计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202220697
61(1): 145−155, 2024

基于深度强化学习的掼蛋扑克博弈求解

葛振兴[1] 向 帅[1] 田品卓[2] 高 阳[1,3]
[1](计算机软件新技术国家重点实验室(南京大学) 南京 210023)
[2](上海大学计算机工程与科学学院 上海 200444)
[3](南京大学深圳研究院 广东深圳 518057)
(zhenxingge@smail.nju.edu.cn)

Solving GuanDan Poker Games with Deep Reinforcement Learning

Ge Zhenxing[1], Xiang Shuai[1], Tian Pinzhuo[2], and Gao Yang[1,3]
[1](*National Key Laboratory for Novel Software Technology* (*Nanjing University*), *Nanjing* 210023)
[2](*School of Computer Engineering and Science*, *Shanghai University*, *Shanghai* 200444)
[3](*Shenzhen Research Institute of Nanjing University*, *Shenzhen*, *Guangdong* 518057)

Abstract Decisions are often made in complex environment without exact information in many real-world occasions. Hence the capability of making proper decisions is expected for artificial intelligence agents. As abstractions of the real world, games provoke interests of researchers with the benefits of well-defined game structure and the facility to evaluate various algorithms. Among these games, GuanDan poker games are typical games with large action space and huge information set size, which exacerbates the problem and increases the difficulty to solve these games. In this work, we propose a novel soft deep Monte Carlo(SDMC) method to overcome the above-mentioned difficulties. By considering how the expert strategy acts in the training process, SDMC can better utilize the expert knowledge and accelerate the convergence of training process. Meanwhile, SDMC applies an action sample strategy in real time playing to confuse the opponents and prohibits the potentional exploitation of them, which could also lead to significant improvement of the performance against different agents. SDMC agent was the champion of the 2nd Chinese Artificial Intelligence Game Algorithm competition. Comprehensive experiments that evaluate the training time and final performance are conducted in this work, showing superior performance of SDMC against other agents such as the champion of 1st competition.

Key words imperfect information; deep reinforcement learning; multi-agent system; soft deep Monte Carlo method; poker game

摘 要 在不确定信息的复杂环境下进行决策是现实中人们经常面对的困难之一,因此具有能够进行良好决策的能力被视为人工智能的重要能力之一.而游戏类型的博弈作为对现实世界的一种高度抽象,具有良定义、易检验算法优劣等特点,成为研究的主流.其中以掼蛋为代表的扑克类博弈不仅具有他人手牌未知这样的难点,还由于可选出牌动作与他人手牌情况数量庞大等特点,难以进行高效求解.因此,提出

收稿日期: 2022-08-08; **修回日期**: 2023-04-06
基金项目: 国家重点研发计划项目(2018AAA0100905); 国家自然科学基金项目(62192783, 62276142, 62206166); 江苏省产业前瞻与关键核心技术竞争项目(BE2021028); 深圳市中央引导地方科技发展资金项目(2021Szvup056); 上海市扬帆计划项目(23YF1413000)
This work was supported by the National Key Research and Development Program of China (2018AAA0100905), the National Natural Science Foundation of China (62192783, 62276142, 62206166), the Primary Research & Developement Plan of Jiangsu Province (BE2021028), the Shenzhen Fundamental Research Program (2021Szvup056), and the Shanghai Yangfan Program (23YF1413000).
通信作者: 高阳(gaoy@nju.edu.cn)

了一种软深度蒙特卡洛（soft deep Monte Carlo, SDMC）求解方法. 该方法能够更好地融合领域知识, 加快策略学习速度, 并采用软动作采样策略调整实时决策, 提升策略胜率. 所提出的 SDMC 方法训练出的策略模型参加第 2 届“中国人工智能博弈算法大赛”时获得冠军. 与第 1 届比赛冠军策略和第 2 届其他策略模型的实验对比证明了该方法在解决掼蛋扑克博弈中的有效性.

