竞争与合作视角下的多 Agent 强化学习研究进展

田小禾[1,2,3,4] 李 伟[1,2,3,4] 许 铮[5] 刘天星[1,2,3,4] 戚骁亚[1,2,3,4,5] 甘中学[1,2,3,4*]

[1](复旦大学工程与应用技术研究院 上海 200433)

[2](上海智能机器人工程技术研究中心 上海 200433)

[3](智能机器人教育部工程研究中心 上海 200433)

[4](季华实验室 广东 佛山 528000)

[5](北京深度奇点科技有限公司 北京 100089)

摘 要 随着深度学习和强化学习研究取得长足的进展,多 Agent 强化学习已成为解决大规模复杂序贯决策问题的通用方法。为了推动该领域的发展,从竞争与合作的视角收集并总结近期相关的研究成果。该文介绍单 Agent 强化学习;分别介绍多 Agent 强化学习的基本理论框架——马尔可夫博弈以及扩展式博弈,并重点阐述了其在竞争、合作和混合三种场景下经典算法及其近期研究进展;讨论多 Agent 强化学习面临的核心挑战——环境的不稳定性,并通过一个例子对其解决思路进行总结与展望。

关键词 深度学习 强化学习 多 Agent 强化学习 环境的不稳定性

中图分类号 TP3 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.04.001

RECENT PROCESS AND PROSPECT OF MULTI-AGENT REINFORCEMENT LEARNING UNDER THE PERSPECTIVE OF COMPETITION AND COOPERATION

Tian Xiaohe[1,2,3,4] Li Wei[1,2,3,4] Xu Zheng[5] Liu Tianxing[1,2,3,4] Qi Xiaoya[1,2,3,4,5] Gan Zhongxue[1,2,3,4*]

[1](*Academy for Engineering and Technology, Fudan University, Shanghai 200433, China*)

[2](*Shanghai Engineering Research Center of AI & Robotics, Shanghai 200433, China*)

[3](*Engineering Research Center of AI & Robotics, Ministry of Education, Shanghai 200433, China*)

[4](*Ji Hua Laboratory, Foshan 528000, Guangdong, China*)

[5](*Beijing Deep Singularity Technology Co., Ltd., Beijing 100089, China*)

Abstract With the rapid development of deep learning and reinforcement learning, multi-agent reinforcement learning (MARL) has become a common approach to solve the large scale complex sequential decision-making problem. In order to promote the development of this field, this paper collects and reviews recent research results from the perspective of competition and cooperation. This paper introduced deep reinforcement learning and introduced the basic theoretical framework of MARL-Markov game and extensive game, and especially emphasized the reinforcement learning algorithms developed recently in three scenarios of competition, cooperation and mixture. This paper discussed the core challenge of MARL that was non-stationary of the environment, and an example was given to summarize and prospect its solutions.

Keywords Deep learning Reinforcement learning Multi-agent reinforcement learning Non-stationary of the environment

---

收稿日期:2021-01-01。广东省季华实验室基金项目(X190021TB190);上海市科学技术委员会项目(1951113200)。**田小禾**,硕士生,主研领域:强化学习,智能机器人。**李伟**,博士。**许铮**,博士。**刘天星**,硕士生。**戚骁亚**,博士。**甘中学**,教授。

0 引 言

随着深度学习(Deep Learning,DL)发展,应用于序贯决策领域的强化学习(Reinforcement Learning,RL)研究尤其是深度强化学习(Deep Reinforcement Learning,DRL)方面取得了长足的进步。

强化学习主要研究的是,Agent 怎样在与环境交互的过程中逐渐学会决策,以获取尽可能高的总奖励值。这种在试错中不断学习和进化的方法与人类的认知发育是很相似的。传统的强化学习饱受维度困扰且难以泛化,基于神经网络的深度强化学习算法有效地解决了这个问题,被广泛运用于单 Agent 系统中最优决策求解[1-4]。

在现实世界中,由于对象系统的复杂性,多 Agent 系统是更加符合实际的建模方式,也是一种分解复杂性问题的解决方案。在多 Agent 系统中,单个 Agent 的总奖励函数不仅与自身决策有关,还与所有 Agent 的联合决策有关,这时仅依靠单 Agent 系统中的 DRL 算法无法实时协调多个 Agent。因此,近年来涌现出大批多 Agent 的强化学习研究工作来克服这方面的困难。

多 Agent 强化学习(multi-agent reinforcement learning,MARL)的研究核心在于实时协调多个 Agent 通信、合作或者竞争关系,并进行资源分配、任务调度等,从而智能地完成特定的系统任务。作为当前 AI 领域的研究热点,MARL 已经在许多领域(例如围棋[5]、扑克[6]、Dota2[7]、StarCraft2[8]和多机器人控制[9])有了超越人类的表现。但是,它也面临着许多挑战,例如采样困难、奖励分配、环境的不稳定性等,理论上仍留有一些值得深入的空白。

作为大规模复杂序贯决策领域的主要解决方法,MARL 不仅有助于研究多 Agent 系统中群体智慧的涌现,更是实现通用人工智能必不可少的智慧工具。目前国内外对 MARL 相关研究工作的综述主要是从分层强化学习[10]、Agent 间通联方式[11]等角度出发,从竞争与合作的视角进行梳理是非常稀缺的,因此,本文从这个视角对近年来 MARL 的最新理论成果进行综述。

1 单 Agent 强化学习

1.1 强化学习

强化学习通常使用马尔可夫决策过程(Markov decision process,MDP)对 Agent 与环境的交互进行建模[12],目标是使 Agent 在交互过程中得到最大的总奖励值。MDP 由五元组构成:$<S,A,T,R,\gamma>$,其中 S 代表状态空间;A 代表动作空间;T 为状态转移概率:$S\times A\times S\rightarrow[0,1]$,代表在任意状态 $s\in S$,采用任意可能动作 $a\in A$,转移到状态 $s'\in S$ 的概率;R 为奖励函数:$S\times A\times S\rightarrow\mathbb{R}$,代表在任意状态 $s\in S$,采用任意可能动作 $a\in A$,转移到状态 s' 所获得的即时奖励 r;γ 为奖励的折扣系数,反映了 Agent 对于未来回报的重视程度。

具体而言,在每个时刻 t,Agent 首先观测环境的状态 $s_t\in S$,随后基于策略 $\pi(s_t)$ 选择动作 a_t,导致环境状态转变到 s_{t+1},同时 Agent 收到环境返还的即时奖励 r_t;Agent 再依据新的环境状态和策略做出新的决策,该交互过程不断重复。

解决 MDP 将会构建一个从状态集向动作集的映射,即策略 $\pi:S\rightarrow A$,最终得到的最优策略 π^* 会最大化总奖励值 $\mathbb{E}\left[\sum_{t=0}^{N}\gamma^t r_t\right]$(假设共有 N 步决策)。

求解 MDP 通常有基于值迭代和策略迭代的方式,以下是这两种方法的介绍。

1.1.1 值迭代方法

值迭代方法的核心是采用函数近似的方法去维护动作值函数。其中的典型算法是 Q-learning 算法,常在环境稳定且完全可知、离散动作空间的单 Agent 系统中使用。Q-learning 算法的关键在于维护一个最优 Q^* 函数的近似估计值 $\hat{Q}(s,a)$,Q^* 代表 Agent 在状态 s 下执行动作 a 之后持续采用最佳策略所获得的总奖励值。$\hat{Q}(s,a)$ 按如下方式进行更新:

$$\hat{Q}(s,a)\leftarrow\hat{Q}(s,a)+\alpha[(r+\gamma\max_{a'}\hat{Q}(s',a'))-\hat{Q}(s,a)] \tag{1}$$

式中:r 是在状态 s 采用动作 a 后转移到状态 s' 所获得的即时奖励,α 为学习率且满足 $\alpha\in[0,1]$。

当状态和动作空间是离散且有限的,学习率的总和趋于无穷大(以确保每个状态-动作对被无限次地访问),且学习率的平方和是有限的(确保收敛的概率是 1),Q-learning 算法被证明收敛于 Q^*[13]。

1.1.2 策略迭代方法

Q-learning 算法通过查表方式来估计给定输入状态下的动作值函数,只适用于离散动作空间。本节介绍的策略迭代算法利用策略梯度去学习最优策略函数,不仅可用于离散动作空间,还可以用于连续动作空间。

REINFORCE 算法[14]是非常经典的策略迭代方法。由蒙特卡洛方法采样来估计全轨迹总奖励值,再按如下方式利用策略梯度去更新策略参数 θ:

$$\theta_{t+1}=\theta_t+\alpha G_t\frac{\nabla\pi(A_t;S_t,\theta_t)}{\pi(A_t;S_t,\theta_t)} \tag{2}$$

式中:G_t 是总奖励值,α 为学习率,$A_t\sim\pi$。

REINFORCE 算法的主要局限是方差较高,可以考虑减去状态的基线值,来弥补其高方差的问题[15]。基线可以是任何与状态有关而与动作无关的函数(从而保证求和平均值为 0),较为常用的基线是状态值函数 $V(s)$。更进一步,可引出 Actor-Critic 算法[16],其中 Actor 代表策略函数,Critic 代表值函数;越好的策略函数可以获得越高的总奖励值,越准确的值函数可以更好地指导策略函数更新。在 Actor-Critic 算法中,将状态-动作值与状态值相减可以得到优势函数 $\hat{A}_t$,用它去代替 G_t不仅可以降低方差,在采用蒙特卡洛采样时还可以做到无偏估计。此时的策略梯度估计如下:

$$\hat{g}=\hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\hat{A}_t] \tag{3}$$

Actor-Critic 算法中的一个例子是确定策略梯度算法(Deterministic Policy Gradient, DPG)[15],它通过 Q-learning 算法去学习 critic,在 DRL 和 MDRL 中均有广泛的扩展。

