33) 资助
Supported by National Natural Science Foundation of China (62003267, 61573285), the Fundamental Research Funds for the Central Universities (G2022KY0602), the Technology on Electromagnetic Space Operations and Applications Laboratory (2022ZX0090), the Key Core Technology Research Plan of Xi′an (21RGZN0016), and the Key Research and Development Program of Shaanxi Province (2023-GHZD-33)
本文责任编委 张敏灵
Recommended by Associate Editor ZHANG Min-Ling
1. 西北工业大学电子信息学院 西安 710129 中国 2. 西安电子科技大学外国语学院 西安 710126 中国 3. 莫斯科航空学院机器人与智能系统学院 莫斯科 125993 俄罗斯
1. School of Electronics and Information, Northwestern Polytechnical University, Xi′an 710129, China 2. School of Foreign Languages, Xidian University, Xi′an 710126, China 3. School of Robotic and Intelligent Systems, Moscow Aviation Institute (National Research University), Moscow 125993, Russia

觉推理的一般决策过程使得其在人工智能领域得到了广泛的应用[3]. 随着应用环境复杂程度的不断提升, “维度灾难”[4] 限制了 RL 的进一步发展. 为了更好地表征复杂任务场景中高维度的状态空间, 谷歌人工智能团队 Deepmind 创新性地将深度学习 (Deep learning, DL) 与 RL 相结合, 提出了人工智能领域的一个新的研究热点 —— 深度强化学习 (Deep reinforcement learning, DRL)[5]. DRL 同时具备了 DL 的特征感知能力和 RL 的决策能力, 能够学习大规模输入数据的抽象表征, 并以此表征为依据进行自我激励, 优化解决问题的策略[6]. 目前, DRL 这种端对端 (End-to-end) 的学习方式已经在游戏博弈[5, 7–9]、机器人控制[10–12]、自动驾驶[13–15]、金融贸易[16–18]、医疗保健[19–20] 等多个领域取得了显著的进展, 其训练的智能体的表现已经接近甚至超越了人类水平.

不同于监督学习和无监督学习, RL 通过智能体与环境的不断交互来对环境进行探索进而获得经验 (样本), 并根据所获得的经验对智能体的策略不断更新, 最终找到一个适应环境的最优策略. 由于 RL 在学习过程中没有固定的数据集, 其需要智能体消耗大量的时间成本来获取交互经验. 在一些复杂的环境尤其是现实环境中 (例如自动驾驶) 会承担很多的风险与代价. 除此之外, 损耗、响应时延等问题也会使得智能体能够收集的经验数量是有限的. 如何合理利用有限的经验来训练出策略尽可能好的智能体已然成为国内外研究者的一个关注重点.

经验回放 (Experience replay, ER) 是一种存储过去的连续经验并对其进行采样以重复使用进而更新智能体行动策略的技术, 其概念于 1992 年被 Lin 等[21] 率先提出. 2015 年, 随着深度 Q 网络算法 (Deep Q-network, DQN)[5] 的提出, 经验回放被证明在 DRL 的突破性成功中发挥了重要的作用. 这一新的研究热点迅速吸引了大量研究者的关注, 到目前为止, 经验回放已成为提高异策略 DRL 算法稳定性和收敛速度的一种主要技术. 在现有文献中, 还没有研究尝试将 DRL 中的经验回放算法进行分类和总结. 本综述以 RL 的基本理论为出发点, 首先介绍了 RL 的基本概念. 随后对 RL 算法依据行为策略与目标策略的一致性进行了分类, 并对其中异策略 DRL 的典型算法进行了介绍. 然后结合近年来公开文献详细梳理了国内外成熟的异策略 DRL 中的经验回放方法, 并将其分为两个大类, 即经验利用和经验增广. 最后, 对异策略 DRL 中的经验回放方法进行了总结与展望.

1 深度强化学习理论基础

1.1 强化学习

RL 是一个学习如何将环境状态映射到智能体的行为以最大化累积回报的过程[2]. 其中智能体与环境的交互过程如图 1 所示, 在每一个时间步, 智能体首先对环境进行观测, 随后根据观测结果依照自身策略选择要采取的动作并执行. 该动作会使得环境的状态发生转移, 并且环境会根据这一动作的优劣对智能体进行奖励或惩罚. 智能体不断重复这个过程, 直至达到设定的终止状态, 结束这一回合的迭代.

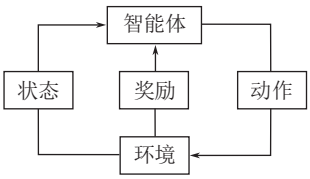

图 1 强化学习过程

Fig. 1 The process of reinforcement learning

马尔科夫决策过程 (Markov decision process, MDP) 为 RL 提供了一个简单易行的框架, 几乎所有的 RL 问题都可以被建模成 MDP. MDP 通常由四元组 $(\boldsymbol{S}, \boldsymbol{A}, P, R)$ 表示, 其中: 1) $\boldsymbol{S}$ 是环境中智能体所有能够到达的状态的集合; 2) $\boldsymbol{A}$ 代表智能体在环境中所有能够选择的动作的集合; 3) P 是智能体在状态 $\boldsymbol{s}$ 下执行动作 $\boldsymbol{a}$ 并到达状态 $\boldsymbol{s}'$ 的概率, 其中 $\boldsymbol{a} \in \boldsymbol{A}$ 并且 $\boldsymbol{s}, \boldsymbol{s}' \in \boldsymbol{S}$; 4) R 代表智能体在状态 $\boldsymbol{s}$ 下执行动作 $\boldsymbol{a}$ 所获得的奖励.

在 RL 中, 策略 π 是从状态空间到动作空间的映射: $\pi : \boldsymbol{S} \to \boldsymbol{A}$. π 中的一个元素 $\pi(\boldsymbol{a} \mid \boldsymbol{s})$ 代表智能体在状态 $\boldsymbol{s}$ 下选择动作 $\boldsymbol{a}$ 的概率. 依据该策略, 智能体能够获得累计奖励 R_t. 对于一个在 T 时间步终止的回合, 在任意时间步 t 时的累计奖励 R_t 的定义如下

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r(\boldsymbol{s}_{t'}, \boldsymbol{a}_{t'}) \tag{1}$$

其中, $r(\cdot)$ 是奖励函数, $\gamma \in [0, 1]$ 是一个折扣系数, 来决定未来奖励对累计奖励的影响, γ 的引入使得距离当前状态越远的奖励, 对当前的累计奖励的影响越小.

定义智能体在状态 $\boldsymbol{s}$ 下执行动作 $\boldsymbol{a}$ 并遵循策略 π 一直到回合结束所获得的累计奖励的数学期望为状态–动作值函数 $Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \mathrm{E}(R_t \mid \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}, \pi) \tag{2}$$

当遵循最优策略 π^* 时, 状态–动作值函数达到最大值

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = \max_{\pi} \mathrm{E}(R_t \mid \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}, \pi) \tag{3}$$

最优状态–动作值函数 $Q^*(\boldsymbol{s}, \boldsymbol{a})$ 满足具有递归属性的贝尔曼方程[22]

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = \mathrm{E}(r_t + \gamma \max_{\boldsymbol{a}_{t+1}} Q^{\pi}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}) \mid \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}) \tag{4}$$

持续地迭代公式 (4) 使状态–动作值函数最终收敛即可获得解决该 RL 问题的最优策略

$$\pi^* = \arg\max_{\boldsymbol{a} \in \boldsymbol{A}} Q^*(\boldsymbol{s}, \boldsymbol{a}) \tag{5}$$

1.2　强化学习算法

1.2.1　强化学习算法分类

RL 中通常包含两种策略, 分别称为行为策略 (Behavior policy) 和目标策略 (Target policy). 行为策略是智能体在与环境交互过程中用来选择动作的策略, 即在智能体训练过程中使用的策略. 而目标策略是指智能体在行为策略产生的经验中不断学习、优化时所采用的动作选择策略.

如图 2 所示, 根据算法中的行为策略和目标策略是否相同可以将 RL 算法分为两类, 同策略强化学习 (On-policy RL) 和异策略强化学习 (Off-policy RL).

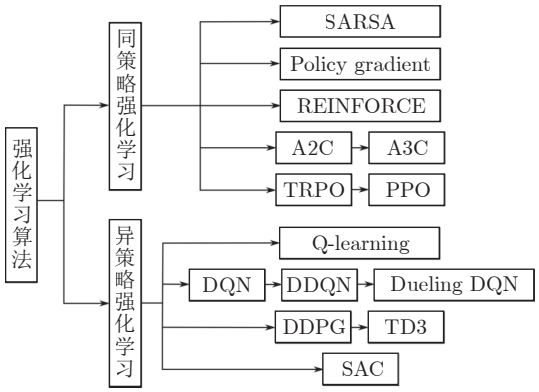

图 2　强化学习算法分类

Fig. 2　The classification of reinforcement learning algorithms

属于同策略 RL 的算法主要有 SARSA[23]、Policy gradient[24]、REINFORCE[25]、基于演员–评论家 (Actor-critic, AC) 架构的优势演员–评论家 (Advantage AC, A2C)[26]、异步优势演员–评论家 (Asynchronous A2C, A3C)[27]、信赖域策略优化 (Trust region policy optimization, TRPO)[28]、近端策略优化 (Proximal policy optimization, PPO)[29] 等算法. 异策略 RL 主要包含 Q-learning[30]、DQN[5] 及其一些改进算法[31−32]、确定策略梯度的深度确定性策略梯度 (Deep deterministic policy gradient, DDPG)[33] 和改进后的双延迟深度确定性策略梯度 (Twin delayed DDPG, TD3)[34]、随机策略梯度的柔性演员–评论家 (Soft AC, SAC)[35] 等算法.

以同策略 RL 和异策略 RL 的经典算法 SARSA 和 Q-learning 为例, 其状态–动作值函数的更新方式分别如下式所示

$$Q(\boldsymbol{s}, \boldsymbol{a}) \leftarrow Q(\boldsymbol{s}, \boldsymbol{a}) + \alpha\left(r + \gamma Q(\boldsymbol{s}', \boldsymbol{a}') - Q(\boldsymbol{s}, \boldsymbol{a})\right) \tag{6}$$

$$Q(\boldsymbol{s}, \boldsymbol{a}) \leftarrow Q(\boldsymbol{s}, \boldsymbol{a}) + \alpha\left(r + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}', \boldsymbol{a}') - Q(\boldsymbol{s}, \boldsymbol{a})\right) \tag{7}$$

二者的区别在于, 在 Q-learning 中, 行为策略采用 ε-greedy 策略进行动作选择来完成智能体与环境的交互过程, 在 Q 值更新时, 目标策略并不关心下一时间步所采取的动作 $\boldsymbol{a}'$, 而是贪婪地使用下一时间步中最大的 Q 值来更新当前时间步的 Q 值. 而在 SARSA 算法中, 智能体交互过程和 Q 值更新过程中的动作选择策略均采用的是 ε-greedy 策略.

同策略 RL 通常在每一时间步都对目标策略进行实时更新, 而异策略 RL 通常设计经验池来对行为策略产生的交互经验进行存储, 以便智能体对其采样和学习, 从而实现对目标策略的不断更新. 二者各自的主要优势如表 1 所示.

表 1　同策略与异策略算法的优势对比

Table 1　Comparison of advantages of on-policy and off-policy algorithms

算法优势	同策略 RL	异策略 RL
收敛速度更快	√	
训练过程更稳定	√	
超参数对算法影响更小	√	
可以平衡探索和利用的问题		√
更易收敛到最优解		√
经验来源更广		√
经验的利用率更高		√
算法的适用范围更广		√

1.2.2　异策略强化学习

为了扩大 RL 算法的使用范围, Mnih 等[5] 将卷积神经网络与 Q-learning 相结合, 提出了基于值函数的 DQN 算法. DQN 主要有以下两个特点:

1) 使用两个独立的网络

如图 3 所示, DQN 有两个超参数相同的深度

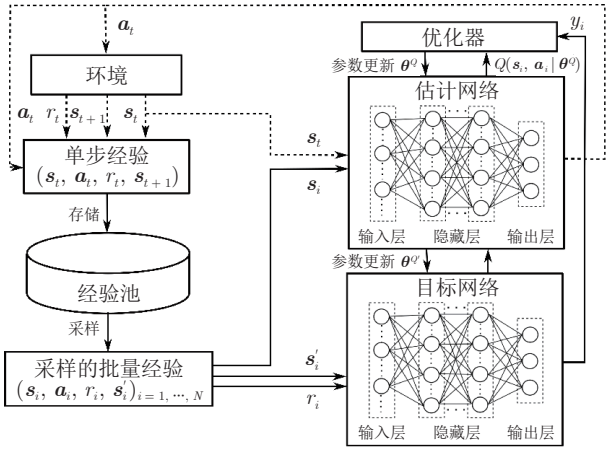

图 3 DQN 算法框架

Fig. 3 The framework of DQN algorithm

神经网络, 分别叫做估计网络 (Eval net) 和目标网络 (Target net). 参数为 $\boldsymbol{\theta}^Q$ 的估计网络的输出为估计 Q 值 $Q(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}^Q)$, 用来近似表示状态–动作值函数, 而参数为 $\boldsymbol{\theta}^{Q'}$ 的目标网络的输出则为 $Q(\boldsymbol{s}'_i, \boldsymbol{a}'_i \mid \boldsymbol{\theta}^{Q'})$, 通过结合奖励值 r_i 来计算目标 Q 值 y_i

$$y_i^{\text{DQN}} = r_i + \gamma \max_{\boldsymbol{a}'_i} Q(\boldsymbol{s}'_i, \boldsymbol{a}'_i \mid \boldsymbol{\theta}^{Q'}) \tag{8}$$

DQN 通过最小化估计 Q 值与目标 Q 值之间的均方误差来更新估计网络的参数

$$L(\boldsymbol{\theta}^Q) = \frac{1}{N}\sum_i \left(y_i^{\text{DQN}} - Q(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}^Q)\right)^2 \tag{9}$$

其中, N 为采样的经验数.

每隔一定的迭代次数, 就将估计网络的参数 $\boldsymbol{\theta}^Q$ 直接复制到目标网络. DQN 的这种网络结构降低了目标网络与估计网络之间的相关性, 从而提升了算法的稳定性.