关键词 非完美信息; 深度强化学习; 多智能体系统; 软深度蒙特卡洛方法; 扑克博弈

中图法分类号 TP391

游戏博弈作为现实世界的一种高度抽象, 具有良定义、易检验算法性能等特点, 成为目前智能决策研究的热点. 近些年, 一系列人工智能方法在游戏中均取得了很好的效果, 甚至战胜人类, 彰显出人工智能在现实应用中极大的潜力, 具有重大的意义. 而且, 不断有新的算法在各类游戏博弈中取得重要进展. 如在围棋中, 以 AlphaGo[1] 和 AlphaZero[2] 为代表的人工智能战胜了李世石、柯洁等人类顶尖高手; Libratus[3] 和 Pluribus[4] 战胜德州扑克(Texas Hold'em)职业冠军; Suphx[5] 在天凤麻将平台超越职业选手段位; AlphaStar[6], OpenAI-Five[7] 分别在星际争霸中与 DOTA2 中战胜人类世界冠军等.

目前针对棋牌类游戏存在多种求解方式, 例如: 1)基于强化学习(reinforcement learning)的方法[1-2, 6-8]采用试错的方式学习智能体在自身观测状态下的最优策略, 通过深度神经网络拟合状态动作值函数或状态动作概率分布的方法, 根据获取到的经验更新相应神经网络, 得到更好的策略; 2)基于反事实遗憾最小化(counterfactual regret minimization, CFR)的方法[3-4, 9-10]采用类似在线学习的方式, 在每一轮迭代中计算每种自身观测状态下所有动作的反事实遗憾, 根据遗憾匹配(regret matching)等遗憾最小化方法生成新一轮策略, 尝试降低新策略的遗憾, 最终输出每一轮的平均策略; 3)通过在线优化的方法, 如一阶方法(first order method)[11], 将中小规模二人零和博弈问题建模为一个凸优化问题进行求解.

国内扑克游戏, 如掼蛋、斗地主等, 作为一类非完美信息博弈, 相较于目前已有较好算法的德州扑克等游戏博弈有较大差异. 国内扑克游戏具有信息集状态多、动作空间复杂、状态动作难以约简等特点[8], 因此大部分现有方法难以应用. 例如用于求解德州扑克的蒙特卡洛反事实遗憾最小化[10,12](Monte Carlo counterfactual regret minimization, MCCFR)算法虽然能缓解在求解德州扑克问题时由于博弈树大小而引发的难以迭代遍历问题[13], 但是斗地主或掼蛋这样无法简单进行状态动作空间约简的扑克游戏, 其博弈树规模仍过于庞大, 无法简单适用; 经典的强化学习方法如 DQN[14-15], A3C[16-17] 等则由于较大的动作空间导致这些算法的网络结构难以较好地拟合扑克类的值函数[8], 从而无法在掼蛋、斗地主等国内扑克类游戏中取得较好的效果.

深度蒙特卡洛[8](deep Monte Carlo, DMC)方法是目前针对国内扑克游戏设计人工智能算法所面临问题的主要解决途径之一. DMC 方法采用蒙特卡洛采样评估状态动作值函数, 其考虑到斗地主等扑克游戏动作不易约简且动作之间由于出牌相似而具有相似关系的特点, 通过将动作进行编码与状态一同作为神经网络的输入, 借此解决动作空间大且不易约简的问题. 同时 DMC 方法采用 TorchBeast[18] 训练框架, 通过大量采样来降低训练方差, 在斗地主游戏中取得了较好的效果. 但是单纯的 DMC 方法在面对以掼蛋为代表的更大规模扑克博弈时, 依然面临一些问题: 1)DMC 方法需要大量的训练时间. 采用 DMC 方法的 DouZero 系统, 在对抗专家策略的监督学习方法时, 在斗地主环境中需要 10 天时间才能达到 50% 的对抗胜率. 因此对于更复杂的扑克博弈如掼蛋, 其信息集数量、信息集大小、动作空间、每局历史信息长度均远超斗地主, 需要更多的训练时间. 2)DMC 方法实际执行策略过程中总是选择第 1 个状态动作值最大的动作, 因此在实际对局过程中更容易被对手利用. 同时由于 DMC 训练过程中的高方差原因, 较小的值扰动也可能造成较大的策略差异, 从而造成策略质量较大的变化.