1.2　深度强化学习

传统的强化学习方法饱受维度困扰,比如:在大的状态空间中学习非常缓慢,无法在状态空间中进行泛化,状态表征需要手动精细设计等。深度强化学习通过使用神经网络去充当函数估计器,既有助于在大的状态空间中进行泛化、提高采样效率,还可以避免手动进行状态表征的麻烦[17]。

深度学习中往往要求训练数据是独立同分布的稳定数据。然而强化学习中的训练数据(即 Agent 与环境的交互记录)往往是高关联性的序列化数据,而且当 Agent 探索状态空间的不同部分时,会导致数据分布不稳定。接下来简要回顾基于值、基于策略的算法是怎样解决这些问题的。

1.2.1　基于值的方法

DQN(deep Q-networks)算法[18]是深度学习与 Q-learning 的结合,它采用深度神经网络来估计状态-动作值函数。为了消除交互数据对的高度关联性,DQN 使用经验回放池去存储交互数据对 $<s,a,r,s'>$,在训练时从池中抽样。为了稳定学习过程并消除数据分布的不稳定性,除了在线网络中使用参数 θ,DQN 在目标网络中使用神经网络参数的复制版本 θ^-,θ^-会阶段性地被设置为 θ。DQN 在每轮迭代训练(i)中以最小化在线网络和目标网络的差异为目标,其损失函数如下:

$$L_i(\theta_i)=E_{s,a,r,s'}[(r+\gamma \max_{a'} Q(s',a';\theta_i^-)-Q(s,a;\theta_i))^2] \tag{4}$$

式中:数据对 $<s,a,r,s'>$是从经验回放池中抽样得到的。

DQN 也有许多的扩展版本,例如在 Double DQN 中使用两个函数估计器去降低过估计偏差[2];在 dueling-DQN 中使用两个分支,分别用来估计状态值和优势值,在最后一层将两个分支结合去计算 Q 值[19]。

DQN 的成功收敛依赖于经验回放池的运用。当环境部分可知时,即时奖励和下一个状态不仅依赖于当前的输入,还依赖于历史及其他因素时,DQN 的效果会受到很大的影响。针对这个问题,也有以下的工作可供参考:深度循环 Q 网络(DRQN)[20]使用 LSTMs 去进行值估计;还可以使用有限状态控制器[21],以便依据完整的历史观测去采取动作。

1.2.2　基于策略梯度的方法

近期,通过使用深度神经网络来估计 Actor-Critic 算法中的策略函数和值函数,诸如 DDPG[1]、TRPO[22]、PPO[4]和 A3C[3]等基于策略梯度的方法被广泛应用在动作空间连续且维度较高的任务中,且均有很好的表现。

这里主要介绍 PPO 算法。Actor-Critic 算法有一个问题是 Agent 与环境交互产生的数据只能用于本轮策略的更新,用完即弃。为了可以离线更新策略,考虑采用重要性采样技术,此时的目标函数如下:

$$J^{\theta_{\text{old}}}(\theta)=\hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}A^{\theta_{\text{old}}}(s_t,a_t)\right] \tag{5}$$

式中:$\pi_{\theta_{\text{old}}}$为用于产生交互数据的行为策略,$\pi_\theta$为需要被优化的目标策略,$A^{\theta_{\text{old}}}(s_t,a_t)$为采用行为策略所得到的优势函数值。为了防止策略函数大幅度更新,即防止 $\pi_{\theta_{\text{old}}}$与 π_θ差距过大,PPO 算法提出目标函数如下:

$$J_{\text{PPO}}^{\theta_{\text{old}}}=\hat{\mathbb{E}}_t\left[\min\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}A^{\theta_{\text{old}}}(s_t,a_t),\ clip\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)},1-\varepsilon,1+\varepsilon\right)A^{\theta_{\text{old}}}(s_t,a_t)\right)\right] \tag{6}$$

式中:ε 为超参数,设置为 0.2。通过对概率比例 $\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$进行裁剪,防止其超出设置范围[$1-\varepsilon$, $1+\varepsilon$]。

2　多 Agent 强化学习

在处理实际的大规模复杂序贯决策问题时,常以多 Agent 的形式来建模。一方面,多 Agent 建模的方式是系统中自然的选择,另一方面,多 Agent 建模有助于将大规模问题分解为单 Agent 系统中已解决的问题。而单 Agent 强化学习算法无法处理多个决策者之间的交互问题。因此,为研究多 Agent 之间相互交互问题如竞争、合作以及通信问题,在 DRL 的基础上,延伸出多 Agent 强化学习(Multi-agent Reinforcement Learning,

MARL)。

以系统任务来分,MARL 无外乎三种方式,分别为竞争、合作或者是混合模式。近年来,MARL 在这三方面的算法研究上都取得了很大的进步。接下来首先对多 Agent 强化学习的两种理论框架进行介绍,然后分别总结竞争、合作和混合模式下的 MARL 近期工作。

2.1 多 Agent 强化学习框架

相比于单 Agent 系统,多 Agent 系统不仅包含环境的变化,还包含多个 Agent 之间的交互动力学过程。从而使得单个 Agent 学习受到所有 Agent 的联合决策的影响。更确切地说,每个 Agent 都需要优化各自的长期奖励值;而这个长期奖励值,不只依赖个体策略,且依赖其他所有 Agent 的策略。

MARL 主要有以下两种理论框架:马尔可夫博弈(Markov Game,MG)和扩展式博弈(Extensive Game,EG),分别适用于完美信息环境和不完美信息环境的建模。在 MG 中,观测到系统状态 s 后,所有 Agent 同时执行自己的动作 a^i,并获得各自的即时奖励 r^i。而在 EG 中,Agent 交替地执行自己的动作 a^i,在博弈结束时获得各自的奖励值 $r^i(z)$ 是终止状态。下面分别对这两种理论框架进行详细介绍。

2.1.1 马尔可夫博弈

与单 Agent 的 MDP 建模过程类似,可以定义多 Agent 的马尔可夫博弈(Markov game,MG),如图 1 所示。不同的是在观测到状态 s 后,每个 Agent 同时执行自己的动作 a^i,并获得各自的即时奖励 r^i。MG[23] 由六元组构成:$(N,S,\{A^i\}_{i\in N},P,\{R^i\}_{i\in N},\gamma)$,其中:$N$ 代表 Agent 数目;S 代表所有 Agent 观测到的状态空间;A^i 为 Agent i 的动作空间,且可以定义联合动作空间 $A:=A^1\times A^2\times\cdots\times A^N$;$P$ 为状态转移概率,满足 $P:S\times A\to\Delta(S)$,代表状态 $s\in S$ 采取联合动作 $a\in A$,向下一个状态 $s'\in S$ 的转移概率;$R^i:S\times A\times S\to R$ 是奖励函数,代表 Agent i 从 (s,a) 转移到 s' 所获得的即时奖励;$\gamma\in[0,1]$ 定义为折扣系数。

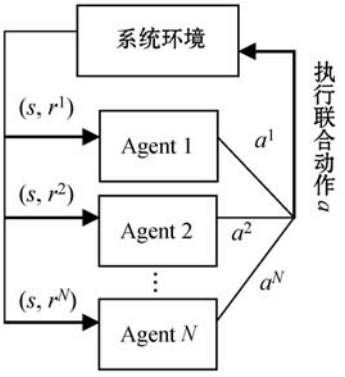

图 1 马尔可夫博弈过程示意图

整个交互过程可以归纳为:在 t 时刻,Agent i 在观测到状态 s_t 后,采取动作 a_t^i;系统接下来转换到状态 s_{t+1},Agent i 获得即时奖励 $R^i(s_t,a_t,s_{t+1})$。而对于 Agent i 来说,其目标是寻找一个策略 $\pi^i:S\to\Delta(A^i)$(即:$a_t^i\sim\pi^i(\cdot\mid s_t)$),使得自己的总奖励值最大。与单 Agent 系统不同,这里 Agent 的值函数 $V^i:S\to\boldsymbol{R}$ 是所有 Agent 联合策略 $\pi:S\to\Delta(A)$(定义为 $\pi(a\mid s):=\prod_{i\in N}\pi^i(a^i\mid s)$)的函数。对于任意联合策略 π 和状态 $s\in S$,值函数可定义为:

$$V^i_{\pi^i,\pi^{-i}}(s):=E\Big[\sum_{t\geqslant 0}\gamma^t R^i(s_t,a_t,s_{t+1})\mid a_t^i\sim\pi^i(\cdot\mid s_t),s_0=s\Big] \tag{7}$$

式中:$-i$ 代表除了 Agent i 的所有其他 Agent。由此看出,仅依据单 Agent 系统 MDP 的解决方案去求解 MG 是不合理的。这是因为每个 Agent 的最优表现不仅由它自身策略所控制,还受到所有其他 Agent 的策略所影响。在不同类型的系统任务中(竞争/协作/混合),MG 的求解问题通常会转换为求解纳什均衡问题[24],可做如下定义:$\mathrm{MG}(N,S,\{A^i\}_{i\in N},P,\{R^i\}_{i\in N},\gamma)$ 的纳什均衡是对任意 $s\in S$ 和 $i\in N$ 均满足以下条件的一个联合策略 $\pi^*=(\pi^{1,*},\pi^{2,*},\cdots,\pi^{N,*})$:

$$V^i_{\pi^{i,*},\pi^{-i,*}}(s)\geqslant V^i_{\pi^i,\pi^{-i,*}}(s)\quad\text{for any }\pi^i \tag{8}$$

纳什均衡是一个任何 Agent 都没有动力去偏移的均衡点 π^*,即对于任意 Agent $i\in N$,其自身策略 $\pi^{i,*}$ 均为其余 Agent 策略 $\pi^{-i,*}$ 的最佳响应策略。在 MARL 中,纳什均衡往往是一个标准的学习目标,需要注意的是,纳什均衡点往往不唯一。

(1) 合作模式的 MG。在完全合作的环境设定中,所有 Agent 的奖励函数是一致的,即每个 Agent 的状态值函数和状态-动作值函数均相同[25]。因此可将它们视为一个 Agent,并应用单 Agent 强化学习算法,使得全局最优合作状态达到纳什均衡。