2) 引入经验回放机制

为了消除经验之间的相关性, 使用于训练的经验满足独立同分布, DQN 首次引入经验回放机制并设计了一个经验池来存储和管理经验. 在每一个时间步, 将智能体与环境的交互经验 $(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}_{i+1})$ 暂存到经验池中. 在训练过程中采取随机抽样的方法从经验池批量选择经验来对估计网络的参数进行更新, 从而实现了对过往经验的充分利用, 进一步提升了算法的性能.

然而由于神经网络估计的 Q 值会在某些时候产生正向或负向的误差, 而根据 DQN 的更新方式 (式 (8)), 这些误差会不断地向正向累积从而产生状态–动作值函数过高估计的问题. 为了解决这个问题, Hasselt 等[31] 提出了深度双 Q 网络 (Double DQN, DDQN). 不同于 DQN 只使用目标网络来计算目标 Q 值, DDQN 将动作选择与策略评估分离开来, 首先使用估计网络选择最优的动作, 随后使用目标网络对该动作进行评估. DDQN 的网络结构和参数更新方式均与 DQN 完全相同, 而其目标 Q 值的计算方式有所改变

$$y_i^{\text{DDQN}} = r_i + \gamma Q(\boldsymbol{s}'_i, \arg\max_{\boldsymbol{a}} Q(\boldsymbol{s}'_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}^Q) \mid \boldsymbol{\theta}^{Q'}) \tag{10}$$

这种通过利用两个网络来计算目标 Q 值的方式使得即便其中某一网络的某个动作存在严重的过估计, 而由于另一网络的存在, 该动作最终使用的 Q 值也不会被过高估计. 实验表明, DDQN 能够对 Q 值进行更为准确的估计, 在多种应用场景中都可以获得更为稳定有效的策略.

在很多场景中, Q 值只受当前状态影响, 智能体所采取的动作对其影响不大. 基于这种现象, Wang 等[32] 提出了竞争深度 Q 网络 (Dueling DQN). 与 DQN 不同的是, 在 Dueling DQN 中经过卷积神经网络处理的特征被分别输入到两个不同的全连接网络中, 即值函数网络 $V(\boldsymbol{s}_i \mid \boldsymbol{\theta}, \boldsymbol{\beta})$ 和优势函数网络 $A(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}, \boldsymbol{\alpha})$, 其中 $\boldsymbol{\theta}$ 是共用的卷积神经网络部分的参数, $\boldsymbol{\beta}$ 和 $\boldsymbol{\alpha}$ 分别为值函数网络和优势函数网络对应全连接层的参数. 将两个网络的输出进行合并得到

$$\begin{aligned} &Q(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \\ &\quad A(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}, \boldsymbol{\alpha}) + V(\boldsymbol{s}_i \mid \boldsymbol{\theta}, \boldsymbol{\beta}) \end{aligned} \tag{11}$$

然而按以上优势函数构造的 Q 函数会导致解不唯一的问题, 在实际使用时, 一般通过将动作优势函数值减去当前状态下所有优势函数的平均值来提高训练过程的稳定性

$$\begin{aligned} &Q(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \\ &\quad A(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}, \boldsymbol{\alpha}) - \frac{1}{|A|}\sum_{\boldsymbol{a}} A(\boldsymbol{s}_i, \boldsymbol{a} \mid \boldsymbol{\theta}, \boldsymbol{\alpha}) + \\ &\quad V(\boldsymbol{s}_i \mid \boldsymbol{\theta}, \boldsymbol{\beta}) \end{aligned} \tag{12}$$

除了上述较为经典的 DDQN、Dueling DQN 算法, 还有很多研究尝试对 DQN 从训练算法、网络结构和学习机制等不同方面进行改进, 例如分布式 DQN (Distributional DQN)[36]、深度循环 Q 网络 (Deep recurrent Q-network, DRQN)[37]、噪声 DQN (Noisy DQN)[38]、Rainbow[39] 等. 刘建伟等[40] 对这些 DQN 的改进算法进行了详细的分析和讨论.

DQN 一类的基于值函数的算法通过对状态–动作值函数的近似表达来进行学习, 并且在处理离散动作空间问题时取得了不错的效果. 然而当问题扩展到连续动作空间时, 贪婪策略需要在每一个时间步进行优化, 这种优化的速度太慢且无法应用于

大型无约束的函数优化器[33]. 而基于策略梯度的算法直接将策略近似, 因此可以很好地在连续空间中对动作进行搜索. 基于策略梯度的算法通常分为两种: 输出动作为状态映射 $\boldsymbol{a}=\mu_{\boldsymbol{\theta}}(\boldsymbol{s})$ 的确定策略梯度算法和输出动作为概率分布 $\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{s})$ 的随机策略梯度算法. 二者的策略梯度分别为

$$\nabla_{\boldsymbol{\theta}}J(\mu_{\boldsymbol{\theta}})=\mathrm{E}_{\boldsymbol{s}\sim\rho^{\mu}}\left(\nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}Q^{\mu}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\mu_{\boldsymbol{\theta}}(\boldsymbol{s})}\right) \tag{13}$$

$$\nabla_{\boldsymbol{\theta}}J(\pi_{\boldsymbol{\theta}})=\mathrm{E}_{\boldsymbol{s}\sim\rho^{\pi},\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}}\left(\nabla_{\boldsymbol{\theta}}\ln(\pi_{\boldsymbol{\theta}}(\boldsymbol{s},\boldsymbol{a}))Q^{\pi}(\boldsymbol{s},\boldsymbol{a})\right) \tag{14}$$

其中, ρ^{π} 和 ρ^{μ} 是状态的采样空间, 而 $\pi_{\boldsymbol{\theta}}$ 是动作的采样空间.

Lillicrap 等[33] 提出了一种基于演员–评论家架构的 DRL 方法 DDPG, 该方法在求解具有连续动作空间的 MDP 时取得了良好的效果. 演员网络用于确定智能体选择动作的概率, 而评论家网络用于根据环境状态对智能体选择的动作进行评估. 如图 4 所示, 与 DQN 的网络结构相同, DDPG 的演员和评论家网络都包含两个结构相同的估计网络和目标网络.

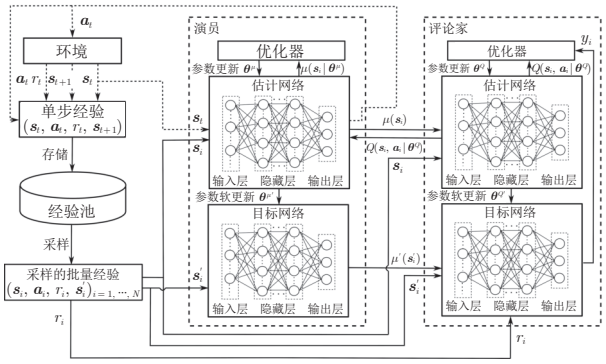

图 4　DDPG 算法框架

Fig. 4　The framework of DDPG algorithm

DDPG 加入了一个独立噪声 $\mathcal{N}_t$ 来增加智能体探索过程的随机性

$$\boldsymbol{a}_t=\mu(\boldsymbol{s}_t\mid\boldsymbol{\theta}^{\mu})+\mathcal{N}_t \tag{15}$$

演员网络使用策略梯度 $\nabla_{\boldsymbol{\theta}^{\mu}}J$ 来近似其估计网络的参数, 而评论家网络通过最小化损失函数 $L\left(\boldsymbol{\theta}^{Q}\right)$ 来更新其估计网络的参数

$$\nabla_{\boldsymbol{\theta}^{\mu}}J\approx\frac{1}{N}\sum_{i}\left(\nabla_{\boldsymbol{a}}Q(\boldsymbol{s},\boldsymbol{a}\mid\boldsymbol{\theta}^{Q})\Big|_{\boldsymbol{s}=\boldsymbol{s}_i,\boldsymbol{a}=\mu(\boldsymbol{s}_i)}\times\nabla_{\boldsymbol{\theta}^{\mu}}\mu(\boldsymbol{s}\mid\boldsymbol{\theta}^{\mu})\Big|_{\boldsymbol{s}=\boldsymbol{s}_i}\right) \tag{16}$$

$$L\left(\boldsymbol{\theta}^{Q}\right)=\frac{1}{N}\sum_{i}\left(y_i-Q(\boldsymbol{s}_i,\boldsymbol{a}_i\mid\boldsymbol{\theta}^{Q})\right)^2 \tag{17}$$

$$y_i=r(\boldsymbol{s}_i,\boldsymbol{a}_i)+\gamma Q'\left(\boldsymbol{s}_i',\mu'(\boldsymbol{s}_i'\mid\boldsymbol{\theta}^{\mu'})\mid\boldsymbol{\theta}^{Q'}\right) \tag{18}$$

在 DDPG 的基础上, Fujimoto 等[34] 提出了一种新的基于策略的算法 TD3, 该算法已被证明是目前最先进的 DRL 算法之一. TD3 算法主要做了以下三个改进来提升 DDPG 算法的性能.

1) 利用双网络结构来避免过估计: TD3 具有两套评论家网络来分别计算 $Q_j(\boldsymbol{s}_i,\boldsymbol{a}_i\mid\boldsymbol{\theta}^{Q_j})$, 并选择其中较小的作为目标, 因此式 (17)、式 (18) 分别为

$$L(\boldsymbol{\theta}^{Q_j})=\min_{j=1,2}\frac{1}{N}\sum_{i}\left(y_i-Q_j(\boldsymbol{s}_i,\boldsymbol{a}_i\mid\boldsymbol{\theta}^{Q_j})\right)^2 \tag{19}$$

$$y_i=r(\boldsymbol{s}_i,\boldsymbol{a}_i)+\gamma\min_{j=1,2}Q_j'(\boldsymbol{s}_i',\mu'(\boldsymbol{s}_i'\mid\boldsymbol{\theta}^{\mu'})\mid\boldsymbol{\theta}^{Q_j'}) \tag{20}$$

2) 延迟更新演员网络来增加稳定性: 不同于 DDPG 算法的同步更新演员和评论家网络参数, TD3 算法让评论家网络的更新频率稍高于演员网络, 以此来提高演员网络的稳定性.

3) 添加动作噪声来平滑目标策略: 为了使学习到的策略更加平滑和稳定, TD3 加入了一个动作噪声 $\varepsilon\sim\mathrm{clip}(\mathrm{N}(0,\tilde{\sigma}),-l,l)$ 来使得在计算 Q 值时动作能够在一定范围内随机变化

$$y_i=r(\boldsymbol{s}_i,\boldsymbol{a}_i)+\gamma\min_{j=1,2}Q_j'(\boldsymbol{s}_i',\mu'(\boldsymbol{s}_i'\mid\boldsymbol{\theta}^{\mu'})+\varepsilon\mid\boldsymbol{\theta}^{Q_j'}) \tag{21}$$

TD3 的网络参数的更新方式则与 DDPG 保持一致, 采用软更新的方法

$$\boldsymbol{\theta}'\leftarrow\tau\boldsymbol{\theta}+(1-\tau)\boldsymbol{\theta}' \tag{22}$$

其中, $\tau\in[0,1]$ 决定了每次更新的幅度.

不同于确定策略梯度, 随机策略梯度能够使智能体在相同的状态下按照概率分布选取不同的动作. 最为广泛使用的异策略 RL 算法就是引入了熵 (Entropy) 的概念的 SAC 算法.

熵是对一个随机变量的随机程度大小的度量. 对于一个随机变量 X, 假设其概率密度为 p, 即 $p(x_i)$ 是随机变量 X 为 x_i 的概率, 那么它的熵就被定义为

$$H(X)=\mathrm{E}_{x_i\sim p}(-\log_2 p(x_i)) \tag{23}$$

在 RL 中常用 $H(\pi(\cdot\mid\boldsymbol{s}))$ 来表示策略 π 在状态 $\boldsymbol{s}$ 下的随机程度. 最大熵强化学习 (Maximum entropy RL), 就是在 RL 的目标中加入熵的正则项来最大化累计奖励, 同时使策略更加随机

$$\pi^{*}=\arg\max_{\pi}\mathrm{E}_{\pi}(R_t+\alpha H(\pi(\cdot\mid\boldsymbol{s}))) \tag{24}$$

其中, α 是控制熵的重要程度的正则化系数.

相较于传统的 RL, 熵的正则化增加了最大熵

RL 算法的探索程度, α 越大, 算法的探索性就越强, 策略学习的速度也就越快, 陷入局部最优解的可能性也就越小.

SAC 算法由 Haarnoja 等[35] 在 2018 年首次提出, 使用随机策略来实现连续控制, 随后他们又在文献 [41] 中基于 Q-learning 舍弃了值函数的应用, 还将熵的权重 α 设计为可自动调整的参数来提高训练的稳定性. 与 TD3 算法相同, SAC 算法也使用了双评论家的网络结构 (包含参数为 $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$ 的估计网络和对应参数为 $\boldsymbol{\theta}_1'$, $\boldsymbol{\theta}_2'$ 的目标网络). 与 TD3 算法不同的是, 由于只包含一个参数为 $\boldsymbol{\theta}$ 的演员网络, SAC 算法在计算 Q 值时仅通过演员网络 $\pi_{\boldsymbol{\theta}}$ 来根据状态进行动作选择 $\boldsymbol{a}_i \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{s}_i)$. 因此, 式 (21) 变为

$$y_i = r_i + \gamma \left(\min_{j=1,2} Q_j'(\boldsymbol{s}_i', \boldsymbol{a}_i') - \alpha \log_2 \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_i' \mid \boldsymbol{s}_i') \right) \tag{25}$$

评论家网络的估计网络则通过最小化损失函数 $L\left(\boldsymbol{\theta}^{Q_j}\right)$ 来进行更新

$$L\left(\boldsymbol{\theta}^{Q_j}\right) = \min_{j=1,2} \frac{1}{N} \sum_i \left(y_i - Q_j(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}^{Q_j})\right)^2 \tag{26}$$

由于在连续动作空间的环境中, SAC 算法演员网络输出的动作是对高斯分布采样得到的, 需要使用重参数化技巧 (Reparameterization trick)[42] 来使得动作采样的过程可导, 从而方便策略梯度的计算

$$\tilde{\boldsymbol{a}}_{\boldsymbol{\theta}}(\boldsymbol{s}, \xi) = \tanh(\mu_{\boldsymbol{\theta}}(\boldsymbol{s}) + \sigma_{\boldsymbol{\theta}}(\boldsymbol{s}) \odot \xi),\ \xi \sim \mathrm{N}(0, 1) \tag{27}$$

其中, $\mu_{\boldsymbol{\theta}}(\boldsymbol{s})$ 和 $\sigma_{\boldsymbol{\theta}}(\boldsymbol{s})$ 分别为在状态 $\boldsymbol{s}$ 下多次选择动作的均值和标准差. 随后使用损失函数 $L_\pi(\boldsymbol{\theta})$ 来对演员网络进行更新

$$L_\pi(\boldsymbol{\theta}) = \frac{1}{N} \sum_i \left(\alpha \log_2 \pi_{\boldsymbol{\theta}}(\tilde{\boldsymbol{a}}_i \mid \boldsymbol{s}_i) - \min_{j=1,2} Q_j(\boldsymbol{s}_i, \tilde{\boldsymbol{a}}_i) \right) \tag{28}$$

对于评论家网络的目标网络, SAC 也同样使用软更新的方式来更新其参数.