为了有效解决训练时间问题, 考虑到在常见的扑克博弈中, 存在大量的已有知识或领域知识, 因此如果能够将现有的先验知识融入算法的训练过程, 将大大提升算法的训练效率. 为此文献 [19] 提出一种暖启动(warm start)方法, 该方法针对反事实遗憾最小化算法进行暖启动, 通过已有策略, 赋予在每个信息集中的动作一个合适的反事实遗憾值, 从而实现对于策略求解的加速计算. 然而, 暖启动方法需要获取整个博弈信息, 从而进行期望值与遗憾值的计

算，因此对于大规模扑克博弈需要进行大幅度的状态与动作的约简，而这对于斗地主、掼蛋等国内主流扑克博弈难以实现，相关方法较难直接应用.

因此本文提出了一种软深度蒙特卡洛（soft deep Monte Carlo, SDMC）方法，对以掼蛋为代表的国内扑克类博弈进行求解. 首先针对 DMC 方法需要大量训练时间的问题，提出通过软启动（soft warm start）方式，结合已有策略知识，在训练过程中进行已有策略决策与 SDMC 策略模型决策的混合决策，辅助进行策略训练，提升策略收敛速度；然后在实际对战过程中依据策略模型状态动作值预测，通过软动作采样（soft action sample, SAS），缓解 DMC 方法仅选择最大值动作时，由于策略固定而易被对手利用等问题，增强策略鲁棒性. 最后，本文在掼蛋博弈中进行实验验证. 本文提出的 SDMC 方法在第 2 届“中国人工智能博弈算法大赛”取得冠军，在与 DMC 方法和第 1 届冠军等其他参赛算法进行对比实验证明了本文所提出的方法在掼蛋扑克博弈中的有效性.

1 相关背景

1.1 掼蛋扑克的博弈问题建模

掼蛋扑克博弈问题由于具有无法观测对手手牌内容、独立决策的特性，经常被建模为一个部分可观测的马尔可夫决策过程（partially observable Markov decision process, POMDP）. 在部分可观测马尔可夫决策过程中，智能体i表示智能体的编号索引，状态s表示当前实际状态. 在每一个时间点t，每一个智能体i都会观测到一个观测状态$o^i=Z(S,i)$，这里函数Z是一个观测函数. 在每次智能体观测到观测状态o^i时，智能体i都可以选择一个动作$\boldsymbol{a}$. 因此，智能体i的策略π_i可以看做一个动作观测历史（action-observation history, AOH）$\tau_t=\{o_0^i,a_0^i,o_1^i,a_1^i,\cdots o_{t-1}^i,a_{t-1}^i,o_t^i\}$的函数. 每当所有智能体执行一个动作后，当前状态$s_t$会根据环境转移函数转换到新的状态$s_{t+1}\sim\mathcal{P}(s_{t+1}|s_t,\boldsymbol{a})$，其中$\boldsymbol{a}=(a^0,a^1,\cdots)$为所有智能体的联合动作，每个智能体$i$会收到环境的奖励$r_t^i=\mathcal{R}^i(s_t,\boldsymbol{a})$. 因此当前状态下的轨迹可用$\tau_t=\{s_0,\boldsymbol{a}_0,s_1,\boldsymbol{a}_1,\cdots,s_{t-1},\boldsymbol{a}_{t-1},s_t\}$表示. 在 POMDP 中，智能体$i$目标在于最大化自身奖励$J_\pi=E_{\tau\sim P(\tau|\pi)}[R(\tau)]$，其中函数$R(\tau)=\sum_t\gamma^t r_t$为智能体收到的折扣累计奖赏，$\gamma$为累计折扣因子.