另一种合作模型是平均奖励模型,即总奖励函数为所有 Agent 奖励函数的平均值[26]。它允许 Agent 之间存在差异性,还可以保护隐私,并促进分布式 MARL 算法的发展。

(2) 竞争模式的 MG。完全竞争的 MARL 一般被建模为马尔可夫零和博弈。为了便于算法的设计和分析,大部分文章更关注于双 Agent 之间的竞争[27-28],其中一个 Agent 的奖励正好是另一个的损失。

(3) 混合模式的 MG。混合设定也被称作一般和博弈,对 Agent 的学习目标和关系没有任何限制。其假定每个 Agent 总是理性且自私的,即使他们的奖励可能与他人的利益冲突。博弈论中的均衡解决方案概

念,例如纳什均衡,对为此通用设置开发的算法具有最重要的影响。

2.1.2　扩展式博弈

MG 通常假设环境是完全已知的(即完美信息环境),指的是每个 Agent 对整个系统的全局状态和动作均有清晰地认识。但是实际问题中,环境往往是部分可知的(即不完美信息环境),即环境的全局信息对于每个 Agent 都有所隐藏。在 MG 的基础上,扩展式博弈(Extensive game,EG)放松了对环境的要求,常用于不完美信息环境中建模。

在图 2 的双 Agent 简化版扩展式博弈中,Agent 交替地执行自己的动作 a^i,在博弈结束时获得各自的奖励值 $r^i(z)$是终止状态)。在不完美信息的环境中,Agent2 是不清楚自己究竟处于博弈中的哪个具体位置的,这就导致信息集的多样性。

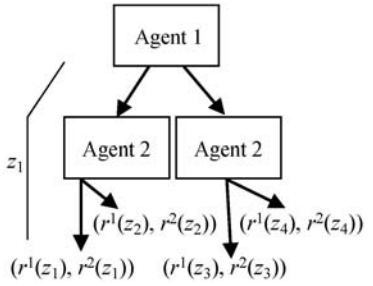

图 2　双 Agent 简化版扩展式博弈示意图

扩展式博弈定义为元组 $(N \cup \{c\}, H, Z, A, \{R^i\}_{i\in N}, \tau, \pi^c, S)$。其中 N 代表 Agent 数目,c 是一个特殊的 Agent——“机会”,它有一个固定的用来指定环境随机性的随机策略;A 是 Agent 们所有可能的动作;H 是所有可能的历史记录,每条历史都是从博弈起始点开始的动作序列的记录。令 $A(h) = \{a \mid ha \in H\}$ 为非终止历史记录 h 后可用的一组动作。假设给定历史 $h \in H$,Agent 采取动作 $a \in A(h)$,产生了新的历史记录 $ha \in H$。Z 是代表博弈结束的终止历史记录的子集。Agent i 的奖励 R^i被定义为在终止状态的一个映射函数:$Z \to \boldsymbol{R}$。标识函数 $\tau: HN \cup \{c\}$,指定在每个历史记录上哪个 Agent 采取行动;如果 $\tau(h) = c$,机会智 Agent 依据它的策略 π^c采取动作 a。S 是 H 的一个分区,被定义为信息集,即对于任意 $s \in S$,任意 $h, h' \in S$,有:$\tau(h) = \tau(h')$和 $A(h) = A(h')$。换句话说,处于相同分区的历史记录 h, h'对于将要采取行动的 Agent $\tau(h)$来讲是不可区分的。直觉上,不完美信息在扩展式博弈中主要表现为 Agent 无法区分处于相同信息集的历史记录。

将多个 Agent 的联合策略定义为 $\pi = (\pi^1, \pi^2, \cdots, \pi^N)$,其中 $\pi^i: S^i \to \Delta(A(s))$是 Agent i 的策略。对于任意历史记录 h 和任意联合策略 π,在 π 下 h 的到达概率定义如下:

$$\eta_\pi(h) = \prod_{h';h'a\subseteq h} \pi^{\tau(h')}(a \mid I(h')) \tag{9}$$

同理,联合策略 π 下信息状态 s 的到达概率定义为:$\eta_\pi(s) = \sum_{h\in s} \eta_\pi(h)$。因此,Agent i 的奖励定义为:$\sum_{z\in Z} \eta_\pi(z) \cdot R^i(z)$。

除了纳什均衡,EG 中还有一种解决理念,ε-纳什均衡,定义如下:

$$R^i(\pi^{i,*}, \pi^{-i,*}) \geqslant R^i(\pi^i, \pi^{-i,*}) - \varepsilon \quad \text{for any policy } \pi^i \text{ of } agent\ i \tag{10}$$

2.2　竞争场景

无论竞争、合作还是混合模式下的 MARL,都可以划分为基于值的方法和基于策略的方法两个大类。它们分别可以视为单体强化学习中基于值的方法和基于策略的方法在多 Agent 环境中的拓展。本文先以竞争场景为主,按照这两个大类来详细说明多 Agent 系统相比于单 Agent 系统在算法上的差别。

大多数竞争场景下,都采用了分布式训练结构,如图 3 所示。

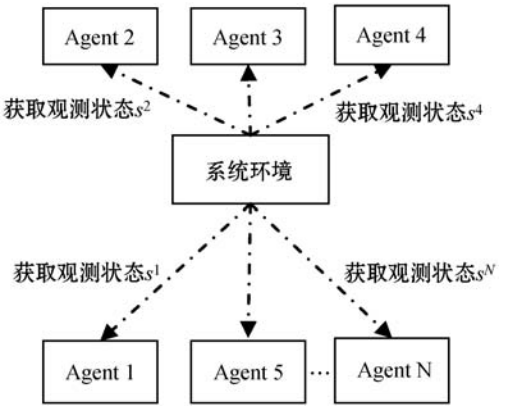

图 3　分布式训练的 MARL 结构示意图

为了简化研究,竞争场景的研究主要以双 Agent 零和博弈为主。所以以下的介绍也主要以双 Agent 零和博弈为例。

2.2.1　基于值的方法

与单 Agent 系统相似,竞争场景下基于值的 MARL 旨在找到最优值函数,进而从中提取均衡策略。

在完美信息的竞争环境中,采用零和 MG 建模。其中两个 Agent 的状态值函数满足 $V^1_{\pi^1,\pi^2} = -V^2_{\pi^1,\pi^2}$,因此任意纳什均衡策略 $\pi^* = (\pi^{1,*}, \pi^{2,*})$均满足:

$$V^1_{\pi^1,\pi^{2,*}}(s) \leqslant V^1_{\pi^{1,*},\pi^{2,*}}(s) \leqslant V^1_{\pi^{1,*},\pi^2}(s) \quad \text{for any } \pi = (\pi^1, \pi^2) \text{ and } s \in S \tag{11}$$

minimax-Q learning[29]可以看作基于 Q-learning 的扩展,它旨在最大化自身的总奖励值,同时尽可能最小化对手的总奖励值。其定义最佳状态值函数 $V^*: S \to$

$\boldsymbol{R}$ 如下：

$$V^* = \max_{\pi^1} m \in {}_{\pi^2} V^1_{\pi^1,\pi^2} = \min_{\pi^2} \max_{\pi^1} V^1_{\pi^1,\pi^2} \quad (12)$$

对于任意 $V: S \rightarrow \boldsymbol{R}$ 和任意状态 $s \in S$，定义动作值函数：

$$Q_V(s,a_1,a_2) = E_{s' \sim P(\cdot|s,a_1,a_2)}[R^1(s,a_1,a_2,s') + \gamma \cdot V(s')] \quad (13)$$

使用线性函数估计和时间差分更新的 minimax-Q learning 被证明收敛于纳什均衡[32]。类似的线性函数估计方法也被运用在具有连续状态-动作空间的零和 MG 中[30-31]。Jia 等[33]研究了一种简化版的零和 MG——零和的基于轮的随机博弈（turn-based stochastic games，TBSG），其环境转移模型可由特征空间进行编码。Sidford 等[34]进一步将适用于 MDP 的 Q-learning 算法扩展到通用模型的零和 TBSG，可达到近乎最优的采样复杂度。

在不完美信息的竞争环境中，采用零和 EG 建模。Buter 等[35]通过将状态空间转换到信念状态空间，使用动态规划来求解零和 EG；Von 等[36]通过序列规范化表征将 EG 转换为随机博弈（stochastic games，SG），然后通过线性规划来求解。这两种方法的局限性在于都只适用于较小规模的问题；参考深度强化学习在大规模完美信息决策问题——围棋博弈中所取得的成功[5]，使用 UCB 标准进行动作选择的蒙特卡洛树搜索算法（Monte Carlo Tree Search，MCTS）[37]也被运用在不完美信息下的双 Agent 多轮博弈中[38]。MCTS 算法结合了随机模拟的一般性和树搜索的准确性，通过将 UCB 标准扩展到 minimax-Q learning，在寻找最优选择时可以在探索访问次数较少的节点（exploration）与利用胜率较高的节点（exploitation）之间进行平衡。MCTS 算法构建的搜索树以非对称的适应搜索空间拓扑结构的形式进行增长，它会更频繁地访问更“有趣”的节点，并将搜索聚焦于更“有意义”的部分。

2.2.2 基于策略的方法

竞争场景下基于策略的 MARL 旨在通过最小化遗憾值来直接求解均衡策略。

任何 Hannan 一致性算法都可以与自对弈结合应用于双人零和博弈中纳什均衡的求解[39-40]。而所谓 Hannan 一致性算法，都具有以下两个属性：首先，当其他 Agent 采用固定策略时，该算法构造的平均策略将收敛到（针对其他 Agent 固定策略的）最佳响应策略。其次，在双人零和博弈中，当两个玩家都采用 Hannan 一致性算法且其平均总遗憾值不超过 2ε 时，其平均策略组合构成了 2ε-纳什均衡。

应用于双人零和博弈中均衡策略求解的 Hannan 一致性算法大致可以分为两类：虚拟对弈（Fictitious Play，FP）[41]和反事实遗憾最小化（Counterfactual Regret Minimization，CFR）[40]。