2 经验回放机制

如图 5 中的外部循环所示, 在异策略 RL 算法中, 经验回放通常是通过设计一个经验池来对智能体与环境的交互经验进行暂存, 以便智能体从中选择合适的经验来学习更新自身的行动策略. 经验池通常是一个固定大小的先进先出 (First in first out, FIFO) 的缓冲区, 其中包含智能体收集的最新的部分经验. 这种缓冲区为 DRL 带来了两个优点: 1) 均匀采样打破了连续经验之间的相关性, 提高了算法

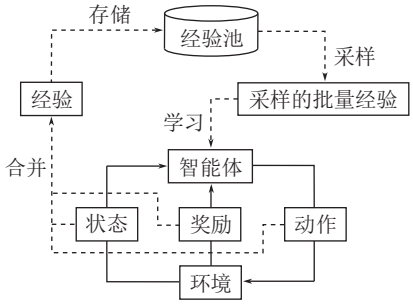

图 5 异策略 RL 的经验回放流程

Fig. 5 The experience replay process of off-policy RL

的稳定性; 2) 大容量缓冲区确保了从长期经验中学习的可能性, 从而避免了 "灾难性遗忘"[5] 现象的发生.

作为一种独立于 RL 循环之外的即插即用的模块, 经验回放已经被广泛应用于各种领域来提升异策略 RL 算法的效果. 如图 6 所示, 本节将经验回放算法分为经验利用和经验增广两大类, 来对目前较为成熟的经验回放领域的相关研究进行详细介绍.

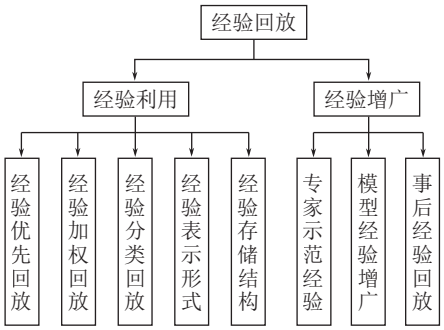

图 6 经验回放分类

Fig. 6 The classification of experience replay

2.1 经验利用

2.1.1 经验优先回放

经验利用的最基本思路就是调整被学习的经验的顺序, 即通过设计一些采样算法来优先选择更适合当前网络收敛的经验进行学习.

2016 年, Schaul 等[43] 首次提出了优先经验回放算法 (Prioritized experience replay, PER), 根据所存储经验各自的时间差分误差 (Temporal difference error, TD error) 的大小来赋予其不同的优先级, 使得 TD error 更大的经验能够获得更大的采样概率. 对于一条经验 $\boldsymbol{e}_i$, 其采样概率计算如下

$$P(\boldsymbol{e}_i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha} \tag{29}$$

$$p_i = |\delta_i| + \varepsilon \tag{30}$$

其中, δ_i 为 TD error, ε 是一个正向的常数, 来保证 TD error 接近零的经验也有被采样到的可能. 除此之外, PER 还设计了一个名为 "sum tree" 的数据结构来保证经验存储和优先性采样的高效性. 实验表明, 结合了 PER 后的 DQN 算法在收敛效率上取得大幅提高, 在 49 个 Atari 游戏中有 41 个的表现都要优于原始的 DQN 算法.

受 PER 算法的启发, Brittain 等[44] 考虑了连续经验之间的关系, 他们认为不仅应该为重要的经验分配更高的采样优先级, 也应该增加导致重要经验产生的前序经验的优先级, 并提出了一种优先序列经验回放算法 (Prioritized sequence experience replay, PSER). PSER 通过引入一个衰减因子 ρ 来依据当前经验优先级 p_n 对前序经验的优先级进行更新

$$p_{n-m} = \max\left\{p_n \rho^m,\, p_{n-m}\right\} \tag{31}$$

其中, m 为前序经验的远近程度. 在 60 款 Atari 游戏的实验测试中, PSER 在其中的 40 款都取得了优于 PER 的收敛效果.

Lee 等[45] 认为 PER 存在包含大量有偏向性的优先化问题. 针对这一问题, 他们设计了三种不同的策略 (TDInit、TDClip 和 TDPred), 并将它们综合在一起提出了预测优先经验回放算法 (Predictive PER, PPER). PPER 在实验测试中不仅消除了优先级的异常值的存在, 还改善了经验池中经验的分布, 平衡了经验池中经验优先性和多样性, 使得算法的稳定性得到了大幅改善和提升.

Cao 等[46] 提出了高价值优先经验回放方法 (High-value PER, HVPER), 将经验对应的状态–动作函数值和 TD error 一同作为衡量其优先级的指标, 并将优先级函数定义为

$$p(i) = \left(\lambda p_{\mathrm{Q}}(i) + (1-\lambda) p_{\mathrm{TD}}(i)\right) u_i \tag{32}$$

其中, p_{Q} 和 p_{TD} 分别为归一化后的 Q 值和 TD error, $u_i = u_0 \mu^{n_i}$ 表示学习次数 n_i 对经验优先级的影响. HVPER 在多种环境的测试实验以及与多种异策略 RL 算法的结合中都取得了不错的效果. 如果能够设计一种基于经验分布在线自适应地调整超参数 λ 的方法, 则能有效提升 HVPER 的适用性.

赵英男等[47] 提出一种二次主动采样算法 (Twice active sampling method, TASM), 来实现经验的分批次优先利用. TASM 算法中的经验是按照序列进行存储的, 在采样时首先以单个序列的累积奖励作为标准筛, 从大经验池中选出多个符合条件的序列构成小经验池, 第二轮再回归 PER 算法以 TD error 为标准从小经验池中选择经验进行学习. TASM 的二次采样有效地实现了经验的高效利用并提升了策略的质量, 但由于其在采样时要对所有序列进行遍历, 算法的时间复杂度稍有提升.

Sun 等[48] 则考虑了过去存储的经验中的状态与智能体当前状态的相似性, 提出了注意力经验回放算法 (Attentive experience replay, AER), 实现了比 PER 更高的经验利用率和算法收敛速度. 如下式所示, 针对向量形式状态的环境和图片形式状态的环境, AER 分别设计了不同的相似性函数来计算状态之间的相似性

$$\mathcal{F}(\boldsymbol{s}_1, \boldsymbol{s}_2) = \frac{\boldsymbol{s}_1 \cdot \boldsymbol{s}_2}{\|\boldsymbol{s}_1\| \, \|\boldsymbol{s}_2\|} \tag{33}$$

$$\mathcal{F}(\boldsymbol{s}_1, \boldsymbol{s}_2) = -\left\|\phi(\boldsymbol{s}_1) - \phi(\boldsymbol{s}_2)\right\|_2 \tag{34}$$

其中, ϕ 是一个结构固定而初始值随机的深度卷积神经网络. 除此之外, AER 还采用多轮次采样的方法来调整优先采样的程度以提高算法的适应性. 然而, AER 在采样时依次计算采样到的大批量经验与当前状态之间的相似性并进行排序, 会耗费大量的时间, 并且相似状态的重复出现导致大量的重复计算, 也对其运行效率产生了不利影响.

Hu 等[49] 同样考虑了当前状态与过去经验之间的相似性, 并针对于无人机自主运动控制场景提出了相似经验学习算法 (Relevant experience learning, REL). 与 AER 不同, REL 使用了一个更有针对性的函数来衡量智能体的状态的价值, 并将该函数值一同存入经验池中, 在采样时直接根据该函数值的差异性来评价状态之间的相似性, 从而避免了 AER 会出现的重复计算现象. REL 还采用了 PER 中的 "sum tree" 结构来进行经验的存取, 大幅降低了寻找相似经验的时间复杂度. 除此之外, REL 还调整了 RL 中动作选择和策略更新的顺序, 使得每一次的策略更新都能及时地作用在当前时间步的动作选择上, 从而充分发挥了过去经验的作用, 加快了算法的收敛速度.

Cicek 等[50] 提出了一种基于 KL (Kullback-Leibler) 散度的批量优先经验回放 (Batch prioritized experience replay via KL divergence, KLPER), 将单个经验优先级排序扩展到批量经验优先级排序. 在每次采样时, KLPER 先采样出多个批量经验并通过 KL 散度对其优先级进行排序, 最终找到其中与智能体的最新策略最相近的批量经验进行学习. 在多种连续的控制任务中, KLPER 在样本效率和收敛性能上均优于随机采样的经验回放算法和 PER 算法. 然而所学习的批量经验不可避免地仍会包含一些价值较低的经验, 对这些经验的剔除或替换则有望进一步提升 KLPER 算法的性能.

为了使智能体能够模拟人类的由简到难的学习

过程, Ren 等 [51] 首次将课程学习 (Curriculum learning, CL)[52−53] 的机制引入到经验回放当中, 提出了深度课程强化学习 (Deep curriculum reinforcement learning, DCRL). DCRL 所定义的复杂性标准包括自定步长优先级和覆盖惩罚. 自定步长优先级反映了 TD error 与当前课程难度之间的关系

$$\mathrm{SP}(\delta,\lambda)=\begin{cases}\exp(|\delta|-\lambda), & |\delta|\le\lambda\\ \dfrac{1}{\ln(1-\lambda)}\ln(|\delta|-2\lambda+1), & \lambda<|\delta|<2\lambda\\ 0, & |\delta|\ge 2\lambda\end{cases}\tag{35}$$

其中, δ 是经验对应的 TD error, 而 λ 是随训练不断增大的课程因子. 覆盖惩罚则用于避免同一经验被多次重复学习

$$\mathrm{CP}(cn)=\exp\left(-\frac{cn^2}{10}\right)\tag{36}$$

其中, cn 是经验被学习的次数. DCRL 通过复杂度函数 $\mathrm{CI}(x_i)=\mathrm{SP}(\delta_i,\lambda)+\eta\mathrm{CP}(cn_i)$ 来自适应地从经验池中选择合适的经验, 从而充分利用了经验回放的优势, 提升了算法的收敛速度. 但由于引入了 CL 来控制所回放经验的难度, DCRL 额外加入了几个环境敏感性较高的超参数, 在实际使用时还需要依据环境特性进行调整.

上述的这些研究虽然用不同的方式在一定程度上弥补了 PER 算法的不足, 但 PER 所存在的根本性问题尚未得到良好的解决. Hu 等[54] 对 PER 进行了详细分析并总结了其 4 个有待改进的问题: 1) TD error 的更新速度太慢会影响被采样到的经验的价值; 2) PER 中的裁剪 (clip) 操作降低了不同经验之间的差异性; 3) 赋予新经验最大的优先级不能保证其优先性; 4) 仅根据 TD error 进行采样可能不是最优的采样方法.

他们尝试从根本上解决这些问题, 并提出了异步课程经验回放算法 (Asynchronous curriculum experience replay, ACER). ACER 算法主要有以下几个贡献: 1) 开启一个子线程来异步更新经验池中经验的优先级; 2) 废除 PER 中的 clip 操作来赋予经验其真实的优先级; 3) 设计了一个临时经验池来充分利用最新产生的经验; 4) 将 FIFO 的经验池进行了更替来使得经验池满时最无用的经验会优先被替换; 5) 引入 CL 使学习过程更为合理的同时解决无 clip 操作带来的问题. ACER 算法在收敛速度上比 PER 取得了较大的提升, 其稳定性也在多个不同的应用场景的测试中得到了验证.

表 2 对上述经验优先回放类算法进行了总结, 可以明显看到, 大多数经验优先回放类算法还是在尝试设计不同的评价指标来衡量经验的优先级. KLPER 则是将单个经验优先回放扩展到批量经验优先回放. DCRL 则将 CL 思想引入经验回放中, 为经验优先回放提供了一种新的思路. 在较为全面的分析和总结后, ACER 对 PER 的缺点逐一做出了改进, 但其所提供的理论支撑还不够完善, 且其作为弥补 PER 算法所有缺点的第一次尝试, 一些设计在合理性和计算效率上有待进一步完善和提高.

表 2 经验优先回放算法对比

Table 2 Comparison of prioritized experience replay algorithms

算法	优先回放指标	采样轮次
PER[43], PSER[44], PPER[45]	TD error	单轮
HVPER[46]	Q 值, TD error	单轮
TASM[47]	序列累计奖励, TD error	多轮
AER[48]	相似性	多轮
REL[49]	TD error, 相似性	多轮
KLPER[50]	批量经验策略的相似性	单轮
DCRL[51]	经验难度, 采样次数	单轮
ACER[54]	经验难度	单轮

2.1.2 经验加权回放

相较于按照特定标准从庞大的经验池中筛选经验的经验优先回放的方法, 近年来一个更为灵活且计算复杂度更低的研究方向是通过重加权的方式赋予更为重要的经验更高的权重, 从而利用经验回放在提高策略准确性的同时加速策略的收敛.