掼蛋扑克博弈，具有序贯决策特性，即每个 AOH 下最多有 1 个智能体进行决策，状态转移函数可约简为$s_{t+1}=\mathcal{P}\left(s_{t+1}|s_t,a^i\right)$，其中$a^i$为当前智能体决策. 由于扑克博弈的关键信息通常由 2 部分组成：手牌、已打出牌等当前状态信息与所有参与博弈的智能体的历史动作信息，且可以通过当前观测状态与历史动作信息对牌局进行复盘，故掼蛋博弈的 AOH 可由$\tau_t^i=\langle Z(s_t,i),H_t\rangle$进行表示，其中$h_t^i\in H_t$为智能体$i$在包含时刻$t$前的动作历史$\left\{a_0^i,a_1^i,\ldots,a_t^i\right\}$. 掼蛋扑克博弈不同于普通的 POMDP，其奖励值往往仅存在于终止状态集合S_{terminal}，因此对于非终止状态s_t，所有智能体获取到的奖励为 0，即$\forall i,s_t\notin S_{\text{terminal}},\mathcal{R}^i(s_t,\boldsymbol{a})=0$，因此累计折扣因子通常可以设置为 1.

1.2 相关研究进展

本节首先介绍现有扑克类博弈的求解方法，并分析各个方法的优缺点；其次着重介绍在斗地主中的最新方法，从而更好地介绍本文提出的 SDMC 方法.

1.2.1 扑克类游戏求解方法

扑克类游戏作为一种天然的非完美信息博弈，已经具有悠久的研究历史.

针对竞争型的扑克类游戏，如德州扑克，通常采取求解纳什均衡的方式. 其中为代表的 CFR[9] 采用自博弈的方式进行训练. 在训练的每一轮中，个体与上一轮训练出的策略进行对抗，并依靠遍历整棵博弈树的方式计算策略的遗憾，通过最小化遗憾的方式最终求解博弈的纳什均衡. 但受制于 CFR 的遍历过程，随着博弈规模的增加，遍历整棵博弈树需要极大的时间与空间，因此 MCCFR[10] 方法采取采样的方式更新博弈策略的遗憾，降低算法需求. 虽然 CFR 类的方法在以德州扑克为代表的博弈中取得惊人的效果，但是其方法限制了其在大规模环境下的应用，往往需要结合博弈约简（abstraction）[20] 方法，降低博弈树规模. 因此难以处理如掼蛋、斗地主等非完美信息博弈.

而对合作型的扑克类游戏如 Hanabi，则有着更为多样的求解范式. 不同于竞争型博弈下致力于求解纳什均衡而将其看作一个优化问题，合作型博弈也可以建模为一类学习问题[21]. 其中贝叶斯动作编码器（Bayesian action decoder, BAD）[22-23] 方法在 Hanabi 中取得了最为理想的成果. BAD 方法使用深度强化学习的方式在公共信念中探索合适的策略，但此类方法仅面向纯合作类场景选取确定性动作，无法简单适用到如掼蛋这样具有竞争合作的混合场景中.

针对掼蛋、斗地主等牌类博弈面对状态、动作空间复杂不易求解等问题，You 等人[24] 提出通过一种组合 Q 网络（combinatorial Q-network）的方式，将决

策过程分为组牌和出牌2个步骤. 但组牌过程耗时巨大, 不利于在大规模环境下的训练.DeltaDou方法[25]通过贝叶斯推断的方式推理对手的卡牌, 之后采用类似于AlphaZero的方式进行蒙特卡洛树搜索, 从而对策略进行训练, 但仍需约20天的训练时间才能在斗地主中达到专家水平.

1.2.2 深度蒙特卡洛方法

DMC方法是由Zha等人[8]提出, 是应用在斗地主环境下DouZero AI系统中的核心算法. DMC方法考虑到斗地主等扑克游戏动作不易约简且动作之间存在相似关系的特点, 通过将动作进行编码与状态一同作为神经网络的输入, 解决现有其他方法在大规模具有相似动作的空间下不适用的问题. 具体来讲, DMC方法尝试训练神经网络V, 使其输出值与实际的值函数尽可能地相近:

$$\theta^* = \arg\min_{\theta} \left\| Q\left(\tau_t^i, a^i\right) - V_\theta\left(\tau_t^i, a^i\right) \right\|, \tag{1}$$

其中θ为神经网络V的参数, 且

$$Q\left(\tau_t^i, a^i\right) = E_{\tau\sim P\left(\tau|\pi,\tau_t^i,a^i\right)}[R(\tau)] \tag{2}$$

为在当前动作观测历史τ_t^i下, 选择动作a^i后, 依据当前策略π所能获得的期望奖励值.