虚拟对弈起源于博弈论，其主旨是玩家反复在参与博弈的过程中，不断针对对手的平均策略做出最佳响应；最终玩家的平均策略将收敛到纳什均衡。它最初应用在 MG 求解中，被证明了其符合 Hannan 一致性[42-43]。

从 1951 年提出的很长一段时间内，虚拟对弈的研究都停留在博弈论领域，原因是对于斗地主、德州扑克和桥牌等现实世界的博弈游戏，确定性行为策略的数量级是指数级的，使用 MG 建模的计算复杂度太高。如果能用 EG 建模，就能将虚拟对弈的应用范围拓宽到很多现实博弈中。按照这一思路，Heinrich 等[6]以典型的不完美信息博弈游戏——无限制下注德州扑克为实验平台，首次将虚拟对弈成功应用于 EG 求解。这里需要注意的是，不完美信息环境中隐藏信息的存在使得其解决难度大幅上升，哪怕是相同规模的博弈，不完美信息环境要比完美信息环境复杂得多。Heinrich 等[6]分别对计算最佳响应策略和将最佳响应策略添加到平均策略这两个操作进行了改写。首先，计算最佳响应策略需要遍历整棵博弈树，计算每个叶子节点的效用，汇总得到最佳响应策略。然后，难点在于将最佳响应策略添加到平均策略。将 Agent i 的信息状态集记作 S^i，当博弈具有完美记忆时，每个 $s^i \in S^i$ 都独一无二地定义了 Agent i 为了到达状态 s^i 的动作序列 σ_{s^i}。Agent i 的任意策略 π^i 为其每个动作序列 σ 都引入了一个实现概率，定义为 $Rp(\pi^i;\sigma) = \prod_{(\sigma_{s'},a)\subseteq\sigma} \pi^i(a \mid s')$，其涵盖所有满足 $(\sigma_{s'},a)$ 是 σ 的子序列的 $s' \in S^i$ 和 $a \in A^i$。对于 Agent i 的任意两个行为策略 π 和 π，分别赋予 λ 和 $1-\lambda$ 的权重（$\lambda \in (0,1)$），进行组合如下：

$$\frac{\lambda \cdot Rp(\pi,\sigma_{s^i}) \cdot \pi(\cdot \mid s^i)}{\lambda \cdot Rp(\pi,\sigma_{s^i}) + (1-\lambda) \cdot Rp(\tilde{\pi},\sigma_{s^i})} + \frac{(1-\lambda) \cdot Rp(\tilde{\pi},\sigma_{s^i}) \cdot \tilde{\pi}(\cdot \mid s^i)}{\lambda \cdot Rp(\pi,\sigma_{s^i}) + (1-\lambda) \cdot Rp(\tilde{\pi},\sigma_{s^i})} \quad \forall s^i \in S^i \quad (14)$$

Heinrich 等[6]提出的虚拟对弈就是通过以上的加权组合方法将最佳响应策略添加到平均策略，来计算策略 $\{\pi_t\}(t \geqslant 1)$。通过将虚拟对弈从 MG 求解迁移到 EG 求解，让计算机求解大规模不完美信息博弈成为可能。

假定 N 个 Agent 的联合策略为 $\pi \in \Delta(A)$，除了第 i 个 Agent 外的所有 Agent 联合策略为 π^{-i}。对于任意 $\varepsilon > 0$ 和任意 $\pi \in \Delta(A)$，定义 Agent i 的 ε-最佳响应策略为 $Br_\varepsilon(\pi^{-i})$，它满足：

$$R^i(Br_\varepsilon(\pi^{-i}),\pi^{-i}) \geqslant_{\mu \in \Delta(A^i)} R^i(\mu,\pi^{-i}) - \varepsilon \quad (15)$$

在第 t 轮迭代中,任意 Agent i 先计算针对其他 Agent 平均策略的 ε_{t+1}-最佳响应策略 $\tilde{\pi}^i_{t+1} \in Br_\varepsilon(\pi_t^{-i})$,然后通过对平均策略 π^i_t 和最佳响应策略 $\tilde{\pi}^i_{t+1}$ 分别赋予 $1-\alpha_{t+1}$ 和 α_{t+1} 的权重进行组合来构建下一轮的平均策略 π^i_{t+1}。当 $t\to\infty$,ε_t 和 α_t 均会收敛到 0;也就是说,Agent 的平均策略在训练后期将越来越稳定,最终收敛于纳什均衡。由于这种方法在每轮迭代中求解最佳响应策略需要遍历博弈树的所有状态,将最佳响应策略添加到平均策略需要记住过往所有的策略,计算成本过高,会导致维度灾难。Heinrich 等[6]进一步提出了虚拟自我对弈(Fictitious Self-Play,FSP),通过引入基于样本的机器学习方法,用单 Agent 强化学习中的 Q-learning 算法来计算最佳响应策略,再用有监督学习的方法来学习平均策略;并且进一步将 FSP 与神经网络结合,在最佳响应策略计算部分使用 DQN 算法,提出了神经网络虚拟自我对弈(Neural Fictitious Self-Play,NFSP)算法,这是第一个在不完美信息博弈中不需要任何先验知识就能学习近似纳什均衡的端到端强化学习方法。NFSP 通过引入预期动态来解决协调问题——Agent 在进行动作选择时总是以 λ 的概率去选择最佳响应策略,以 $1-\lambda$ 的概率去选择平均策略。这里分别为最佳响应策略和平均策略设立了两个经验数据池。如果 Agent 选择了最佳响应策略,就将 (s_t,a_t) 存入平均策略的数据池。不难看出,平均策略是对过往最佳响应策略的模仿,这样做是为了在不稳定的不完美信息环境中,可以应对对手不断变化的策略。而不管其选择哪个策略,总是将 (s_t,a_t,r_t,s_{t+1}) 存入最佳响应策略的经验数据池中。

然而,由于对手策略的复杂性和 DQN 算法在离线模式下学习的特点,NFSP 在搜索空间和搜索深度规模较大的环境中表现较差,且收敛速度较慢,甚至难于收敛。浙江大学团队提出了蒙特卡洛神经虚拟自我对弈(Monte Carlo Neural Fictitious Self Play,MC-NFSP)[44],在最佳响应策略计算时采用了 MCTS 算法。实验表明,在奥赛罗棋中,MC-NFSP 将收敛到近似纳什均衡,但 NFSP 无法做到。为加快训练速度,该团队还提出了异步神经网络虚拟自我对弈(ANFSP)算法[44],使用并行的 actor learner 来稳定和加速训练。Agent 共享 DQN 中的动作值网络和有监督学习网络,在 DQN 中累积多个步骤的梯度,并在有监督学习中计算小批量的梯度。与 NFSP 相比,ANFSP 减少了数据存储所需的空间,可以更加稳定和快速地接近近似纳什均衡。

与以上介绍的由虚拟对弈派生出的方法着重于全局策略的求解不同,由反事实遗憾最小化派生出的方法侧重于子博弈求解。

CFR 算法是在解决大规模 EG 问题中的又一个重要突破,其主旨是通过最小化遗憾值去求解最优策略。它将下一轮选择某动作的概率与本轮未选择该动作所产生的遗憾值进行匹配,即若未选择某动作产生的遗憾值越小,下一轮博弈选择该动作的概率值越大。经过多次迭代,CFR 算法求解出的策略组合可逼近纳什均衡。在有 N 个 Agent、T 个步骤的 EG 中,Agent i 的遗憾值是从 Agent i 的所有可能策略中采样的最大值,定义如下:

$$Reg^i_T = \max_{\pi^i}\sum_{i=1}^{T}\left[R^i(\pi^i,\pi_t^{-i}) - R^i(\pi_t^i,\pi_t^{-i})\right] \tag{16}$$

2.1.2 节提到的到达概率 $\eta_\pi(h)$ 可以拆解为 Agent i 的贡献与其他 Agent 的贡献,即 $\eta_\pi(h)=\eta^i_\pi(h)\cdot\eta^{-i}_\pi(h)$。对于任意状态 $s\in S^i$ 以及任意动作 $a\in A(s)$,定义 $Z(s,a)=\{(h,z)\in H\times Z \mid h\in s, ha\subseteq z\}$,包括在状态 s 所有可能的历史对和在状态 s 执行动作 a 后的所有终止历史记录。由此可定义反事实动作-状态值函数和状态值函数如下:

$$Q^i_{\mathrm{CF}}(\pi,s,a) = \sum_{(h,z)\in Z(s,a)}\eta^{-i}_\pi(h)\cdot\eta_\pi(z\mid ha)\cdot R^i(z) \tag{17}$$

$$V^i_{\mathrm{CF}}(\pi,s) = \sum_{a\in A(s)}Q^i_{\mathrm{CF}}(\pi,s,a)\cdot\pi^i(a\mid s) \tag{18}$$

其中设定 Agent i 总是默认直接选择可到达状态 s 的动作。$Q^i_{\mathrm{CF}}(\pi,s,a)$ 和 $V^i_{\mathrm{CF}}(\pi,s)$ 之间的差值可以看作在状态 s 执行动作 a 的价值,Agent i 在状态 s 的反事实遗憾值定义如下:

$$Reg^i_T(s) = \max_{a\in A(s)}\sum_{i=1}^{T}\left[Q^i_{\mathrm{CF}}(\pi,s,a) - V^i_{\mathrm{CF}}(\pi,s)\right]$$
$$\forall s\in S^i \tag{19}$$

为了最小化总遗憾值,可以采用 EXP3[45]、Hedge[46] 和遗憾值匹配[47]去更新策略;这些方法可以保证在任意 $s\in S^i$,反事实遗憾值是 $\mathcal{O}(\sqrt{T})$ 量级,也就是保证了总遗憾值的上界是 $\mathcal{O}(\sqrt{T})$ 量级。因此,将 CFR 类型的算法运用于双人零和扩展式博弈时,在 T 步之后,平均策略是 $\mathcal{O}\left(\sqrt{\frac{1}{T}}\right)$-估计纳什均衡。