Kumar 等[55] 针对 Q-learning 和 AC 算法缺少纠正性反馈 (Corrective feedback) 而导致这些算法所存在的易收敛到次优解、学习过程不稳定、信噪比较高时学习效果差等问题, 提出了一种分布校正 (Distribution correction, DisCor) 的方法. DisCor 使用神经网络来估计 Q 值的累计误差 $\Delta_{\phi}(\boldsymbol{s},\boldsymbol{a})$, 并理论推导出了下式来计算所学习经验在第 k 次更新时的权重 w_k

$$w_k(\boldsymbol{s},\boldsymbol{a})\propto\exp\left(-\frac{\gamma\left[P^{\pi_{k-1}}\Delta_{k-1}\right](\boldsymbol{s},\boldsymbol{a})}{\tau}\right)\tag{37}$$

其中, $P^{\pi_{k-1}}$ 是第 $k-1$ 次更新后策略 π_{k-1} 的状态转移矩阵, τ 为常数. 在每次学习时, Q 值和误差网络分别按照下式进行参数更新

$$\boldsymbol{\theta}_{k+1}\leftarrow\arg\min_{\boldsymbol{\theta}}\frac{1}{N}\sum_{i=1}^{N}w_k(\boldsymbol{s}_i,\boldsymbol{a}_i)\left(Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i,\boldsymbol{a}_i)-y_i\right)^2\tag{38}$$

$$\boldsymbol{\phi}_{k+1}\leftarrow\arg\min_{\boldsymbol{\phi}}\frac{1}{N}\sum_{i=1}^{N}\left(\Delta_{\boldsymbol{\theta}}(\boldsymbol{s}_i,\boldsymbol{a}_i)-\hat{\Delta}_i\right)^2\tag{39}$$

其中

$$\hat{\Delta}_i = |Q_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{a}) - y_i| + \gamma \Delta_{k-1}(\boldsymbol{s}'_i, \hat{\boldsymbol{a}}_i) \tag{40}$$

$$\hat{\boldsymbol{a}}_i = \arg\max_{\boldsymbol{a}} Q_{k-1}(\boldsymbol{s}'_i, \boldsymbol{a}) \tag{41}$$

通过对不同经验的重加权来纠正 Q 值的累计误差, DisCor 在表格环境、连续控制环境甚至于多任务环境中都展现出了其策略收敛的高效性和稳定性.

受 DisCor 的启发, Lee 等[56] 提出了一种不同的思路并设计了基于集成学习的强化学习框架 (Simple unified framework for reinforcement learning using ensembles, SUNRISE). 与 A3C 类似, SUNRISE 集成了 N 对演员和评论家网络, 在使用采样的经验对每一个评论家网络进行更新时, 经验的权重设计为

$$w(\boldsymbol{s}, \boldsymbol{a}) = \sigma\left(-\bar{Q}_{\text{std}}(\boldsymbol{s}, \boldsymbol{a})T\right) + 0.5 \tag{42}$$

其中, $\bar{Q}_{\text{std}}(\boldsymbol{s}, \boldsymbol{a})$ 是 N 个评论家网络的目标 Q 值的标准差, $\sigma(\cdot)$ 是 Sigmoid 函数, T 为常数.

SUNRISE 显著地提高了 Q 值更新过程中的信噪比, 使学习过程更加稳定. 通过与 SAC 算法和 Rainbow 算法的结合, SUNRISE 在低维和高维环境下的连续和离散控制任务中均取得了优于最先进的 RL 算法的收敛效果.

Sinha 等[57] 则认为应该按照当前策略下的经验分布来设计经验在回放时权重的大小, 并提出了一种无似然重要性加权方法 (Likelihood-free importance weighting, LFIW), 优先回放那些出现频率较高的状态–动作对 $(\boldsymbol{s}, \boldsymbol{a})$. LFIW 通过设计大小两个经验池 $\mathcal{D}_{\text{s}}$ 和 $\mathcal{D}_{\text{f}}$ 来分别存储过去不同策略指导下的经验和最近策略指导下的经验. 除此之外, LFIW 使用了一个参数为 $\boldsymbol{\psi}$ 的神经网络来估计状态–动作对 $(\boldsymbol{s}, \boldsymbol{a})$ 的权重, 并通过损失函数 $L_w(\boldsymbol{\psi})$ 来进行参数更新

$$\begin{aligned} L_w(\boldsymbol{\psi}) := {} & \mathrm{E}_{\mathcal{D}_{\text{s}}}(f^*\left(f'\left(w_{\boldsymbol{\psi}}(\boldsymbol{s}, \boldsymbol{a})\right)\right)) - \\ & \mathrm{E}_{\mathcal{D}_{\text{f}}}(f'\left(w_{\boldsymbol{\psi}}(\boldsymbol{s}, \boldsymbol{a})\right)) \end{aligned} \tag{43}$$

其中, f 是一个满足 $f(1) = 0$ 的下半连续函数. LFIW 使用一个常数 T 对权重网络的输出进行归一化从而得到合理的经验概率化权重

$$\tilde{w}_{\boldsymbol{\psi}}(\boldsymbol{s}, \boldsymbol{a}) := \frac{\left(w_{\boldsymbol{\psi}}(\boldsymbol{s}, \boldsymbol{a})\right)^{\frac{1}{T}}}{\mathrm{E}_{\mathcal{D}_{\text{s}}}\left(\left(w_{\boldsymbol{\psi}}(\boldsymbol{s}, \boldsymbol{a})\right)^{\frac{1}{T}}\right)} \tag{44}$$

在与 SAC 算法和 TD3 算法结合后, LFIW 在大多数的 Mujoco 环境中都展现出来优于 PER 的性能. 但其在更大规模的环境 (例如 Atari) 中的表现有待进一步的实验验证.

基于 TD error 的 PER 算法和基于纠正性反馈的 DisCor 算法都没有直接针对 RL 的目标最小化策略遗憾来进行经验回放, 而是采用其他的替代指标作为回放标准. 经过大量的理论分析, Liu 等[58] 提出了基于神经网络的遗憾最小化经验回放方法 (Regret minimization experience replay using neural network, ReMERN), 对以下几种经验在回放时赋予更高的权重: 1) 具有更高的事后贝尔曼误差的经验; 2) 与当前策略更一致的经验 (LFIW 算法的关注点); 3) 与真实的最优价值估计更接近的经验; 4) 行动概率较小的经验. 在结合了 DisCor 和 LFIW 算法的优点后, ReMERN 算法也分别使用了两个参数为 $\boldsymbol{\phi}$ 和 $\boldsymbol{\psi}$ 的神经网络来估计 Q 值的累计误差和状态–动作对的权重. ReMERN 算法中经验的权重计算方式如下式所示

$$w_k(\boldsymbol{s}, \boldsymbol{a}) \propto \frac{d^{\pi_k}(\boldsymbol{s}, \boldsymbol{a})}{\mu(\boldsymbol{s}, \boldsymbol{a})} \exp\left(-\gamma \left[P^{\pi^{w_{k-1}}} \Delta_{k-1}\right](\boldsymbol{s}, \boldsymbol{a})\right) \tag{45}$$

其中, $d^{\pi_k}(\boldsymbol{s}, \boldsymbol{a})/\mu(\boldsymbol{s}, \boldsymbol{a})$ 衡量了状态–动作对 $(\boldsymbol{s}, \boldsymbol{a})$ 在第 k 次更新后的策略 π_k 下出现的概率, 其估计方式参考了 LFIW 算法的做法 (式 (43) 和式 (44)).

由于使用神经网络来估计 Q 值的累计误差耗时较长且准确性难以保证, Liu 等[58] 又提出了一种时间正确性估计的方法 (Temporal correctness estimation, TCE)

$$\begin{aligned} & |\, Q_k(\boldsymbol{s}, \boldsymbol{a}) - Q^*(\boldsymbol{s}, \boldsymbol{a}) \,| \approx \mathrm{E}_\tau(\mathrm{TCE}_c(\boldsymbol{s}, \boldsymbol{a})) = \\ & \quad \mathrm{E}_\tau\left(f\left(h_\tau^{\pi_{k-1}}(\boldsymbol{s}, \boldsymbol{a})\right)\left(L_{Q_{k-1}} + c\right) + \gamma^{h_\tau^{\pi_{k-1}}(\boldsymbol{s}, \boldsymbol{a})+1} c\right) \end{aligned} \tag{46}$$

其中, h_τ^π 为遵循策略 π 到轨迹 τ 的终点的时间距离, c 为状态 $\boldsymbol{s}$ 下最优动作 $\boldsymbol{a}^*$ 与当前动作 $\boldsymbol{a}$ 的真实 Q 值之差, $L_{Q_{k-1}}$ 为贝尔曼误差, 而 $f(t) = (\gamma - \gamma^t)/(1-\gamma)$.

在这种基于时序结构的遗憾最小化经验回放方法 (Regret minimization experience replay using temporal structure, ReMERT) 中, 经验的权重计算方式变为

$$w_k(\boldsymbol{s}, \boldsymbol{a}) \propto \frac{d^{\pi_k}(\boldsymbol{s}, \boldsymbol{a})}{\mu(\boldsymbol{s}, \boldsymbol{a})} \exp(-\mathrm{E}_\tau(\mathrm{TCE}_c(\boldsymbol{s}, \boldsymbol{a}))) \tag{47}$$

相较于 DisCor、SUNRISE 和 LFIW, ReMERN 和 ReMERT 较为全面地分析和证明了何种经验具有较高的回放价值. 除此之外, 这两种方法结合了多种经验加权回放算法的优势, 是此类算法中目前较为成熟的算法. 然而, 由于对 Q 值的累计误差的估计方式不同, 这两种算法适用于不同类型的 MDP. ReMERN 使用神经网络来进行误差估计, 能够适

用于多种不同的环境, 具有较强的鲁棒性. 而 ReMERT 在一些目标位置随机的环境中, 所提供的优先级权重差异性较大, 从而可能对策略的收敛产生误导, 但其估计方式的简便性仍是其不可忽略的优势.

2.1.3 经验分类回放

智能体与环境的大量交互经验具有不同的特征, 设定不同的指标 (分类标准) 来对经验进行分类存储和回放也是一种提升经验利用效率的有效途径.

经验池中存储的经验是在不同的训练阶段产生的, Zhang 等[59] 首先以经验的新鲜程度为标准进行了尝试, 设计了时间复杂度为 O(1) 的联合经验回放 (Combined experience replay, CER) 采样方法, 在智能体学习时将当前时间步的经验与采样的批量经验相结合来研究当前经验对算法性能的影响. 在不同环境的测试实验中, CER 可以有效减轻大规模经验池所产生的消极影响, 展现了强大的效率和稳定性.

由于 CER 仅对最新的经验 (当前经验) 进行了回放, 而没有对这些基于最新策略产生的最新经验进行充分利用. 针对这一问题, Hu 等[54] 从理论和实验的角度分别验证了最新经验对于智能体策略收敛的重要性. 他们所设计的 ACER 算法根据经验的新鲜程度设计了一个小容量的 FIFO 临时经验池暂存最新的交互经验. 在采样时, ACER 则将临时池中的经验完全复制并与原经验池中优先采样的经验结合共同构成采样的批量经验, 供智能体更新策略.

与经验的新鲜程度类似, Novati 等[60] 从策略的角度将经验池中经验按照其与当前策略的差异程度分为近策略 (Near-policy) 和远策略 (Far-policy), 并提出记忆与遗忘经验回放算法 (Remember and forget experience replay, ReFER). ReFER 通过只使用近策略的经验来更新策略, 并利用 KL 散度来限制策略的变化程度, 使网络更新和目标策略的收敛更为稳定. 通过与多种异策略 RL 算法的结合, ReFER 可以在多种连续控制任务中加快策略的收敛速度, 使算法的性能得到了明显提升.

不同于上述算法, 时圣苗等[61] 根据经验的 TD error 和奖励大小来对经验进行分类存储, 提出了时间差分误差分类 (TD error classification, TDC) 和奖励分类 (Reward classification, RC) 两种经验分类方法. 这两种方法采用同一训练架构, 均设定了两个相同大小的经验池来存储经验, 采样时采用固定比例的静态采样方法. 在与 DDPG 算法结合后, TDC 和 RC 均在连续控制任务中表现出较优的结果. 然而在训练初期, TD error 和奖励值较大的经验的数量相对较少, 从而导致这些较优的经验被多次重复学习, 这可能会对算法的稳定性产生影响.

刘晓宇等[62] 同样以 TD error 作为经验权重, 提出了动态优先级并发接入算法 (Concurrent access algorithm with dynamic priority, CADP), 将经验按照 TD error 进行分类存储. 与 TDC 算法不同的是, CADP 采取了一种变化比例的动态采样方式分别从不同经验池进行采样学习

$$g^{r}_{\mathrm{A,B}} = g^{\mathrm{ini}}_{\mathrm{A,B}} - g^{\mathrm{dec}}_{\mathrm{A,B}} \tag{48}$$

其中, $0 < g^{r}_{\mathrm{A,B}} \leq 1$是 A、B 经验池的采样概率, $g^{\mathrm{ini}}_{\mathrm{A,B}}$ 为 A、B 经验池的初始采样概率, $g^{\mathrm{dec}}_{\mathrm{A,B}}$ 为采样概率的衰减值.

受 AER 算法启发, 智能体的状态也可以作为经验的分类标准. Hu 等[49] 提出了经验分割算法 (Experience pool split, EPS) 来实现根据智能体状态的不同对经验进行分类存储和利用. 在训练前, EPS 需要根据过往经验的比例对经验池进行分割. 训练时, EPS 根据所设定的基于场景的评价指标将交互经验存入对应经验池, 并依据当前智能体状态从对应经验池中采样进行训练, 在训练后期再将分割的经验池进行合并并打乱所有经验的排列顺序. 然而在训练的过程中, 经验的比例是不断变化的, 如何确定 EPS 中使结果最优的经验池分割比例仍有待进一步探究.

相较于难以实现的对所有经验进行分类存储, 朱斐等[63] 从智能体状态安全性的角度只考虑探索失败时的危险状态对应的经验, 提出了一种基于双深度网络的安全深度强化学习算法 (Dual deep network based secure DRL, DDN-SDRL). DDN-SDRL 在经验回放方面设计了危险样本经验池和安全样本经验池来对经验进行分类存储, 并引入安全强化学习的概念定义了两种危险状态: 1) 智能体在任务失败时的状态; 2) 智能体在任务失败前的 m 个时间步状态. DDN-SDRL 方法针对性地学习了危险状态的经验, 智能体在这些危险状态附近的行动策略会受到较大影响, 从而避免陷入局部最优, 有效地限制了智能体向危险状态方向的探索.

如表 3 所示, 经验分类回放的标准无非还是经验优先回放类算法的优先性标准及其变体, 使用较多的仍是基于策略的、基于 TD error 的和基于状态的. 从本质上讲, 经验分类回放就是按照经验优先回放的标准对经验进行分类存储, 在采样时利用特定的策略从而实现对这些符合标准的经验的优先回放. 大多数的经验分类回放算法采用多经验池的架构来减小经验优先回放类算法的采样时间复杂度, 但这个做法会引入一个动态或静态的采样策略,

表 3　经验分类回放算法对比

Table 3　Comparison of classification experience replay algorithms

算法	分类标准	经验池形式	采样策略
CER[59]	是否为当前经验	单经验池 + 临时存储	随机采样 + 当前经验
ACER[54]	是否为最新经验	多经验池	优先采样 + 最新经验
ReFER[60]	经验策略与当前策略的差异	单经验池	随机采样 + 经验过滤
RC[61]	奖励	多经验池	静态采样
TDC[61]	TD error	多经验池	静态采样
EPS[49]	基于场景的评价指标	多经验池 + 单经验池	静态采样
CADP[62]	TD error	多经验池	动态采样
DDN-SDRL[63]	状态的危险程度	多经验池	静态采样

该策略的好坏将会对算法的效果产生巨大的影响.