在训练过程中, DMC方法采用ϵ-贪心的策略选择方法, 即给定神经网络和参数V_θ, 训练过程中选择的策略

$$\pi_\epsilon\left(\tau_t^i, a^i\right) = (1-\epsilon) I\left(a^i == a^*\right) + \frac{\epsilon}{|A|}, \tag{3}$$

其中$|A|$为当前可选动作数量, $a^* = \arg\max_{a'} V\left(\tau_t^i, a'\right)$, $I(\cdot)$为指示函数(indicating function)且仅在参数为真时结果为1, 否则为0, arg max函数选择值最大的第1个动作.

在实际博弈过程中, DMC方法直接选择值函数最大的动作, 即在AOH选择τ_t^i, 可选动作集合A下, 策略为

$$\pi\left(\tau_t^i, a^i\right) = I\left(a^i == \arg\max_{a'} V\left(\tau_t^i, a'\right)\right). \tag{4}$$

2 软深度蒙特卡洛方法

本节将介绍SDMC方法, SDMC方法包含软启动与软动作采样2个过程, 解决现有方法在以掼蛋为例的扑克博弈中的问题. 同时, 为了更好地进行深度学习训练, 本文亦创新性地提出了一种针对深度学习的掼蛋扑克博弈编码方法.

2.1 软启动蒙特卡洛方法

DMC方法将动作观测历史与可选择动作进行结合, 作为神经网络输入, 通过蒙特卡洛采样方式对值函数进行拟合, 采样策略根据神经网络预测值采用ϵ-贪心的策略.

由于DMC方法采用随机网络进行初始化, 再通过自博弈的方式不断自我对战产生样本, 并进行更新. 因此在训练初期DMC自博弈产生的博弈轨迹样本的值更多是面对对手使用随机策略时的动作值, 由此产生出的策略往往并不具有实用价值, 只是训练迭代过程中的中间策略, 为后续更强策略的训练提供基础.

为了加快策略的训练过程并尽量降低因加速训练过程而产生的影响, 本文提出了软启动DMC方法, 通过软启动的方式进行训练, 借助已有策略, 尽量加速训练过程.

具体的, 对于已有策略π_{E}, 软启动DMC尝试融合借鉴预训练方法. 常见的预训练通过训练一个神经网络模型V_θ, 使得其输出预测值与已有策略值函数尽可能相近, 即

$$\theta^* = \arg\min_{\theta} \left\| Q_{\mathrm{E}}\left(\tau_t^i, a^i\right) - V_\theta\left(\tau_t^i, a^i\right) \right\|, \tag{5}$$

其中$Q_{\mathrm{E}}\left(\tau_t^i, a^i\right)$为已有策略的值函数.

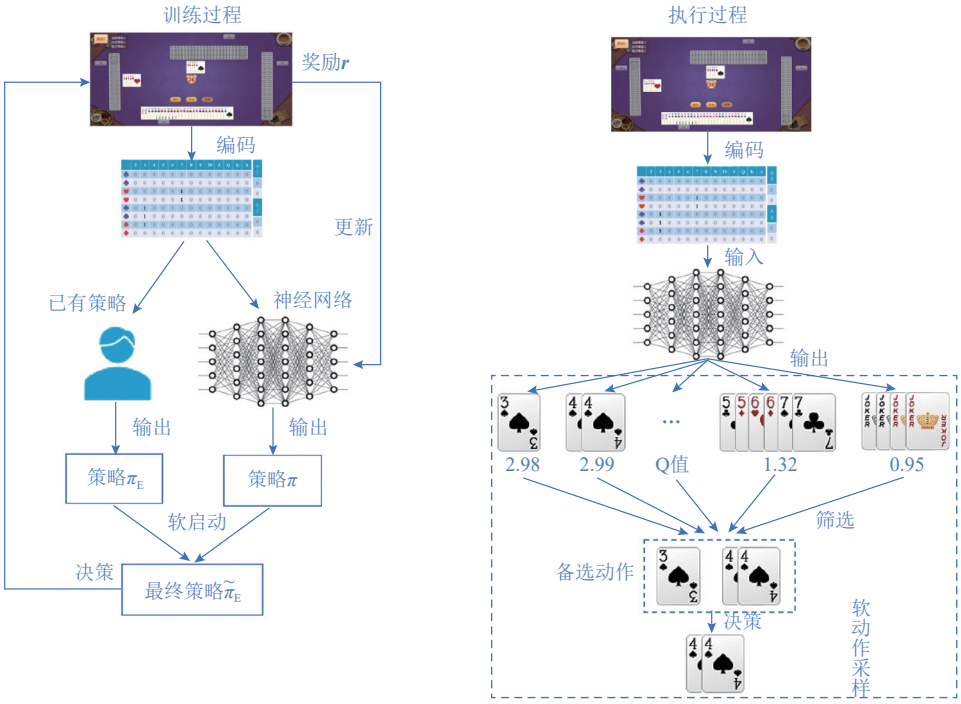
但是已有策略的值函数由于掼蛋扑克博弈规模过大通常难以直接获取, 因此可以通过蒙特卡洛采样的方式进行自博弈模拟评估:

$$Q_{\mathrm{E}}\left(\tau_t^i, a^i\right) = E_{\tau\sim P\left(\tau|\pi_{\mathrm{E}},\tau_t^i,a^i\right)}[R(\tau)]. \tag{6}$$

但由于已有策略具有先验知识, 有较多的动作并不会主动选择, 故直接自博弈评估时可能存在较多的动作没有给出评估值, 从而出现如过估计[26]等问题. 而通过在已有策略π_{E}中加入ϵ-贪心的方式可以部分缓解该问题, 但仍面临若ϵ设置较大, 则自博弈的评估并非已有策略值; 而若ϵ设置较小, 则由于探索样本比例较低, 需要大量的训练才可进行较好的拟合, 出现与最初降低训练时间需求的初衷相违背的问题.

考虑到由于已有策略并非一定最优, 经过初始化过后仍需采取自博弈的方式进行训练, 因此软启动考虑在训练过程中融入已有策略而非如文献[19]直接去拟合已有策略方式, 如图1(a)所示. 这样既融入了已有策略对当前模型训练进行加速, 同时又避免了普通的暖启动方法所面临的过估计等问题.

具体来讲, 软启动结合已有策略与当前模型生成策略的自博弈评估值$\widetilde{Q}_{\mathrm{E}}\left(\tau_t^i, a^i\right)$代替式(5)中的已



(a) SDMC训练过程 (b) SDMC实际对抗时的执行过程

Fig. 1 Overview of SDMC

图 1 SDMC 的整体流程

有策略评估值$Q_E(\tau_t^i,a^i)$，即

$$\theta^*=\arg\min_\theta\left\|\tilde{Q}_E(\tau_t^i,a^i)-V_\theta(\tau_t^i,a^i)\right\|. \tag{7}$$

其中软启动所采用的评估值由策略$\tilde{\pi}_E$生成，$\tilde{\pi}_E$结合了式(4)中当前模型生成的策略π与已有策略π_E:

$$\tilde{\pi}_E(\tau_t^i,a^i)=\frac{\epsilon}{|A|}+(1-\omega-\epsilon)\pi(\tau_t^i,a^i)+\omega\pi_E(\tau_t^i,a^i), \tag{8}$$

为混合了 2 种决策模型的ϵ-贪心策略，其中权重ω为软启动参数，可随着训练的进行而衰减.

具体算法流程如算法 1 所示.

算法 1. 软启动蒙特卡洛.

初始化：初始化经验缓存$\{B_i\}_{i=1}^n$与经验缓存$\{D_i\}_{i=1}^n$为空，其中n为智能体数量；随机初始化 SDMC 神经网络$\{V_i\}_{i=1}^n$的参数θ_i.