CFR 算法在每次更新策略时都需要遍历整个博弈树,这就导致运算成本过高、速度减慢。

为了提高运算效率,Brown 等[48]用神经网络去估计遗憾值。基于 CFR 算法最出色的应用是两个德扑 AI,Deepstack[49] 和 Libratus[50]。对于无限制下注德州扑克,较为标准的策略计算方法是先对其进行抽象,得到一个最大程度上保留原始游戏策略特征的小型版本

的游戏。通过对抽象版本的游戏进行求解(理论上可以选择任何可行的解),得到初步的策略——蓝图策略。然而,蓝图策略可能距离真实的解法差距较大。子博弈求解试图通过实时求解更为精细的抽象游戏来改进蓝图策略;而之所以子博弈求解在不完美信息博弈中显得尤为重要,是因为在完美信息博弈中,例如围棋大师,求解当前的子博弈只需要考虑以当前游戏状态为根节点的子博弈树即可,但是这些搜索方法并不适用于不完美信息博弈,因为它们并不考虑对手转移到叶节点之外策略的能力。这个弱点令搜索算法产生了脆弱的、不平衡的策略,从而使对手可以快速发现这个错误。在不完美信息博弈中,隐藏信息的存在使得游戏的子博弈不能孤立地考虑,对于给定子博弈的最优策略可能取决于在游戏其他部分里还未发生的情况下将要使用的策略。与以往将新出现的不包含在当前博弈树中的动作近似为已有动作的方法不同,Libratus 和 Deepstack 为新的动作实时创建一个新的子博弈;通过 CFR 算法来求解嵌套子博弈。接下来,将为新动作所构建的子博弈策略与原有的抽象博弈的策略进行融合,即可得到更细粒度抽象之后的游戏的策略。这种解决思路不仅可以大幅提升算法的精度,而且在解决特定子博弈时考虑到了其他子博弈可能对其产生的影响。具体来讲,Libratus 和 Deepstack 的根本区别在于是否使用神经网络来辅助限制子博弈树的深度。Libratus 只在前两轮下注中使用 CFR 算法求解嵌套子博弈,子博弈树的深度从当前子博弈的根节点开始到博弈结束。然后根据提前计算得出的蓝图策略向下执行(如果赌注较高时,考虑进行子博弈求解);这也导致它之后会把对手做出的博弈树之外的动作(这里的动作指下注大小)近似到某个相近的、已经经过抽象的动作上去。在前两轮获取蓝图策略时,采用较高密度的抽象可以改善这个问题。而 DeepStack 会在前两轮下注中将子博弈树的深度限制在每一轮的结束,求解一个有限深度的子博弈树,而这个子博弈树叶节点的动作值估计是通过神经网络估计的。换言之,在前两轮下注中所遇到的子博弈树只需在本轮结束前使用 CFR 算法求解,至于本轮结束时的虚拟动作值通过神经网络来估计。Deepstack 将其通过神经网络习得的面对具体博弈状态即可预测出虚拟动作值的能力,称之为“直觉”;这种直觉使其可以实时计算如何应对对手所做出的博弈树之外的行为。

表 1 对竞争场景下的 MARL 算法进行了总结。

表 1 竞争场景下 MARL 算法总结分析

算法/研究者	基于值/策略的方法	应用环境	贡献
minimax-Q learning 算法[29]	基于值	完美信息环境	旨在最大化自身的“最小”总奖励值
Sidford 等[34]	基于值	完美信息环境	将 Q-learning 算法扩展至通用模型的零和 TBSG
Buter 等[35]	基于值	小规模的不完美信息环境	将状态空间转换到信念状态空间
Von 等[36]	基于值	小规模的不完美信息环境	通过序列规范化表征将 EG 转换为 SG
Deepmind 研究团队[5]	基于值	完美信息环境	将 MCTS 算法与神经网络结合,运用于完美信息环境的大规模双 Agent 复杂序贯决策问题——围棋博弈中
Kaufmann 等[38]	基于值	不完美信息环境	将 MCTS 算法运用在不完美信息环境的双 Agent 竞争场景中
虚拟对弈(FP)算法[41]	基于策略	完美信息环境	Agent 不断根据对手的平均策略做出最佳响应,最终平均策略组合收敛到纳什均衡。
神经网络虚拟自我对弈(NFSP)算法[6]	基于策略	不完美信息环境	将虚拟对弈(FP)从 MG 求解迁移至 EG 求解,并使用神经网络来估计动作值函数。NFSP 算法是第一个在不完美信息博弈(德州扑克)中不需要任何先验知识就能学习近似纳什均衡的端到端强化学习方法。
CFR 算法[40]	基于策略	不完美信息环境	通过最小化遗憾值去求解最优策略
德扑 AI:Libratus[50] &Deepstack[49]	基于策略	不完美信息环境	通过 CFR 算法与神经网络结合来实时进行嵌套子博弈求解

2.3　合作场景

在合作场景的多 Agent 系统中,有些时候 Agent 是完全同质的,而有些时候 Agent 并不是同质的。

当有大量 Agent 时,“同质”意味着在系统进化中,Agent 是可以互换的且彼此难以区分,系统的总奖励函数与所有 Agent 的奖励是一致的。2.1.1 节提到,MG 中完全合作模式下,所有 Agent 的总奖励函数是一致的,因此可以使用中心化训练-分布式执行的结构,如图 4 所示。

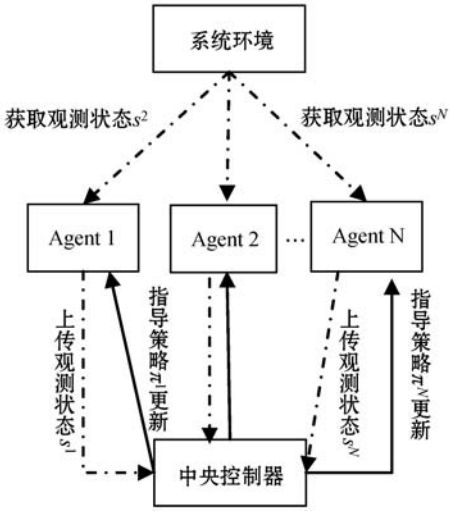

图 4　集中式训练-分布式执行的 MARL 结构示意图

为了解决环境的部分可观测性,这种结构由中央控制器去集成所有 Agent 的信息(包括联合动作甚至策略、联合奖励和联合观测状态),将隐藏的信息完全可视化[51-53]。集中控制器与 Agent 之间进行信息交换:Agent 将观测状态上传给中央控制器,中央控制器再指导 Agent 去更新策略。在合作模式下,由于学习目标的一致性,这种结构可以很大程度上简化训练。在非合作模式下,由于学习目标的差异,这种结构并没有很明显的简化效果。但是这样做也存在以下问题。首先,当均衡策略不唯一时,只有其中一部分均衡策略是最优的,此时每个 Agent 的均衡策略组合起来可能不会达到最优纳什均衡,Agent 间需要协调去寻找最优纳什均衡策略组合。Wang 等[54]提出最优自适应学习(Optimal Adaptive Learning,OAL)算法,它是第一个在团队 MG 中被证明收敛于纳什均衡的 MARL 算法。其次,集中式训练的方法会面临维度灾难,可扩展性不高,而且没有很好的方法来提取最优策略。Sunehag 等[61]提出介于分布式训练和集中式训练之间的 VDN 算法。不再直接求解联合动作值函数,而是假设系统的联合动作值函数能够被分解为各 Agent 动作值函数的累加,使用 DQN 算法的更新规则去集中学习近似的联合动作值函数。VDN 算法不仅可以减少运算规模,还可以直接从单 Agent 动作值函数中提取最佳策略。Rashid 等[52]提出 QMIX 算法,在 VDN 算法的基础上对单 Agent 动作值函数和联合动作值函数之间的映射关系进行改进,通过使用超网络将单 Agent 动作值函数向联合动作值函数进行非线性映射,不仅在性能和速度上获得提升,还可以在不完美信息环境中将额外状态信息加入到决策过程。

然而,大多数的实际合作场景中,例如智能传输系统[55]、群体机器人[56],Agent 并不是同质的,即它们不再共享相同的奖励函数,也有着各自的隐私偏好;之前提到的集中控制器可能也不复存在。这时采用一种新的结构,如图 5 所示。

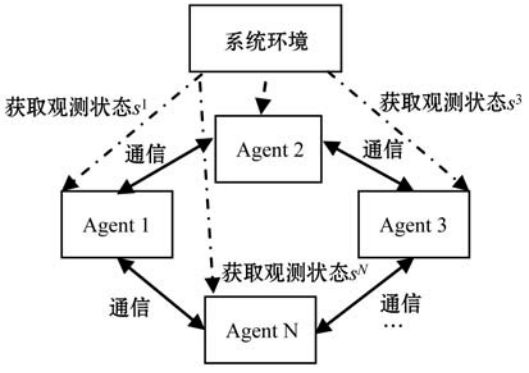

图 5　分布互联式训练的 MARL 结构示意图

在不同的时间节点,Agent 之间由不同的通信网络所连接;通过与邻居 Agent 的交流,本地信息得以在网络中进行传播。Kar 等[57]首次提出基于这种结构的可收敛的 MARL 算法——QD-learning,其将 Agent 与邻居间动作值的差异纳入 Q-learning 算法的更新规则,分布互联式的 Agent 通过局部处理和稀疏(可能是随机)通信网络上的信息交换进行协作,使网络平均无限期折扣成本最小化。当每个 Agent 只知道其本地在线成本数据,且 Agent 间通信网络弱连通时,QD-learning 算法收敛到期望值函数和每个 Agent 的最优平稳控制策略。为解决运算规模问题,Zhang 等[58]将使用神经网络的 Actor-Critic 算法应用于这种结构。