2.1.4　经验表示形式

在 DRL 中, 交互经验 $(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}_{i+1})$ 通常是高维度的向量, 使用大容量的经验池存储数以百万计的经验需要耗费大量的计算机内存. 一些研究通过将向量形式的经验转换为其他的表示形式, 从而实现对经验更高效的存储和利用.

Wei 等[64] 创新性地将量子的一些特性引入到 RL 中, 来对经验回放机制进行改进, 提出了一种符合自然规律且易于使用的量子经验回放方法 (Quantum-inspired experience replay, QER). QER 通过将经验转换为量子化的表达, 同时又对量子表达使用酉变化, 使得 RL 中容量为 M 的经验池的状态可以被表示为量子子系统张量积的形式

$$|\psi^{\text{total}}\rangle = |\psi^{(1)}\rangle \otimes |\psi^{(2)}\rangle \otimes \cdots \otimes |\psi^{(M)}\rangle \tag{49}$$

其中, $|\psi^{(k)}\rangle$ 为第 k 个经验的量子表示形式. 如图 7 所示, QER 设计了准备操作和折旧操作两种酉操作. 准备操作可以使得量子化表达的经验的概率幅与其 TD error 相匹配, 而折旧操作可以使同一经验被多次重复学习的概率降低, 使得采样到的经验更多样化, 回放更均匀.

在 QER 的基础上, Li 等[65] 进行了更具有实际意义的探索. 他们将无人机的导航任务映射到 MDP 上, 考虑了时间成本和预期中断持续时间的加权和的最小化问题, 利用 QER 制定了一种智能的无人机导航方法来帮助无人机在每个时间步内找到最佳飞行方向, 并在复杂 3D 环境任务中验证了 QER 的有效性和稳定性.

Chen 等[66] 设计了一种局部敏感性经验回放算法 (Locality-sensitive experience replay, LSER), 该算法使用局部敏感的哈希法将 RL 中的高维经验映射到低维表示, 解决了 RL 应用于推荐系统时经验维度过高的问题. 除此之外, LSER 采用了一种状态感知–奖励驱动的采样策略, 即采样与当前状

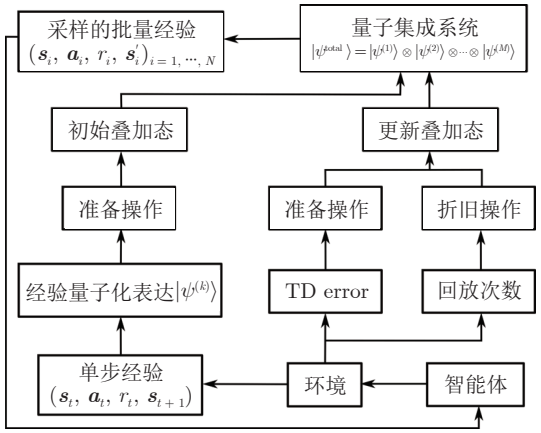

图 7　QER 的算法框架

Fig. 7　The framework of QER algorithm

态位于同一哈希域中前 N 个具有最高奖励值的经验. LSER 可以高效地选择所需要的经验来训练智能体, 在多个仿真平台的实验中证明了其可行性以及相对于其他经验回放方法的优越性.

对于经验表示形式的相关研究相对较少, 这是由于无论什么形式的经验都需要被存储, 即使变换存储形式来减小所需的内存空间, 经验的编码和解码也会对算法的效率产生一定的影响. 除此之外, 很难找到符合条件的一一映射来将高维经验进行降维存储, 这也限制了此方向的进一步发展.

2.1.5　经验存储结构

设计良好的经验存储结构能够更高效地实现大容量经验池的存储、更新和采样等操作, 这也是一种提升 DRL 算法效率的有效途径.

Schaul 等[43] 在设计 PER 算法时就考虑到了传统数组或堆栈形式的数据结构在进行经验优先利用时的过高的时间复杂度, 设计了一种名为 "sum-tree" 的完全二叉树的数据结构. 如图 8 所示, "sum-tree" 的每一个叶子结点存储的都是对应经验的优先级, 在采样时遵循以下两个步骤: 1) 判断当前节

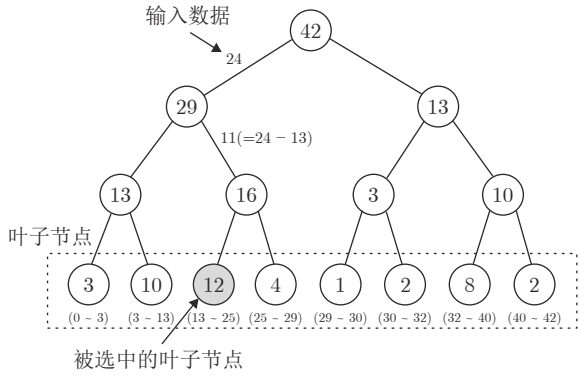

图 8 “sum-tree” 采样流程

Fig. 8 The sampling process of “sum-tree”

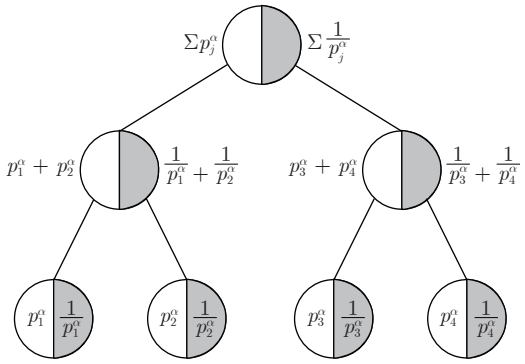

图 9 “double sum-tree” 数据结构

Fig. 9 The data structure of “double sum-tree”

点是否为叶子节点, 如果是, 则当前节点是应该采样的节点; 2) 比较输入值与当前节点的左子节点的值. 如果左子节点的值较大, 则将左子节点设置为当前节点并重复步骤 1), 否则将右子节点设置为当前节点, 并用输入值与左子节点的值的差值替换输入数据, 然后重复步骤 1).

“sum-tree” 的使用将采样具有最大优先级的经验和更新被采样的批量经验的优先级的时间复杂度分别降低至 O(1) 和 $O(\log_2 N)$ (N 为每次采样的经验数量), 大幅降低了经验选择对算法运行速率的影响, 有效地提升了 PER 算法的效率.

Hu 等[54] 提出的 ACER 算法也在经验池的数据结构上做出了改进. 与 PER 不同, 他们认为当经验池满时, 不应替换较古老的经验而应该替换较无用的经验, 进而提出了一个先入无用出 (First in useless out, FIUO) 的数据结构 “double sum-tree”.

如图 9 所示, “double sum-tree” 相较于 “sum-tree” 在每个叶子节点多存储了对应经验的优先级的倒数, 使得经验 $\boldsymbol{e}_i$ 被替换的概率为

$$P_{\text{rep}}\ (\boldsymbol{e}_i) = \frac{\frac{1}{p_i^\alpha}}{\sum\limits_j \frac{1}{p_j^\alpha}} \tag{50}$$

ACER 在每个时间步仅耗费了多于 PER 算法 $O(\log_2 N)$ 的时间复杂度来更新优先级的倒数, 这相较于其对算法性能的提升是可以被接受的.

类似地, Chen 等[66] 提出的 LSER 算法和 Brain 等[67] 提出的基于分布的经验保留算法 (Distribution based experience retention, DER) 也都尝试从更新逻辑上对经验进行改进. 当经验池满时, LSER 会优先替换经验池中奖励值较低的经验, 而 DER 则给出一种可以使得经验池中的经验保持在状态–动作空间上近似均匀分布的方式来从经验池中选择经验进行替换. 然而, 在不对数据结构进行改变的情况下, 每一条新经验的到来都需要在庞大的经验池中筛选其要替换的经验, 这无疑会给算法的计算效率带来巨大的压力.

Li 等[68] 则从硬件架构方面入手, 使用了一种硬件–软件协同的方法来设计基于关联存储器的优先经验回放算法 (Associative memory based PER, AMPER). AMPER 使用三元内容可寻址存储器 (Ternary content addressable memory, TCAM) 取代了 PER 一类算法中广泛使用的较为耗时的基于树结构遍历的优先级采样. 除此之外, AMPER 还使用了一种基于关联存储器的内存计算硬件架构, 通过利用并行内存搜索操作来支持算法的运算. 在文献 [68] 所建议的硬件上运行时, AMPER 算法表现出与 PER 算法相当的学习性能, 同时实现了 55 ~ 270 倍的延迟改进.

表 4 给出了上述经验回放算法在经验存储结构方面相较于传统的经验回放算法做出的优化. 可以看到, 目前大多数研究主要从数据结构、更新逻辑和硬件架构三个方面来对经验池进行优化. 除了传统的 FIFO 数据结构, 目前经验回放类算法最常使用的就是 “sum-tree” 结构, 而最近提出的 “double sum-tree” 结构还未得到广泛的认可和应用. 经验池的更新逻辑方面, 除了传统的经验回放算法依据经验的存储时间进行更新, 新的算法主要以 TD error 和奖励作为经验被替换的标准. 而 AMPER 首次从硬件的角度入手, 为此方向的研究提供了一个新的思路.

2.2 经验增广

2.2.1 专家示范经验

RL 算法通常需要大量数据才能使智能体获得较为合理的策略. 这对于在仿真平台上的 RL 任务来说也许是可以接受的, 但这严重限制了 RL 对许多实际任务的适用性. 在 RL 这种智能体从对环境

表 4　经验存储结构算法的优化途径

Table 4　Optimization approaches of experience storage structure algorithms

算法	数据结构	更新逻辑	硬件架构
PER[43]	√		
ACER[54]	√	√	
LSER[66]		√	
DER[67]		√	
AMPER[68]	√		√

一无所知经过不断探索后找到最优策略的过程中, 如果在训练过程中能为智能体提供较为基础可靠的行动策略, 将大大减少智能体训练前期的探索过程. 一些研究者从经验回放的角度使用模仿学习 (Imitation learning, IL)[69−70] 这种监督学习的思路将人类专家或其他来源的策略以不同形式的经验记录下来, 以供智能体进行预训练或在训练过程中对智能体的策略进行修正和完善, 从而改善 RL 算法的性能.

Hester 等[71] 从预训练的角度提出了示范深度 Q 学习 (Deep Q-learning from demonstrations, DQfD). DQfD 的训练分为预训练和实际训练两个阶段. 在训练开始前, 先由人类专家在该游戏环境中操作 3 到 12 个回合来生成示范经验. 在训练时, DQfD 会先根据人类的示范经验进行固定回合的预训练, 随后回归原始的自主交互式的学习. 实验结果显示, DQfD 在 42 种视频游戏中的 14 种都超越了人类最佳的示范水平, 并且在 11 种游戏中取得了最先进的结果. 然而这种利用专家示范经验进行预训练的算法, 专家示范的经验数量对其预训练效果有很大影响, 一般来说, 专家经验的数量越多, 预训练的效果就越好, 其收集专家经验所耗费的资源也就越多.

对于动作空间连续的任务场景, Vecerik 等[72] 将这种专家示范的思路引入到了 DDPG 算法中并设计了示范深度确定性策略梯度算法 (DDPG from demonstrations, DDPGfD). 与 DQfD 不同, 对于每一个任务, DDPGfD 都提前收集了 100 个回合的人类示范经验, 并永久存入经验池中以便后续学习. DDPGfD 使用经验优先回放在示范经验和交互经验共同构成的经验池中进行优先级排序, 以一种自然的方式控制两者之间的数据比例, 其中经验的优先级计算方法如下

$$p_i = \delta_i^2 + \lambda \left|\nabla_{\boldsymbol{a}} Q(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}^Q)\right|^2 + \varepsilon + \varepsilon_{\mathrm{D}} \tag{51}$$

其中, δ 为 TD error, ε 是正向常数, ε_{D} 是一个常数来增加示范经验的优先级, λ 是用来调整权重的常数, $\left|\nabla_{\boldsymbol{a}} Q\left(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \boldsymbol{\theta}^Q\right)\right|^2$ 为作用于演员网络的损失. 在多种复杂度机械臂控制任务中, DDPGfD 不仅能够很好地完成任务, 并且能够达到比人类示范更可靠的效果.

与 DQfD 直接对网络进行预训练不同, Guillen-Perez 等[73] 额外使用了一个名为 “Oracle” 的演员网络, 在训练时令 “Oracle” 网络直接从仿真平台提供的示范经验进行学习, 并在每一次参数更新时对 TD3 算法的演员网络进行小规模的软更新. 这种从 “Oracle” 示范中学习 (Learning from Oracle demonstrations, LfOD) 的方法在进行动作选择时, 智能体会按照一个随训练过程不断变化的概率来选择使用 “Oracle” 或 TD3 的演员网络进行动作选择. 在经验存储方面, LfOD 有两个规模不同的经验池: $D^{\mathrm{imitation}}$ 和 $D^{\mathrm{reinforcement}}$. $D^{\mathrm{imitation}}$ 仅存储由 “Oracle” 选择的动作产生的经验, 而 $D^{\mathrm{reinforcement}}$ 存储所有演员网络产生的经验并使用 PER 进行优先经验回放. 实验表明, LfOD 能够有效地提升 RL 应用于自动路口管理 (Autonomous intersection management) 任务的效率.