需要注意的是,Agent 之间交流的效率非常重要。Chen 等[59]提出一种通过设计通信触发原则来减少 Agent 与集中控制器间通信次数的分布式策略梯度算法。北京大学团队提出了一个基于注意力机制(ATOC Architecture)的通信模型[60],让 Agent 具备自主选择通信对象的能力,使 Agent 在大型 MARL 的部分可观测分布式环境下学习高效的通信。具体来讲,受视觉注意力循环模型的启发,研究者设计了一种注意力单元,它可以接收并编码局部观测结果和某个 Agent 的行动意图,并决定该 Agent 是否要与其他 Agent 进行通信并在可观测区域内合作。如果 Agent 选择合作,

则称其为发起者,它会为了协调策略选择协作者来组成一个通信组。通信组进行动态变化,仅在必要时保持不变。研究者利用双向 LSTM 单元作为信道来连接通信组内的所有 Agent。LSTM 单元将内部状态(即编码局部观测结果和行动意图)作为输入并返回指导 Agent 进行协调策略的指令。与 CommNet 和 BiCNet 分别计算内部状态的算术平均值和加权平均值不同,LSTM 单元有选择地输出用于协作决策的重要信息,这使得 Agent 能够在动态通信环境中学习协调策略。与现有的方法相比,ATOC Agent 被证明能够开发出更协调复杂的策略,并具备更好的可扩展性。

表 2 对合作场景下的 MARL 算法进行了总结。

表 2 合作场景下 MARL 算法总结分析

算法/研究者	结构	贡献	缺陷
COMA 算法[53]	集中式训练-分布式执行	采用集中式训练-分布式执行的 Actor-Critic 算法框架,消解了多 Agent 系统中环境的不稳定性;采用反事实基线计算优势函数	集中式训练需要输入全局状态与动作信息,运算成本较高,使得该算法难以扩展及落地
VDN 算法[61]	介于分布式训练与集中式训练之间	VDN 算法将联合动作值函数分解为各 Agent 动作值函数的累加,减小了运算规模	纯线性方式的值函数整合方法略显单薄,而且未能利用好状态信息
QMIX 算法[52]	介于分布式训练与集中式训练之间	在 VDN 算法的基础上,QMIX 算法使用超网络将单 Agent 动作值函数向联合动作值函数进行非线性映射。不仅提高了性能,还有利于在不完美信息环境中捕捉额外状态信息	有待于向拥有更大数量和密度的单元的任务进行扩展
QD-learning[57]	分布互联式训练	QD-learning 算法将 Agent 与邻居间动作值的差异纳入 Q-learning 算法的更新规则,分布互联式的 Agent 通过局部处理和稀疏(可能是随机)通信网络上的信息交换进行协作	运算规模问题亟待解决
Zhang 等[58]	分布互联式训练	将使用神经网络的 Actor-Critic 算法应用于分布互联式训练结构,解决了运算规模问题	
基于注意力机制(ATOC Architecture)的通信模型[60]	分布互联式训练	ATOC 首次将注意力机制通信运用于 MARL,让 Agent 具备自主选择通信对象的能力,使 Agent 在大型部分可观测分布式环境下学习高效地通信	

2.4 混合场景

相比于单纯的合作模式和竞争模式,混合模式下的 MARL 算法研究更加有难度,并且留有更多的理论空白。在这里简要介绍近期一些非常出色的工作。

Brown 等[62]提出了应用于六人德州扑克中具有超人表现的德扑 AI Pluribus。它不再执着于求解纳什均衡策略组合,而是寻找总能击败人类选手的策略组合。首先,Pluribus 通过自对弈计算出“蓝图”策略。起初其行动完全随机,但它会采用蒙特卡洛反事实遗憾最小化算法(MCCFR)来不断改进自己的策略(MCCFR 随机考虑一部分行动,而不是整个行动空间)。然后,Pluribus 和真实玩家对战,基于“蓝图”策略,用实时搜索(real-time search)技术寻找更好的策略。由于德州扑克是典型的不完美信息环境,Agent 对其他玩家的特征、策略和奖励都没有完整的了解。所以为了应对对手不断变化的策略,作者假设每个玩家会有自己的 4 个策略,包括“蓝图”策略和它的三个变种,并且会在博弈中选择其中一种,以计算可以平衡不同情况的策略。实际运行中,Pluribus 平均每回合只需要 20 秒思考时间,比专业选手快一倍;研究者设计了两个比赛,分别是 5H + 1AI(H 代表杰出人类玩家),以及 1H + 5AI。在 5H + 1AI 中,Pluribus 平均每局能赢 48mbb(milli big blinds),在六人德州扑克中这是极好的成绩;在 1H + 5AI 中,Pluribus 以平均每局 32mbb 的成绩击败人类。

混合模式下比较典型的一个例子是两组 Agent 的零和博弈,组内 Agent 共享本组的奖励;相比双 Agent 零和博弈,它更为复杂,因为不仅要考虑两个团队间的竞争,还要考虑组内 Agent 间的合作。

Dota2 是这种结构的重要实验平台之一,两个团队互相攻击并且自我守卫,每个团队有 5 个玩家,每个玩家独立控制一个“英雄”角色并且只能观测到部分环境信息。OpenAI Five[7]是非常出色的 Dota2 游戏 AI,击败了人类世界冠军玩家。将强化学习应用于 Dota2 游戏主要有以下两个难点:首先是“稀疏奖励”问题,

由于 Dota2 游戏中需要经过非常多的复杂序列动作才能获得最终的奖励,而强化学习在起初训练阶段动作完全随机,这就使得 Agent 较难取胜,也难以学习到有效经验。其次,即使取得胜利,那么怎样在 Agent 所采取的一系列动作中区分出没有贡献甚至不正确的动作和贡献较大的动作呢? 针对这两个难点,OpenAI Five 通过手动设计即时小目标奖励以及使用“团队灵魂”这个超参数来进行奖励重塑。在策略学习过程中,OpenAI Five 采用自对弈形式的 PPO 算法,且在 Actor 网络和 Critic 网络中应用可学习序列特征表示的 LSTM 结构,这样做不仅可以帮助解决“稀疏奖励”问题,将长期奖励分配到过往的状态-动作序列中,而且有助于消解环境的部分可观测性。最终 OpenAI Five 以悬殊的差距击败了人类联赛中的高手们。

StarCraft2 也是 MARL 较为重要的实验平台之一。与 Dota2 类似,StarCraft2 也存在以下困难:动作空间组合数目过多;稀疏奖励和不完美信息环境。之前提出的一些针对 StarCraft2 的游戏 AI 都对游戏进行了简化或者使用了人工设计的子系统,Vinyals 等[8]首次提出应用于 StarCraft2 的端到端训练 AI——AlphaStar。不同于 OpenAI Five 中纯采用自对弈来学习策略,AlphaStar 先使用人类数据来进行有监督学习,目标是依据给定观测状态来预测下一步动作选择;再使用强化学习 A3C 算法来学习怎样最大化胜率,并通过模仿学习来尽量靠近能够取得较高收益的轨迹。AlphaStar 最关键的技术在于其同时训练了针对不同对手策略的策略集合,包括 main agents、main exploiters 和 league exploiters。其中 main agents 使用先验虚拟自对弈(Prioritized Fictitious Self-Play,PFSP)算法,以较高的概率去与历史对抗中对其胜率较高的 Agent 进行对战,它的对手从所有策略集合中(包括历史策略)选择;main exploiters 的对手是当前 main agents,其目的是寻找当前策略集合的弱点;league exploiters 也使用 PFSP 算法,其对手为历史 main agents,其目的是寻找系统性弱点。main exploiters 和 league exploiters 都会定期被重置为通过有监督学习得到的 Agent,以增强策略集合对抗人类策略的稳定性。经过在 32 块 TPU 上 44 天的训练,AlphaStar 学习完成之后的版本 AlphaStar Final 的测试得分超过了 99.8% 的玩家。

表 3 对混合场景下典型的 MARL 运用进行了总结。

表 3　混合场景下 MARL 运用总结分析

算法	实验场景	贡献
Pluribus[62]	六人德州扑克	由于德州扑克是典型的不完美信息环境,为了应对对手不断变化的策略,Pluribus 假设每个玩家会有自己的 4 个策略,包括“蓝图”策略和它的三个变种,并且会在博弈中选择其中一种,以计算可以平衡不同情况的策略
OpenAI Five[7]	Dota2	1. 通过手动设计即时小目标奖励以及使用“团队灵魂”这个超参数来进行奖励重塑。 2. 在 Actor 网络和 Critic 网络中应用可学习序列特征表示的 LSTM 结构。不仅可以帮助解决“稀疏奖励”问题,而且有助于消解环境的部分可观测性
AlphaStar[8]	StarCraft2	1. AlphaStar 先使用人类数据来进行有监督学习,目标是依据给定观测状态来预测下一步动作选择;再使用 A3C 算法来学习怎样最大化胜率,并通过模仿学习来尽量靠近能够取得较高收益的轨迹。 2. AlphaStar 最关键的技术在于其同时训练了针对不同对手策略的策略集合

3　MARL 面临的挑战

MARL 领域起步较晚,面临着诸多挑战。本节将分别简述 MARL 在技术层面以及理论层面上面临的挑战。在技术层面上,MARL 面临的主要挑战是对计算资源的较高要求,这一方面承袭于 DRL,另一方面是由于多 Agent 问题本身的复杂性导致的。在理论层面上,本节对 MARL 面临的主要挑战之一,环境的不稳定性进行介绍,并在本节的最后将以对 NFSP 算法的改进为例进行实验,探讨其可能的解决途径。

3.1　技术层面

在技术层面,MARL 面临的主要挑战是对计算能力和资源的门槛过高。首先,由于超参数变化会很明显的影响不同环境中深度强化学习算法的效果(Henderson 等[63]测试了 TRPO、DDPG、PPO 和 ACKTR),所以将其应用于多 Agent 系统时,更需要对超参数进行精细地调整。其次,深度强化学习算法总需要 Agent 与环境进行大量的交互以获取有效的样本,例如 A3C 算法经过 2 亿轮模拟才学会玩一个 Atari 游戏,需要 16 个 CPU 进行长达 4 天的训练[3];而将其应用于多 Agent 系统时还需要考虑怎样协调多个 Agent 完成特定任