Huang 等[74] 提出的模仿专家经验算法 (Imitative expert priors, IEP) 也通过训练一个专家网络来实现对专家策略的模仿, 并通过正则化智能体策略 $\pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{s}_i)$ 与专家策略 $\pi^{\mathrm{E}}(\cdot \mid \boldsymbol{s}_i)$ 之间的 KL 散度来指导自动驾驶智能体的学习. IEP 提供了两种不同的方式来实现智能体对专家经验的充分学习:

1) 对值函数添加惩罚项

$$\bar{r}(\boldsymbol{s}_i, \boldsymbol{a}_i) = r(\boldsymbol{s}_i, \boldsymbol{a}_i) - \alpha D_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{s}_i), \pi^{\mathrm{E}}(\cdot \mid \boldsymbol{s}_i)) \tag{52}$$

2) 在策略优化期间控制智能体策略与专家策略之间的偏差

$$L(\boldsymbol{\theta}) = \mathop{\mathrm{E}}_{\boldsymbol{s}_i \sim D, \boldsymbol{a}_i' \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{s}_i)} \Big(- \min_{j=1,2} Q_j(\boldsymbol{s}_i, \boldsymbol{a}_i') + \lambda \left(D_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{s}_i), \pi^{\mathrm{E}}(\cdot \mid \boldsymbol{s}_i)) - \varepsilon \right) \Big) \tag{53}$$

实验表明, 使用这两种方式的 IEP 均能在多个城市驾驶场景中获得最高的成功率, 并能够有效地表现出人类专家所展示的多样化的驾驶行为, 而对值函数添加惩罚项的 IEP 算法在稀疏奖励的场景中表现效果更优.

Hu 等[75] 在处理无人机的运动控制问题时, 设计了一个多经验池算法 (Multiple experience pool, MEP). MEP 通过使用模型预测控制 (Model predictive control) 来预测智能体在未来多个时间步的动作序列, 并利用模拟退火算法 (Simulated annealing) 从预测结果中选择最佳序列, 从而利用传统控制算法实现了对专家经验的高效模拟. 这种多经验池的架构可以存储多种不同来源的高质量经

验, 使智能体可以在多种专家策略之间学习. 但多种专家经验之间的矛盾可能会给智能体策略的收敛造成困难, 如何使智能体良好地平衡策略之间的差异性还有待进一步的研究.

Wan 等[76] 则将这种使用其他算法来模拟专家经验的思路应用到多智能体 RL (Multi-agent RL, MARL) 中, 提出了一种混合经验算法 (Mixed experience, ME), 并结合多智能体深度确定性策略梯度算法 (Multi-agent DDPG, MADDPG)[77] 在多无人车的运动规划任务中展现了优越的效果. ME 算法通过一个基于人工势场法 (Artificial potential field) 的经验生成器在训练时为智能体提供具有指导性的高质量经验, 并使用动态混合采样的策略, 以可变的比例混合来自不同来源的训练经验, 来优化智能体的运动策略.

针对紧急情况下自主水下航行器在三维空间中的浮面控制问题, Zhang 等[78] 提出了一种基于 DDPG 的无模型 DRL 算法 (Variable delay DDPG from demonstration, VD4). VD4 不仅使用与 DQfD 类似的思路来利用专家示范经验对目标网络进行预训练, 还在实际训练时按照比例对专家经验和交互经验进行采样. 这种做法有效地提升了算法的收敛速度, 提高了应对对抗性攻击时的鲁棒性. 除此之外, VD4 还采用了 TD3 中的演员网络延迟更新操作来提高算法的稳定性. 在多种不同的复杂测试环境下, 相较于 DDPG、TD3 等算法, VD4 算法所训练的智能体都能够达到最高的任务成功率.

表 5 对上述的专家示范经验算法从专家经验来源、作用方式、经验池形式、采样策略和算法应用场景进行了总结. 总的来看, 使用不同形式的专家示范来实现经验的增广能够有效地提升 DRL 算法的性能, 尤其是在一些复杂的控制任务中. 但由于专家示范时的思路或策略可能与智能体所学习的策略不同, 甚至专家可能会使用一些智能体无法理解和表示的策略, 这都会对智能体的策略收敛产生不利影响, 如何缩小专家示范与交互经验之间的差异性是在进行此方向研究时要考虑的主要问题.

2.2.2 模型经验增广

环境在 RL 中主要起到根据智能体动作进行状态转移并给予其反馈的作用. 一些研究者通过在 RL 训练过程中构建一个合理的环境模型来实现对环境的模拟. 如图 10 所示, 这类基于模型的经验增广方法在原始的 RL 循环 (虚线内所示) 外增添了环境模型的训练过程, 在环境模型的准确性能够保证的情况下, 可以使智能体在与环境模型的交互过程中轻松获得大量高质量经验, 从而实现策略的快速收敛.

Sutton 等[79] 最先将建立环境模型的想法引入到 Q-learning 算法中并提出了 Dyna-Q 算法. 在 Dyna-Q 中, 智能体与环境的在线交互经验不仅用于智能体策略的更新, 还用于环境模型的训练. Dyna-Q 使用了一种 Q-planning 的方法来随机生成多个之前经历过的状态和在该状态下执行过的动作作为输入, 与环境模型的输出共同构成模拟经验来更新状态–动作值函数, 从而实现经验的增广, 使得智能体在实际任务中能够获得更优的策略.

Silver 等[80] 在 Dyna-Q 的基础上进一步探索, 提出了具有 SARSA 算法的更新机制、永久和瞬态记忆以及线性函数逼近的 Dyna-2 算法. Dyna-2 将 Q 函数分为永久记忆 $Q(\boldsymbol{s}, \boldsymbol{a})$ 和瞬态记忆 $Q'(\boldsymbol{s}, \boldsymbol{a})$. 永久记忆根据智能体的交互经验进行更新, 而瞬态记忆则在与模型的模拟过程中得到更新来形成对永久记忆的局部校正. 在每次动作选择前, 智能体会根据环境模型执行一个从当前状态持续到回合结束的模拟过程, 随后根据完整的 Q 值选择动作

$$\bar{Q}(\boldsymbol{s}, \boldsymbol{a}) = Q(\boldsymbol{s}, \boldsymbol{a}) + Q'(\boldsymbol{s}, \boldsymbol{a}) \tag{54}$$

针对 Dyna-Q 在生成模拟经验时随机性过大的问题, Santos 等[81] 提出了一种结合启发式搜索的算法框架 Dyna-$\mathcal{H}$. Dyna-$\mathcal{H}$ 设计了一个启发式规划模块 $\mathcal{H}$, 通过计算状态 $\boldsymbol{s}'$ 与目标位置 $goal$ 之间的欧氏距离来评价在状态 $\boldsymbol{s}$ 下所选择的动作 $\boldsymbol{a}$ 的好坏

表 5 专家示范经验算法对比

Table 5 Comparison of expert demonstration experience algorithms

算法	专家经验来源	专家经验作用方式	经验池形式	采样策略	应用场景
DQfD[71]	人类示范	预训练	单经验池	优先采样	视频游戏
DDPGfD[72]	人类示范	实际训练	单经验池	优先采样	机械臂控制
LfOD[73]	仿真平台	专家网络 + 实际训练	多经验池	动态采样 + 优先采样	自动路口管理
IEP[74]	人类示范	专家网络 + 实际训练	单经验池	随机采样	自动驾驶
MEP[75]	模拟退火算法	实际训练	多经验池	动态采样	无人机运动控制
ME[76]	人工势场法	实际训练	多经验池	动态采样	多无人车运动规划
VD4[78]	人类示范	预训练 + 实际训练	多经验池	优先采样	自主水下航行器控制

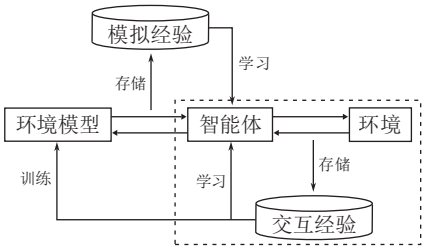

图 10　模型经验增广算法的框架图

Fig. 10　The framework of model experience augmentation algorithms

$$\mathcal{H}(\boldsymbol{s},\boldsymbol{a})=\|\boldsymbol{s}'-goal\|^2 \tag{55}$$

其中, $\boldsymbol{s}'$ 是环境模型给出的下一时刻状态. 在模型训练时, Dyna-$\mathcal{H}$ 根据 $\mathcal{H}$ 选择启发式的动作 $h_{\boldsymbol{a}}$ 来构成模拟经验

$$h_{\boldsymbol{a}}(\boldsymbol{s},\mathcal{H})=\arg\max_{\boldsymbol{a}}\mathcal{H}(\boldsymbol{s},\boldsymbol{a}) \tag{56}$$

实验表明, 相对于 Dyna-Q, Dyna-$\mathcal{H}$ 是一种效率更高的算法, 尤其适用于最优路径搜索这一类决策问题.

针对在生成模拟经验时搜索控制的局限性, Pan 等[82] 提出了一种爬山 Dyna (Hill climbing Dyna, HC-Dyna). HC-Dyna 在学习到的值函数上使用了一种噪声不变的投影梯度上升策略的爬山法来生成模拟经验, 并引入了一个阈值来保证模拟经验之间的差异性, 从而使智能体能够提前更新其接下来可能访问的区域. HC-Dyna 还使用了一种经验混合机制, 将搜索控制产生的模拟经验和交互产生的真实经验按比例采样, 共同用于智能体策略的更新. 在多个 Atari 游戏场景的实验中, HC-Dyna 都显示了优于 DQN 的性能.

除了模拟经验的质量, Dyna 框架中模拟经验的使用顺序也很重要. Pan 等[83] 在研究了不同经验的重要性后, 引入了一种重加权经验模型 (Reweighted experience models, REM) 的半参数模型学习方法来调整模拟经验的使用顺序. REM 具有以下几个优势: 1) 可以快速地选择和采样某个经验的前序或后序经验; 2) 包含需要学习的参数较少, 数据高效性较强; 3) 可以提供足够的模型复杂性. 实验表明, REM 可以高效地使用模拟经验和交互经验. 更进一步的探索表明, 相较于线性模型和神经网络模型, REM 是更适合 Dyna 框架的模型.

构建环境模型能够在减少智能体与环境交互的情况下产生大量的模拟经验. 从早期的 Dyna-$\mathcal{H}$ 到近几年所提出的 HC-Dyna, 如何产生更优的经验一直是此领域研究所关注的重点. REM 算法则对 Dyna 框架下模拟经验的利用顺序进行了探索, 为模型经验增广提供了一个新的研究方向.

2.2.3　事后经验回放

除了上述通过智能体与环境交互过程外部提供增广经验的方式, 一些研究者希望通过交互过程本身而不引入其他经验来源的方式实现经验的增广.

Andrychowicz 等[84] 创新性地提出了事后经验回放 (Hindsight experience replay, HER). 作为一种多目标 RL (Multi-goal RL, MGRL)[85] 的方法, HER 在动作选择和反馈奖励时需要同时参考状态 $\boldsymbol{s}$ 和目标 $\boldsymbol{g}$. 对于要重塑的经验 $(\boldsymbol{s}_t\|\boldsymbol{g},\boldsymbol{a}_t,r_t,\boldsymbol{s}_{t+1}\|\boldsymbol{g})$, 首先根据特定方法选择附加目标 $\boldsymbol{g}'$, 随后根据附加目标评价在状态 $\boldsymbol{s}_t$ 下动作 $\boldsymbol{a}_t$ 的好坏并获取新的奖励值 r_t', 从而构成一个新的经验 $(\boldsymbol{s}_t\|\boldsymbol{g}',\boldsymbol{a}_t,r_t',\boldsymbol{s}_{t+1}\|\boldsymbol{g}')$ 用于后续的采样和训练. HER 通过对过去经验进行重塑, 显著地提高了稀疏奖励环境中高质量经验的百分比, 从而加速了智能体的训练过程. 由于在任务中使用了固定的奖励函数, HER 可以保证事后产生增广经验的高质量, 并确保智能体收敛策略的统一性.

为了克服 HER 基于均匀采样来生成附加目标的局限性, Luu 等[86] 提出了事后目标排名算法 (Hindsight goal ranking, HGR). 基于 TD error, HGR 使用了两种优先化操作, 即回合优先和目标优先

$$n\sim P(n)=\frac{\left|\delta^{(n)}\right|^{\alpha}}{\sum\limits_{n}\left|\delta^{(n)}\right|^{\alpha}} \tag{57}$$

$$i\sim P'(j,i)=\frac{|\delta_{ji}|^{\alpha'}}{\sum\limits_{j=1}^{H-1}\sum\limits_{i=j+1}^{H}|\delta_{ji}|^{\alpha'}},\ i\in\{j+1,\cdots,H\} \tag{58}$$

其中, $\delta^{(n)}=\sum_{k=1}^{K}|\delta_k^{(n)}|/K$ 是长度为 K 的回合中所有经验的 TD error 的平均值, $\delta_k^{(n)}$ 为第 n 个回合中第 k 条经验的优先级. 在采样时, HGR 首先根据回合采样概率 P 在所有存储的回合中进行优先采样, 随后再对采样到的回合根据目标采样概率 P' 在其中优先采样出多个经验用以生成附加目标. 实证结果表明, HGR 能够更有效地使用经验, 在多个具有挑战性的模拟机器人操作任务上都能够显著加快 DDPG 算法的收敛速度. 但毫无疑问这种对所有存储的经验进行遍历的方法的效率有待进一步提高.

为了能够使 HER 算法适用于动态目标问题, Fang 等[87] 提出了动态事后经验回放算法 (Dynamic hindsight experience replay, DHER). DHER 设计了一个存储器来存储所有的失败回合. 在回放时, DHER 会搜索满足 $\boldsymbol{g}_{i,p}^{ac}=\boldsymbol{g}_{j,q}^{de}$ 的两条相匹配的失败

轨迹 E_i 和 E_j ($i \neq j$), 其中 $\boldsymbol{g}^{ac}_{i,p}$ 是在回合 i 中时间步 p 时智能体的位置, $\boldsymbol{g}^{de}_{j,q}$ 是在回合 j 中时间步 q 时目标的位置. 随后将使用 $\boldsymbol{g}^{de}_{j,t}$ 替换 E_i 中原始的目标位置, 其中 $t \leq \min\{p, q\}$ 来保证新生成的成功回合 E'_i 中的目标轨迹和智能体轨迹长度相等.

DHER 在处理动态目标问题时取得了很好的效果, 但它仍然存在以下几个缺点: 1) 失败的回合经验的存储需要大量的计算机内存; 2) 在存储的失败经验中搜索和匹配的时间复杂度巨大; 3) 在高维连续状态空间环境中很难找到两条匹配的失败轨迹; 4) 并非所有的增广经验都值得被存储和学习; 5) 无法保证存储在庞大经验池中的增广经验可以被学习到. 针对这些问题, Hu 等[88] 提出了想象过滤事后经验回放 (Imaginary filtered hindsight experience replay, IFHER). IFHER 通过合理想象失败回合中的目标轨迹来生成成功回合

$$\tilde{\boldsymbol{g}}_t = \boldsymbol{g}_{T-n-1+t} + \tilde{\boldsymbol{g}}^k - \boldsymbol{g}_{T-1} + \tilde{\varepsilon} \tag{59}$$

其中, $\tilde{\boldsymbol{g}}^k \in \tilde{\boldsymbol{G}}$ 是选择的附加目标的轨迹终点, $\tilde{\boldsymbol{g}}_t$ 是想象的目标位置, $\tilde{\varepsilon}$ 是一个随机目标噪声, 在增加目标多样性的同时增强对目标空间 $\boldsymbol{G}$ 的探索. 通过这种方式, IFHER 使得每一个失败的回合都能够被重塑为成功的回合. 除此之外, IFHER 还设计了目标、情节、质量和采样过滤策略来分别确保存储的增广经验的高质量和增广经验被采样的优先性.

从附加目标多样性的角度出发, Fang 等[89] 提出了一种课程指导的事后经验回放算法 (Curriculum-guided hindsight experience replay, CHER), 通过对失败经验自适应地选择来动态控制探索与利用的平衡. CHER 设计了如下的效用得分函数来评估附加目标优劣

$$F(i \mid A) = \lambda \mathrm{sim}(\boldsymbol{g}_i, \boldsymbol{g}) + \sum_{j \in B} \max\left\{0, \mathrm{sim}(\boldsymbol{g}_i, \boldsymbol{g}_j) - \max_{l \in A} \mathrm{sim}(\boldsymbol{g}_l, \boldsymbol{g}_j)\right\} \tag{60}$$

其中, $\mathrm{sim}(\cdot, \cdot)$ 是距离函数, 用来估计两个目标之间的相似性, A 是采样的批量大小, B 是整个经验池. 前一项衡量了附加目标与真实目标的接近程度, 后一项衡量了附加目标的多样性. 在训练过程中, CHER 逐渐增加权重 λ 的大小, 在早期的训练阶段强制学习目标更多样的经验, 并逐渐更改学习目标更接近真实目标的经验. 实验结果表明, 在多种具有挑战性的机器人环境中, CHER 都可以进一步提升当前最优算法的表现.

Yang 等[90] 则将环境模型引入 HER, 并提出了模型事后经验回放算法 (Model-based hindsight experience replay, MHER). 在为经验 ($\boldsymbol{s}_t \| \boldsymbol{g}$, $\boldsymbol{a}_t$, r_t, $\boldsymbol{s}_{t+1} \| \boldsymbol{g}$) 选择附加目标时, MHER 会先使用环境模型根据状态 $\boldsymbol{s}_t$ 向前进行 n 个时间步的模拟, 随后从这些状态中选择附加目标进行经验重塑. MHER 主要具有以下几个优势: 1) 通过模型交互产生的附加目标不再局限于真实经验; 2) 附加目标的生成遵循一种策略指导的高效课程; 3) 策略更新同时利用了强化学习和监督学习. 在多种连续的多目标任务中的实验表明, MHER 具有优于 HER 和 CHER 的样本效率.

事后经验回放发展至今, 从 HER 的均匀采样, 到 HGR 基于 TD error 的采样, 再到目前基于课程和基于模型采样的 CHER 和 MHER, 如何为一条失败的轨迹确定附加目标始终是研究者们关注的重点. 采用特定的策略、选择合适的附加目标, 从而生成高质量的增广经验来提升算法的效率在今后依然会是此类方法的主要研究目标.

3 总结与展望

RL 通过智能体与环境的交互过程不断获取经验, 来优化智能体自身的行动策略以期获得最大的累积奖励. 作为一种数据驱动的机器学习算法, 经验 (数据) 决定了 RL 智能体最终策略的优劣. 在异策略 RL 中, 经验池的存在造就了经验回放这一研究热点. 通过经验回放, 智能体能够按照需求合理利用多来源的经验, 避免灾难性遗忘的发生, 更快地得到更优的行动策略, 减小训练过程中的成本代价. 因此, 对经验回放机制进行研究有着十分重要的实际意义和发展前景. 本文从 RL 的基础知识出发, 介绍了常用的异策略 RL 算法, 并从经验利用和经验增广两个角度对经验回放机制的相关研究进行了详细的介绍和总结, 弥补了国内相关研究领域的空缺.

现有的经验回放方法已经取得了初步的成果, 理论和实践都证明了经验回放对于异策略 RL 的重要性, 但其仍面临着一些问题和挑战.

1) 算法的适用性: 很多算法过分关注于某类问题而局限了其适用范围. 而且大量的算法使用了神经网络或引入了大量的超参数, 参数敏感性无疑会影响经验回放算法对不同环境的适用效果. 除此之外, 作为 RL 的一个模块, 经验回放算法同样面临着虚拟到现实的落地困难的窘境.

2) 算法的可解释性: RL 作为人工智能领域的一个研究方向, 毫无疑问在经验回放的研究过程中, 会有很多思路和设计来源于对人类或其他物种的行为模拟. 然而, 这些所谓移植性创新往往缺乏可靠

的理论支撑, 可解释性较差, 有时难以让人信服.

3) 算法的效率: 无论是在数以百万计的经验池中进行筛选或排序, 还是从无到有地增广数以百万计的经验, 经验回放算法的运算效率始终面临着严峻的考验.

针对上述问题, 本文仍从经验利用和经验增广两个方面分别指出各个方向可能的突破口, 为相关领域的学者提供一些研究思路.

在经验利用方面:

1) 经验优先利用标准: 除了已得到广泛认可的 TD error 常用来作为经验优先利用的标准外, 经验与当前状态的相似性、经验难度等指标也逐渐被应用于各种 RL 控制问题中. 调整智能体对经验池中所存储的经验的学习顺序, 在不同的训练阶段使得最适合当前智能体策略更新的经验被学习, 仍然是一种提高 RL 效率的有效办法[51]. 除了上述的优先标准, 将其他领域例如人类教育学、心理学或者生物学的知识或现象进行建模, 设计出普适于多种任务环境的参数不敏感的优先回放标准仍是经验回放领域今后研究的重点.

2) 经验加权回放的适用性: 经验加权回放是近几年经验回放领域新兴的研究方向, 相关的研究还相对较少. 目前较为成熟的算法还难以简便、准确地应用于所有的实验环境. 因此, 设计更具有普适性且计算更为简洁的经验加权回放方法仍是现阶段需要主要考虑的问题[58]. 除此之外, 由于发展时间较短, 还尚未有研究将此类方法延伸于真实的应用环境, 其在真实环境中是否仍具有较好的效果还有待进一步研究.

3) 经验池的结构设计: 随着任务复杂程度的提升, 经验池的规模也在不断扩大, 经验池的结构设计面临着巨大的压力. 通过改变原有数组形式的存储结构, 采用树、队列、堆栈甚至于图等数据结构来构建经验池, 能在降低计算机内存消耗的同时提升算法的采样效率. 合理设计经验池的更新逻辑也能够充分发挥各种经验优先利用算法的优势[54]. 除此之外, 对计算机的硬件架构进行针对性的设计也是进一步提升经验回放算法效率的有效途径.

4) 经验的表示形式及转换效率: 利用其他领域例如量子力学的理论, 将原始向量形式的经验进行编码, 随后进行存储. 在使用时也可以结合不同的优先利用算法来进行优先采样, 将采样后的经验重新解码以便智能体进行学习. 这种从经验的表示形式入手的方法也是近年来经验回放领域的一个新兴研究方向, 具有一定的研究价值. 但这种先编码后解码的过程需要耗费一定的计算资源, 如何提升经验编码和解码的效率是此类方法需要考虑的重点.

在经验增广方面:

1) 多来源数据的准确性: 在处理较为复杂任务时, 结合不同来源的指导经验进行学习可以大幅减少智能体与环境的交互, 提升算法的收敛速度. 但如何确保其他数据的准确性和有效性是此研究方向不可避免的问题. 专家经验通常是基于专家自身实际操作经验, 智能体的交互经验是在奖励函数的指导下产生的. 专家经验与交互经验所蕴含的策略的统一性, 将在很大程度上影响智能体的表现[49]. 环境模型也是在智能体训练过程中不断优化的, 其收敛速度直接导致了智能体用于训练的模拟经验的准确性[79]. 如何设计更准确的模型结构并提升模型的收敛速度是此方向需要长期关注的问题.

2) 经验重塑算法的效率: 在固定奖励函数的情况下, 对智能体自身的交互经验进行重塑以实现经验的大规模增广是经验回放领域近年来的研究热点. 附加经验的准确性和策略一致性保证了收敛后智能体优秀的性能. 在庞大的经验中选择合适的附加目标以期产生高质量的附加经验仍然是此类方法关注的重点. 但毫无疑问, 大规模的搜索和重塑会对算法的计算效率产生影响[88], 使用多线程进行并行计算或设计其他高效的计算框架或许是经验重塑类方法提升效率的有效途径.

References

1 Gao Yang, Chen Shi-Fu, Lu Xin. Research on reinforcement learning technology: A review. *Acta Automatica Sinica*, 2004, **30**(1): 86−100
(高阳, 陈世福, 陆鑫. 强化学习研究综述. 自动化学报, 2004, **30**(1): 86−100)

2 Sutton R S, Barto A G. *Reinforcement Learning: An Introduction*. Cambridge: MIT Press, 1998.

3 Li Chen-Xi, Cao Lei, Zhang Yong-Liang, Chen Xi-Liang, Zhou Yu-Huan, Duan Li-Wen. Knowledge-based deep reinforcement learning: A review. *Systems Engineering and Electronics*, 2017, **39**(11): 2603−2613
(李晨溪, 曹雷, 张永亮, 陈希亮, 周宇欢, 段理文. 基于知识的深度强化学习研究综述. 系统工程与电子技术, 2017, **39**(11): 2603−2613)

4 Bellman R. *Dynamic Programming*. Princeton: Princeton University Press, 1957.

5 Mnih V, Kavukcuoglu K, Silver D, Rusu A A, Veness J, Bellemare M G, et al. Human-level control through deep reinforcement learning. *Nature*, 2015, **518**(7540): 529−533

6 Liu Quan, Zhai Jian-Wei, Zhang Zong-Chang, Zhong Shan, Zhou Qian, Zhang Peng, et al. A survey on deep reinforcement learning. *Chinese Journal of Computers*, 2018, **48**(1): 1−27
(刘全, 翟建伟, 章宗长, 钟珊, 周倩, 章鹏, 等. 深度强化学习综述. 计算机学报, 2018, **48**(1): 1−27)

7 Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing Atari with deep reinforcement learning. arXiv preprint arXiv: 1312.5602, 2013.

8 Cheng Y H, Chen L, Chen C L P, Wang X S. Off-policy deep reinforcement learning based on Steffensen value iteration. *IEEE Transactions on Cognitive and Developmental Systems*, 2021,

13(4): 1023−1032

9 Silver D, Huang A, Maddison C J, Guez A, Sifre L, Driessche G V D, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016, **529**(7587): 484−489

10 Chen P Z, Lu W Q. Deep reinforcement learning based moving object grasping. *Information Sciences*, 2021, **565**: 62−76

11 Jin Z H, Wu J H, Liu A D, Zhang W A, Yu L. Policy-based deep reinforcement learning for visual servoing control of mobile robots with visibility constraints. *IEEE Transactions on Industrial Electronics*, 2022, **69**(2): 1898−1908

12 Li X J, Liu H S, Dong M H. A general framework of motion planning for redundant robot manipulator based on deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 2022, **18**(8): 5253−5263

13 Chen S Y, Wang M L, Song W J, Yang Y, Li Y J, Fu M Y. Stabilization approaches for reinforcement learning-based end-to-end autonomous driving. *IEEE Transactions on Vehicular Technology*, 2020, **69**(5): 4740−4750

14 Qi Q, Zhang L X, Wang J Y, Sun H F, Zhuang Z R, Liao J X, et al. Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 2020, **69**(11): 13861−13874

15 Kiran B R, Sobh I, Talpaert V, Mannion P, Sallab A A A, Yogamani S, et al. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2022, **23**(6): 4909−4926

16 Taghian M, Asadi A, Safabakhsh R. Learning financial asset-specific trading rules via deep reinforcement learning. *Expert Systems With Applications*, 2022, **195**: Article No. 116523

17 Tsantekidis A, Passalis N, Tefas A. Diversity-driven knowledge distillation for financial trading using deep reinforcement learning. *Neural Networks*, 2021, **140**: 193−202

18 Park H, Sim M K, Choi D G. An intelligent financial portfolio trading strategy using deep Q-learning. *Expert Systems With Applications*, 2020, **158**: Article No. 113573

19 Tan W S, Ryan M L. A single site investigation of DRLs for CT head examinations based on indication-based protocols in Ireland. *Journal of Medical Imaging and Radiation Sciences*, DOI: 10.1016/j.jmir.2022.03.114

20 Allahham M S, Abdellatif A A, Mohamed A, Erbad A, Yaacoub E, Guizani M. I-SEE: Intelligent, secure, and energy-efficient techniques for medical data transmission using deep reinforcement learning. *IEEE Internet of Things Journal*, 2021, **8**(8): 6454−6468

21 Lin L J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 1992, **8**: 293−321

22 Bellman R. A Markovian decision process. *Indiana University Mathematics Journal*, 1957, **6**(4): 679−684

23 Rummery G A, Niranjan M. On-line Q-learning Using Connectionist Systems, Technical Report GUED/F-INFENG/TR 166, Engineering Department, Cambridge University, England, 1994.

24 Sutton R, Mcallester D A, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS). Denver, Colorado, USA: MIT Press, 1999. 1057−1063

25 Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992, **8**: 229−256

26 Mnih V, Badia A P, Mirza M, Graves A, Harley T, Lillicrap P T, et al. Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning (ICML). New York, USA: ACM, 2016. 1928−1937

27 Babaeizadeh M, Frosio I, Tyree S, Clemons J, Kautz J. Reinforcement learning through asynchronous advantage actor-critic on a GPU. arXiv preprint arXiv: 1611.06256, 2017.

28 Schulman J, Levine S, Moritz P, Jordan M I, Abbeel P. Trust region policy optimization. arXiv preprint arXiv: 1502.05477, 2015.

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

30 Watkins C J C H, Dayan P. *Q*-learning. *Machine Learning*, 1992, **8**(3): 279−292

31 Hasselt H V, Guez A, Silver D. Deep reinforcement learning with double Q-learning. arXiv preprint arXiv: 1509.06461, 2015.