务,这就导致了计算资源的进一步上升。基于高计算量的方法可作为基准测试的前沿,它们表明随着数据和计算规模的增长所能够达到的理想结果,例如 OpenAI Five 使用 128 000 个 CPU 核和 256 个 GPU 每天可以产生 180 年的游戏数据。为了加速 MARL 的落地,使其更广泛地应用于现实问题的解决,研究者们需要平衡计算成本和算法效果,在减少计算资源需求的同时尽可能保障算法效果。

3.2 理论层面

本文针对 MARL 在理论层面的核心挑战—环境的不稳定性进行了考察。

在多 Agent 系统中,Agent 不仅要与环境交互,还需要与其他 Agent 协作/竞争去完成特定的任务。这意味着系统中有多个 Agent 同时进行学习与演化,其中任何一个 Agent 的动作选择都很可能会影响其他 Agent 的动力学过程,这就导致了从单个 Agent 的角度来看,环境变得不再稳定(仅凭借单个 Agent 自身的策略无法解释环境的变化)。具体来讲,在环境部分可知的系统中,如果全局状态与之前不同,Agent 的动力学过程就会发生变化;就算环境是完全可知的,Agent 可以观测到全局状态,一旦其他 Agent 的策略发生改变,其动力学过程依然会变化。

环境的不稳定性带来的主要问题是经验回放失效。在单 Agent 强化学习算法中,当采用线下训练方法,常常将过往的动力学过程存入经验池中,后期通过经验回放进行训练。这就要求环境是稳定的,即 Agent 现在的动力学过程只依赖于其自身之前的状态和动作;而多 Agent 系统中环境的不稳定性是违背了这个要求的,训练中经验池里存储的过往动力学过程与当前实际的动力学过程不符,会导致经验回放会失效甚至误导训练[64]。

针对环境的不稳定性导致的经验回放失效,可以通过向经验数据中添加信息来解决,近期的方法大概可以分为两类。

第一类方法通过直接将其他 Agent 的策略纳入考虑,来消解多个 Agent 同时演化带来的环境不稳定性。例如,DPIRQN 算法[65]直接通过辅助任务的方法来学习其他 Agent 的策略特征,再基于这些策略特征来训练自己的值网络。Lowe 等[66]提出了 MADDPG,它基于 DDPG,采用集中式训练-分布式执行的结构。具体来讲,在训练时 MADDPG 向 Critic 网络输入所有 Agent 的状态和动作,并在经验池中记录所有 Agent 的动力学过程,在执行时仅需要将单个 Agent 的状态输入 Actor 网络来获取策略即可。这类方法中,Agent 需要考虑联合状态和动作空间,会使得计算维度随 Agent 数量增加而迅速上升;这被称为 MARL 的“复合本质”(Combinatorial Nature)[67]。为了解决这个问题,2.3 节中介绍的 VDN 算法和 QMIX 算法,将依赖联合动作选择的值函数或者奖励函数构建为可分解的结构。假设 Agent 之所以会互相协作,完全是因为“委派奖励”不同;通过使用“委派奖励”这个中间变量,将 Agent 之间的互动在形式上解耦[68]。

第二类方法主旨是在经验回放时对经验池中存储的过往动力学过程进行矫正。例如,在经验回放时对于靠近结束状态的样本赋予较高的优先级[69],但是这种方法对于场景的局限性较大;Bishop 等[70]提出适用于多 Agent 的重采样方法,与之前单 Agent RL 算法中的重采样类似,它采用联合动作概率对过往经验进行矫正。

本文 2.3 节中介绍了适用于双 Agent 零和博弈的 NFSP 算法,其中 Agent 在求解最佳响应策略时采用 DQN 算法。我们知道,DQN 算法的收敛依赖于经验回放的成功运用。但是,在多 Agent 系统尤其是环境部分可知的系统中,环境是不稳定的,经验回放会失效。所以,我们借鉴刚才提到的对过往经验进行矫正的方法,在 NFSP 算法求解最佳响应策略的部分,采用 1.2.2 节中介绍的 PPO 算法,通过重采样方法去进行策略更新。

为了验证上述改进的效果,本文在由 DeepMind 团队提供的双人射击仿真游戏环境 Laser Tag[71]下进行了测试。在 Laser Tag 实验环境中,Agent 只能观测到局部视野(包括其前方 17 个空格、左右两侧各 10 个空格和后方 2 个空格),无法了解全局信息;它有左转、右转、前进、后退和射击等动作选择。每轮游戏开始时,两个 Agent 被随机初始在生成点;如果一个 Agent 连续击中另一个 Agent 两次,那么它将获得即时奖励值为 1。

总共进行了 1 500 回合的模拟。每回合包含 1 000 步迭代,其中 NFSP 算法训练 1 000 次,本文改进的 NFSP 算法训练了 320 次。实验结果如图 6 – 图 8 所示。

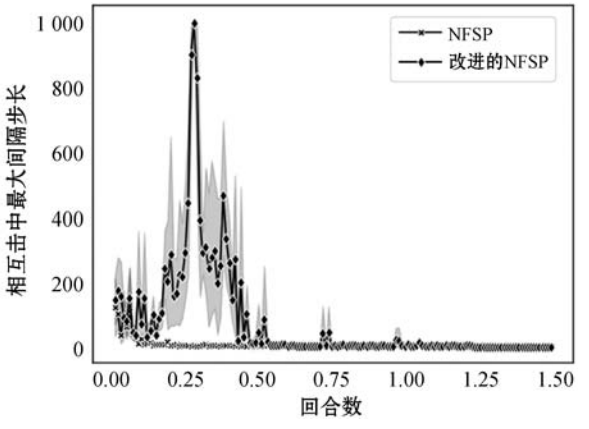

图 6 纳什均衡收敛曲线

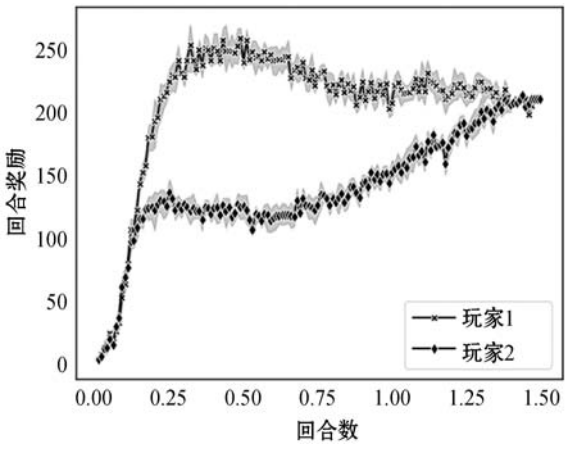

图 7　NFSP 算法双 Agent 奖励曲线(每回合训练 1 000 次)

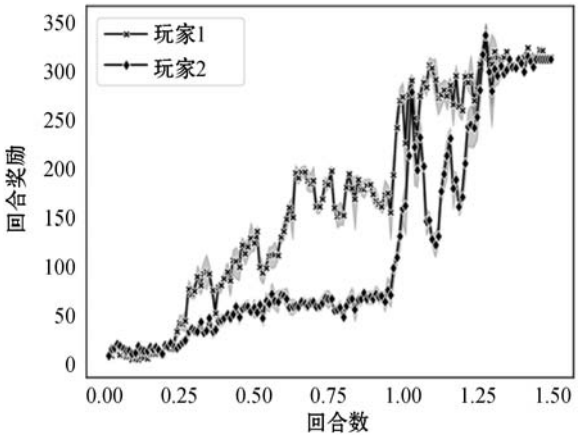

图 8　改进的 NFSP 算法双 Agent 奖励曲线(每回合训练 320 次)

由图 6 可知,虽然改进的 NFSP 算法的相互击中最大间隔步长在训练前期略有震荡,但是在后期它和 NFSP 算法中的相互击中最大间隔步长一样,收敛于 0;也就是说两个玩家逐步学会了击中对方的“技巧”,这表明改进的 NFSP 算法向纳什均衡的收敛性是可以保证的。由图 7 和图 8 可知,玩家获得的回合奖励在 NFSP 算法中收敛于 200 附近,而在改进的 NFSP 算法中其收敛于 300 附近;这表明在 Laser Tag 实验环境中,我们改进的 NFSP 算法仅需要 NFSP 算法约 1/3 的训练次数,可以使玩家的回合奖励获得接近较大的提升。由此看出,在多 Agent 系统中,对过往经验采用重采样方法进行矫正,可以提升 Agent 总奖励值,增强 Agent 利用和探索的能力,并且在一定程度上消解了环境的不稳定性。

4　结　语

鉴于近年来 MARL 在大规模复杂序贯决策问题中的广泛应用,本文从交互模式出发对 MARL 进行了系统的综述。首先,本文对 RL 和 DRL 进行了回顾;然后,本文介绍了 MARL 的两种理论框架,并将 MARL 按系统任务划分为三类,分别综述了近年基于合作、竞争和混合场景下非常经典的一些 MARL 工作。最后,本文对 MARL 面临的主要挑战及对应的解决方案进行了分析,并尝试采用重采样对过往经验进行矫正,以弥补由环境不稳定性导致的经验回放失效对于训练的干扰和误导。本文还通过实验初步证明了这种改进的可行性和有效性。

虽然 MARL 在模拟实验中已经有许多超越人类的卓越表现,但是在理论研究上成熟度较低,依然有很多空白的值得深入研究的地方。我们期待这些理论上的空白在未来能够被看到、被解决,同时也期待看到更多 MARL 算法的实际落地与应用,为实现通用人工智能注入新的驱动力。

参　考　文　献

[1] Lillicrap T P, Hunt J, Pritzel A, et al. Continuous control with deep reinforcement learning[C]//International Conference on Learning Representations,2015.

[2] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature,2015,518:529 - 533.

[3] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C]//33rd International Conference on Machine Learning,2016:1928 - 1937.