32 Wang Z Y, Tom S, Matteo H, Hado V H, Marc L, Nando D F. Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning (ICML). New York, USA: ACM, 2016. 1995−2003

33 Lillicrap T P, Hunt J J, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv: 1509.02971, 2015.

34 Fujimoto S, Hoof V H, Meger D. Addressing function approximation error in actor-critic methods. arXiv preprint arXiv: 1802.09477, 2018.

35 Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv: 1801.01290, 2018.

36 Nair A, Srinivasan P, Blackwell S, Alcicek C, Fearon R, Maria A D, et al. Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv: 1507.04296, 2015.

37 Hausknecht M, Stone P. Deep recurrent Q-learning for partially observable MDPs. arXiv preprint arXiv: 1507.06527, 2015.

38 Plappert M, Houthooft R, Dhariwal P, Sidor S, Chen R Y, Chen X, et al. Parameter space noise for exploration. arXiv preprint arXiv: 1706.01905, 2018.

39 Hessel M, Modayil J, Hasselt H V, Schaul T, Ostrovski G, Dabney W, et al. Rainbow: Combining improvements in deep reinforcement learning. arXiv preprint arXiv: 1710.02298, 2017.

40 Liu Jian-Wei, Gao Feng, Luo Xiong-Lin. Survey of deep reinforcement learning based on value policy gradient. *Chinese Journal of Computers*, 2019, **42**(6): 1406−1438
(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)

41 Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, et al. Soft actor-critic algorithms and applications. arXiv preprint arXiv: 1812.05905, 2018.

42 Jang E, Gu S X, Poole B. Categorical reparameterization with Gumbel-Softmax. arXiv preprint arXiv: 1611.01144, 2017.

43 Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. arXiv preprint arXiv: 1511.05952, 2016.

44 Brittain M, Bertram J, Yang X X, Wei P. Prioritized sequence experience replay. arXiv preprint arXiv: 1905.12726, 2019.

45 Lee S, Lee J, Hasuo I. Predictive PER: Balancing priority and diversity towards stable deep reinforcement learning. arXiv preprint arXiv: 2011.13093, 2020.

46 Cao X, Wan H Y, Lin Y F, Han S. High-value prioritized experience replay for off-policy reinforcement learning. In: Proceedings of the IEEE 31st International Conference on Tools With Artificial Intelligence (ICTAI). Portland, OR, USA: IEEE, 2019. 1510−1514

47 Zhao Ying-Nan, Liu Peng, Zhao Wei, Tang Jiang-Long. Twice sampling method in deep Q-network. *Acta Automatica Sinica*, 2019, **45**(10): 1870−1882
(赵英男, 刘鹏, 赵巍, 唐降龙. 深度 Q 学习的二次主动采样方法. 自动化学报, 2019, **45**(10): 1870−1882)

48 Sun P Q, Zhou W G, Li H Q. Attentive experience replay. In:

Proceedings of the 34th AAAI Conference on Artificial Intelligence. New York, USA: AAAI Press, 2020. 5900−5907

49 Hu Z J, Gao X G, Wan K F, Zhai Y W, Wang Q L. Relevant experience learning: A deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments. *Chinese Journal of Aeronautics*, 2021, **34**(12): 187−204

50 Cicek D C, Duran E, Saglam B, Mutlu F B, Kozat S S. Off-policy correction for deep deterministic policy gradient algorithms via batch prioritized experience replay. In: Proceedings of the 33rd IEEE International Conference on Tools With Artificial Intelligence (ICTAI). Washington, DC, USA: IEEE, 2021. 1255−1262

51 Ren Z P, Dong D Y, Li H X, Chen C L. Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2018, **29**(6): 2216−2226

52 Bengio Y, Louradour J, Collobert R, Weston J. Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning (ICML). Montreal, Quebec, Canada: ACM, 2009. 41−48

53 Wang X, Chen Y D, Zhu W W. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022, **44**(9): 4555−4576

54 Hu Z J, Gao X G, Wan K F, Wang Q L, Zhai Y W. Asynchronous curriculum experience replay: A deep reinforcement learning approach for UAV autonomous motion control in unknown dynamic environments. arXiv preprint arXiv: 2207.01251, 2022.

55 Kumar A, Gupta A, Levine S. DisCor: Corrective feedback in reinforcement learning via distribution correction. arXiv preprint arXiv: 2003.07305, 2020.

56 Lee K, Laskin M, Srinivas A, Abbeel P. SUNRISE: A simple unified framework for ensemble learning in deep reinforcement learning. arXiv preprint arXiv: 2007.04938, 2020.

57 Sinha S, Song J M, Garg A, Ermon S. Experience replay with likelihood-free importance weights. arXiv preprint arXiv: 2006.13169, 2020.

58 Liu X H, Xue Z H, Pang J C, Jiang S Y, Xu F, Yu Y. Regret minimization experience replay in off-policy reinforcement learning. arXiv preprint arXiv: 2105.07253, 2021.

59 Zhang S T, Sutton R S. A deeper look at experience replay. arXiv preprint arXiv: 1712.01275, 2018.

60 Novati G, Koumoutsakos P. Remember and forget for experience replay. arXiv preprint arXiv: 1807.05827, 2019.

61 Shi Sheng-Miao, Liu Quan. Deep deterministic policy gradient with classified experience replay. *Acta Automatica Sinica*, 2022, **48**(7): 1816−1823
(时圣苗, 刘全. 采用分类经验回放的深度确定性策略梯度方法. 自动化学报, 2022, **48**(7): 1816−1823)

62 Liu Xiao-Yu, Xu Chi, Zeng Peng, Yu Hai-Bin. Deep reinforcement learning-based high concurrent computing offloading for heterogeneous industrial tasks. *Chinese Journal of Computers*, 2021, **44**(12): 2367−2380
(刘晓宇, 许驰, 曾鹏, 于海斌. 面向异构工业任务高并发计算卸载的深度强化学习方法. 计算机学报, 2021, **44**(12): 2367−2380)

63 Zhu Fei, Wu Wen, Fu Yu-Chen, Liu Quan. A dual deep network based secure deep reinforcement learning method. *Chinese Journal of Computers*, 2019, **42**(8): 1812−1826
(朱斐, 吴文, 伏玉琛, 刘全. 基于双深度网络的安全深度强化学习方法. 计算机学报, 2019, **42**(8): 1812−1826)

64 Wei Q, Ma H L, Chen C L, Dong D Y. Deep reinforcement learning with quantum-inspired experience replay. *IEEE Transactions on Cybernetics*, 2022, **52**(9): 9326−9338

65 Li Y J, Aghvami A H, Dong D Y. Path planning for cellular-connected UAV: A DRL solution with quantum-inspired experience replay. *IEEE Transactions on Wireless Communications*, 2022, **21**(10): 7897−7912

66 Chen X C, Yao L N, Wang X Z, McAuley J. Locality-sensitive experience replay for online recommendation. arXiv preprint arXiv: 2110.10850, 2021.

67 Bruin T D, Kober J, Tuyls K, Babuska R. Improved deep reinforcement learning for robotics through distribution-based experience retention. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Daejeon, South Korea: IEEE, 2016. 3947−3952

68 Li M Y, Kazemi A, Laguna A F, Hu X S. Associative memory based experience replay for deep reinforcement learning. arXiv preprint arXiv: 2207.07791, 2022.

69 Schaal S. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 1999, **3**(6): 233−242

70 Attia A, Dayan S. Global overview of imitation learning. arXiv preprint arXiv: 1801.06503, 2018.

71 Hester T, Vecerik M, Pietquin O, Lanctot M, Schaul T, Piot B, et al. Deep Q-learning from demonstrations. arXiv preprint arXiv: 1704.03732, 2017.

72 Vecerik M, Hester T, Scholz J, Wang F M, Pietquin O, Piot B, et al. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. arXiv preprint arXiv: 1707.08817, 2017.

73 Guillen-Perez A, Cano M. Learning from Oracle demonstrations —— A new approach to develop autonomous intersection management control algorithms based on multiagent deep reinforcement learning. *IEEE Access*, 2022, **10**: 53601−53613

74 Huang Z Y, Wu J D, Lv C. Efficient deep reinforcement learning with imitative expert priors for autonomous driving. *IEEE Transactions on Neural Networks and Learning Systems*, DOI: 10.1109/TNNLS.2022.3142822

75 Hu Z J, Wan K F, Gao X G, Zhai Y W, Wang Q L. Deep reinforcement learning approach with multiple experience pools for UAV′s autonomous motion planning in complex unknown environments. *Sensors*, 2020, **20**(7): Article No. 1890

76 Wan K F, Wu D W, Li B, Gao X G, Hu Z J, Chen D Q. ME-MADDPG: An efficient learning-based motion planning method for multiple agents in complex environments. *International Journal of Intelligent Systems*, 2022, **37**(3): 2393−2427

77 Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, CA, USA: Curran Associates Inc., 2017. 6382−6393

78 Zhang T Z, Miao X H, Li Y B, Jia L, Zhuang Y H. AUV surfacing control with adversarial attack against DLaaS framework. *IEEE Transactions on Computers*, DOI: 10.1109/TC.2021.3072072

79 Sutton R S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: Proceedings of the 7th International Conference on Machine Learning (ICML). Austin, Texas, USA: ACM, 1990. 216−224

80 Silver D, Sutton R S, Müller M. Sample-based learning and search with permanent and transient memories. In: Proceedings of the 25th International Conference on Machine Learning (ICML). Helsinki, Finland: ACM, 2008. 968−975

81 Santos M, Jose A, Lopez V, Botella G. Dyna-*H*: A heuristic planning reinforcement learning algorithm applied to role-playing-game strategy decision systems. *Knowledge-Based Systems*, 2012, **32**: 28−36

82 Pan Y C, Yao H S, Farahmand A, White M. Hill climbing on value estimates for search-control in Dyna. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI). Macao, China: AAAI Press, 2019. 3209−3215

83 Pan Y C, Zaheer M, White A, Patterson A, White M. Organizing experience: A deeper look at replay mechanisms for sample-based planning in continuous state domains. In: Proceedings of the 27th International Joint Conference on Artificial Intelli-

gence (IJCAI). Stockholm, Sweden: AAAI Press, 2018. 4794–4800

84 Andrychowicz M, Wolski F, Ray A, Schneider J, Fong R, Welinder P, et al. Hindsight experience replay. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS). Long Beach, CA, USA: Curran Associates Inc., 2017. 5055–5065

85 Schaul T, Horgan D, Gregor K, Silver D. Universal value function approximators. In: Proceedings of the 32nd International Conference on Machine Learning (ICML). Lille, France: JMLR.org, 2015. 1312–1320

86 Luu T M, Yoo C D. Hindsight goal ranking on replay buffer for sparse reward environment. *IEEE Access*, 2021, **9**: 51996–52007

87 Fang M, Zhou C, Shi B, Gong B Q, Xu J, Zhang T. DHER: Hindsight experience replay for dynamic goals. In: Proceedings of the 7th International Conference on Learning Representations (ICLR). New Orleans, LA, USA: OpenReview.net, 2019. 1–12

88 Hu Z J, Gao X G, Wan K F, Evgeny N, Li J L. Imaginary filtered hindsight experience replay for UAV tracking dynamic targets in large-scale unknown environments. *Chinese Journal of Aeronautics*, DOI: 10.1016/j.cja.2022.09.008

89 Fang M, Zhou T Y, Du Y L, Han L, Zhang Z Y. Curriculum-guided hindsight experience replay. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS). Vancouver, BC, Canada: MIT Press, 2019. 12623–12634

90 Yang R, Fang M, Han L, Du Y L, Luo F, Li X. MHER: Model-based hindsight experience replay. arXiv preprint arXiv: 2107.00306, 2021.

胡子剑 西北工业大学电子信息学院博士研究生. 2018 年获得西北工业大学探测制导与控制技术学士学位. 主要研究方向为强化学习理论与应用.
E-mail: huzijian@mail.nwpu.edu.cn
(**HU Zi-Jian** Ph.D. candidate at the School of Electronics and Information, Northwestern Polytechnical University. He received his bachelor degree in detection guidance and control technology from Northwestern Polytechnical University in 2018. His research interest covers reinforcement learning theory and applications.)

高晓光 西北工业大学电子信息学院教授. 1989 年获得西北工业大学系统工程博士学位. 主要研究方向为机器学习理论, 贝叶斯网络理论和多智能体控制应用.
E-mail: cxg2012@nwpu.edu.cn
(**GAO Xiao-Guang** Professor at the School of Electronics and Information, Northwestern Polytechnical University. She received her Ph.D. degree in system engineering from Northwestern Polytechnical University in 1989. Her research interest covers machine learning theory, Bayesian network theory, and multi-agent control application.)

万开方 西北工业大学电子信息学院副研究员. 2016 年获得西北工业大学系统工程博士学位. 主要研究方向为多智能体理论, 近似动态规划和强化学习. 本文通信作者.
E-mail: wankaifang@nwpu.edu.cn
(**WAN Kai-Fang** Associate researcher at the School of Electronics and Information, Northwestern Polytechnical University. He received his Ph.D. degree in system engineering from Northwestern Polytechnical University in 2016. His research interest covers multi-agent theory, approximate dynamic programming, and reinforcement learning. Corresponding author of this paper.)

张乐天 西安电子科技大学外国语学院硕士研究生. 主要研究方向为科技翻译, 翻译理论和机器翻译.
E-mail: 22091213382@stu.xidian.edu.cn
(**ZHANG Le-Tian** Master student at the School of Foreign Languages, Xidian University. Her research interest covers scientific translation, translation theory, and machine translation.)

汪强龙 西北工业大学电子信息学院博士研究生. 主要研究方向为深度学习, 强化学习.
E-mail: wql1995@mail.nwpu.edu.cn
(**WANG Qiang-Long** Ph.D. candidate at the School of Electronics and Information, Northwestern Polytechnical University. His research interest covers deep learning and reinforcement learning.)

NERETIN Evgeny 莫斯科航空学院教授. 2011 年获得莫斯科航空学院技术科学博士学位. 主要研究方向为航空电子, 智能决策.
E-mail: evgeny.neretin@gmail.com
(**NERETIN Evgeny** Professor of Moscow Aviation Institute. He received his Ph.D. degree in technical sciences from Moscow Aviation Institute in 2011. His research interest covers avionics and intelligent decision-making.)