[4] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[EB]. arXiv:1707.06347,2017.

[5] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature,2016,529:484 - 489.

[6] Heinrich J, Lanctot M, Silver D. Fictitious self-play in extensive-form games[C]//32nd International Conference on Machine Learning,2015:805 - 813.

[7] OpenAI. OpenAI five[EB/OL].[2022 - 12 - 20]. https://openai.com/research/openai-five.

[8] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning[J]. Nature,2019,575:350 - 354.

[9] 原魁,李园,房立新. 多移动机器人系统研究发展近况[J]. 自动化学报,2007,33(8):785 - 794.

[10] 赖俊,魏竞毅,陈希亮. 分层强化学习综述[J]. 计算机工程与应,2021,57(3):72 - 79.

[11] 孙彧,曹雷,陈希亮,等. 多智能体深度强化学习研究综述[J]. 计算机工程与应用,2020,56(5):13 - 24.

[12] Puterman M L. Markov decision processes: Discrete stochastic dynamic programming[M]. New York: John Wiley &

Sons,1994.

[13] Watkins C. Learning from delayed rewards[D]. Cambridge: University of Cambridge,1989.

[14] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning,1992,8(3-4):229-256.

[15] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. 2nd Edition. Cambridge: MIT Press,2018.

[16] Konda V R, Tsitsiklis J N. Actor-critic algorithms[C]//Advances in Neural Information Processing Systems,2000.

[17] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015,521:436-444.

[18] Hasselt H V. Double Q-learning[C]//Advances in Neural Information Processing Systems,2010:2613-2621.

[19] Wang Z Y, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[C]//33rd International Conference on Machine Learning, 2016: 1995-2003.

[20] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural Computation,1997,9(8):1735-1780.

[21] Meuleau N, Peshkin L, Kim K E, et al. Learning finite-state controllers for partially observable environments[C]// 15th Conference on Uncertainty in Artificial Intelligence, 1999:427-436.

[22] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization[C]//32nd International Conference on Machine Learning,2015:1889-1897.

[23] Shapley L S. Stochastic games[J]. Proceedings of the National Academy of Sciences of the United States of America, 1953,39(10):1095-1100.

[24] Basar T, Olsder G J. Dynamic noncooperative game theory [M]//Classics in Applied Mathematics. 2nd ed. New York: Society for Industry and Applied Mathematic,1999.

[25] Zazo S, Macua S V, Sanchez-Fern M, et al. Dynamic potential games with constraints: Fundamentals and applications in communications[J]. IEEE Transactions on Signal Processing,2016,64(14):3806-3821.

[26] Doan T, Maguluri S, Romberg J. Finite-time analysis of distributed TD(0) with linear function approximation on multi-agent reinforcement learning[C]//36th International Conference on Machine Learning,2019:1626-1635.

[27] Littman M L. Markov games as a framework for multi-agent reinforcement learning[C]//18th International Conference on Machine Learning,1994:157-163.

[28] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550:354-359.

[29] Neumann J, Morgenstern O. Theory of games and economic behavior[M]. 60th Anniversary Commemorative Edition. Princeton: Princeton University Press.

[30] Al-Tamimi A, Abu-Khalaf M, Lewis F L. Adaptive critic designs for discrete-time zero-sum games with application to H_∞ control[J]. IEEE Transactions on Systems, Man, and Cybernetics,2007,37(1):240-247.

[31] Al-Tamimi A, Lewis F L, Abu-Khalaf M. Model-free Q-learning designs for linear discrete-time zero-sum games with application to H_∞ control[C]//European Control Conference,2007:1668-1675.

[32] Zou S F, Xu T Y, Liang Y B. Finite-sample analysis for SARSA with linear function approximation[EB]. arXiv: 1902.02234,2019.

[33] Jia Z Y, Yang L F, Wang M D. Feature-based Q-learning for two-player stochastic games[EB]. arXiv:1906.00423, 2019.

[34] Sidford A, Wang M D, Yang L F, et al. Solving discounted stochastic two-player games with near-optimal time and sample complexity[EB]. arXiv:1908.11071,2019.

[35] Buter B J. Dynamic programming for extensive form games with imperfect information[D]. Amsterdam: Universiteit van Amsterdam,2012.

[36] Stengel B. Computing equilibria for two-person games[J]. Handbook of Game Theory with Economic Applications, 2002,3:1723-1759.

[37] Chang H S, Fu M C, Hu J, et al. An adaptive sampling algorithm for solving Markov decision processes[J]. Operations Research,2005,53:126-139.

[38] Kaufmann E, Koolen W M. Monte-Carlo tree search by best arm identification[C]//31st International Conference on Neural Information Processing Systems,2017:4904-4913.

[39] Hannan J. Approximation to Bayes risk in repeated play[J]. Contributions to the Theory of Games,1957,3:97-139.

[40] Zinkevich M, Johanson M, Bowling M, et al. Regret minimization in games with incomplete information[C]//20th International Conference on Neural Information Processing Systems,2007:1729-1736.

[41] Brown G W. Iterative solution of games by fictitious play [J]. Activity Analysis of Production and Allocation,1951, 13:374-376.

[42] Hart S, Mas-Colell A. A general class of adaptive strategies [J]. Journal of Economic Theory,2001,98(1):26-S54.

[43] Benaim M, Faure M. Consistency of vanishingly smooth fictitious play[J]. Mathematics of Operations Research,2013,

38:437 – 450.

[44] Zhang L, Wang W, Li S, et al. Monte Carlo neural fictitious self-play: Approach to approximate Nash equilibrium of imperfect-information games[EB]. arXiv:1903.09569,2019.

[45] Auer P, Cesa-Bianchi N, Freund Y, et al. The non-stochastic multiarmed bandit problem[J]. SIAM Journal on Computing,2002,32(1):48 – 77.

[46] Freund Y, Schapire R E. Adaptive game playing using multiplicative weights [J]. Games and Economic Behavior, 1999,29(1 – 2):79 – 103.

[47] Hart S, Mas-Colell A. A simple adaptive procedure leading to correlated equilibrium[J]. Econometrica,2000,68(5): 1127 – 1150.

[48] Brown N, Lerer A, Gross S, et al. Deep counterfactual regret minimization[C]//36th International Conference on Machine Learning,2019:793 – 802.

[49] Moravcik M, Schmid M, Burch N, et al. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker [J]. Science,2017,356(6337):508 – 513.

[50] Brown N, Sandholm T. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals [J]. Science, 2018,359(6374):418 – 424.

[51] Gupta J K, Egorov M, Kochenderfer M. Cooperative multi-agent control using deep reinforcement learning[C]//International Conference on Autonomous Agents and Multi-Agent Systems,2017:66 – 83.

[52] Rashid T, Samvelyan M, Schroeder C, et al. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning[C]//35th International Conference on Machine Learning,2018:4295 – 4304.

[53] Foerster J N, Farquha G, Afouras T, et al. Counterfactual multi-agent policy gradients[C]//32nd AAAI Conference on Artificial Intelligence,2018:2974 – 2982.

[54] Wang X, Sandholm T. Reinforcement learning to play an optimal Nash equilibrium in team Markov games[C]//15th International Conference on Neural Information Processing Systems,2002:1603 – 1610.

[55] Zhang K Q, Lu L Q, Lei C, et al. Dynamic operations and pricing of electric unmanned aerial vehicle systems and power networks[J]. Transportation Research Part C: Emerging Technologies,2018,92:472 – 485.

[56] Corke P, Peterson R, Rus D. Networked robots: Flying robot navigation using a sensor net[M]//Robotics Research. Berlin: Springer,2005:234 – 243.

[57] Kar S, Moura J M, Poor H V. QD-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations[J]. IEEE Transactions on Signal Processing,2013,61(7):1848 – 1862.

[58] Zhang K Q, Yang Z R, Liu H, et al. Fully decentralized multiagent reinforcement learning with networked agents [C]//I 35th International Conference on Machine Learning, 2018:5872 – 5881.

[59] Chen T Y, Zhang K Q, Giannakis G B, et al. Communication-efficient distributed reinforcement learning[EB]. arXiv:1812.03239v1,2018.

[60] Jiang J C, Lu Z Q. Learning attentional communication for multi-agent cooperation[C]//32nd International Conference on Neural Information Processing Systems, 2018: 7265 – 7275.

[61] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning[EB]. arXiv: 1706.05296,2017.

[62] Brown N, Sandholm T. Superhuman AI for multiplayer poker [J]. Science,2019,365(6465):885 – 890.

[63] Henderson P, Islam R, Bachman P, et al. Deep Reinforcement Learning That Matters[C]//32nd AAAI Conference on Artificial Intelligence,2018:3207 – 3214.

[64] Lin L J. Self-improving reactive agents based on reinforcement learning, planning and teaching[J]. Machine learning,1992,8(3 – 4):293 – 321.

[65] Hong Z W, Su S Y, Shann T Y, et al. A deep policy inference Q-network for multiagent systems[C]//17th International Conference on Autonomous Agents and Multiagent Systems,2018:1388 – 1396.

[66] Lowe R, Wu Y, Tamar A, et al. Multiagent actor-critic for mixed cooperative-competitive environments[C]//31st International Conference on Neural Information Processing Systems,2017:6382 – 6393.

[67] Hernandez-Leal P, Kartal B, Taylor M E. A survey and critique of multiagent deep reinforcement learning[EB]. arXiv:1810.05587v1,2018.

[68] Zhang T J, Xu H Z, Wang X L, et al. Multiagent collaboration via reward attribution decomposition[EB]. arXiv:2010.08531,2020.

[69] Bloembergen D, Kaisers M, Tuyls K. Lenient frequency adjusted Q-learning[C]//22nd Belgian/Netherlands Artificial Intelligence Conference,2010.

[70] Bishop C M. Pattern recognition and machine learning[M]. New York: Springer,2006.

[71] Shoham Y, Leyton-Brown K. Multiagent systems: Algorithmic, game-theoretic, and logical foundations [M]. Cambridge: Cambridge University Press.