① for *episode* = 1 to *max_episodes*

② for $t=0$ to T

③ $i\leftarrow$当前行动智能体编号；

④ 智能体i观测到动作观测历史τ_t^i；

⑤ 由式(8)计算软启动策略$\tilde{\pi}_E^i$；

⑥ 选取动作$a^i\sim\tilde{\pi}_E^i$；

⑦ 存储样本$\{\tau_t^i,a^i\}$至经验缓存B_i；

⑧ end for

⑨ 获得环境奖励$\boldsymbol{r}=(r_1,r_2,\ldots,r_n)$；

⑩ for $i=1$ to n

⑪ for $\{\tau_t^i,a^i\}$ in B_i

⑫ 存储样本$\{\tau_t^i,a^i,r_i\}$至经验缓存D_i；

⑬ end for

⑭ 清空经验缓存B_i；

⑮ while $D_i.length > batch\_size$

⑯ 从D_i采样并更新神经网络V_i；

⑰ end while

⑱ end for

⑲ end for

⑳ 输出：神经网络模型$\{V_i\}_{i=1}^n$.

对于掼蛋这类大规模扑克博弈，可以通过 TorchBeast[18] 框架进行并行训练. 具体对于算法 1 来说，每一个 actor 执行算法 1 中 ①～⑭步，并在每次循环开始之前与 learner 同步模型；learner 执行⑮～⑰步.

2.2 软动作采样

传统 DMC 方法在使用模型决策时，一般选择最大值的动作，即对于动作观测历史τ_t^i和可选动作集合A，由式(4)选择动作. 但由于仅选择最大值的动作容易受到微小扰动的干扰，如训练方差等，导致策略大幅度变化，因此评估值较为接近的动作都有可能是最好的动作，且在实际使用过程中完全确定性的策略较易被对手猜测出自身手牌等信息，因此采用带有软动作采样(soft action sample, SAS)的动作选择

方式, 流程如图 1(b)所示, 在保证所选动作在当前模型的评估下评估值变化不大的前提下, 通过舍弃可选动作集合中值较低的动作, 保留评估值与最大值接近的备选动作, 构造备选动作集合 $\hat{A}$, 并在备选动作集合$\hat{A}$中对每一个动作的值进行 softmax 处理, 按比例分配被选择的概率:

$$P\left(a^i|\tau_t^i,\theta\right)=\mathrm{e}^{V_\theta\left(\tau_t^i,a^i\right)}/\sum_{a'\in\hat{A}}\mathrm{e}^{V_\theta\left(\tau_t^i,a'\right)},\tag{9}$$

最终根据概率分布$P\left(a|\tau_t^i,\theta\right)$选择动作. 其中备选集合$\hat{A}$的选择方式可通过设定最低阈值的形式, 即对于阈值$v_{\tau_t^i}$, 备选集合$\hat{A}$为

$$\hat{A}=\left\{a^i|\forall a^i\in A,V_\theta\left(\tau_t^i,a^i\right)\geqslant v_{\tau_t^i}\right\}.\tag{10}$$

对于阈值的选择需要保证其与最大值尽量接近, 可根据当前所有动作的值的分布选择. 具体而言, 可以通过设置较小的阈值权重ω'选择$v_{\tau_t^i}$:

$$\begin{aligned}v_{\tau_t^i}=&\max_{a^i}V_\theta\left(\tau_t^i,a^i\right)-\omega'\left(\max_{a^i}V_\theta\left(\tau_t^i,a^i\right)-\min_{a^i}V_\theta\left(\tau_t^i,a^i\right)\right)=\\&(1-\omega')\max_{a^i}V_\theta\left(\tau_t^i,a^i\right)+\omega'\min_{a^i}V_\theta\left(\tau_t^i,a^i\right).\end{aligned}\tag{11}$$

原则上, 阈值的选择应保证将所有最优动作筛选出来, 并摒弃所有非最优动作. 若能精确估计所有动作的Q值, 则权重ω'应设为 0, 即仅选择最优的动作. 但由于训练过程中的采样带来的方差扰动, 使得最优动作的Q值并非准确值, 故而需采用较小的权重. 随着训练过程的增加, Q值愈发准确, 可适当降低权重大小.

软动作采样算法流程如算法 2 所示.

算法 2. 软动作采样.

输入: 决策模型V_θ, 动作观测历史τ_t^i, 可选动作集合A, 阈值权重ω';

输出: 模型最终选择动作a^i.

① 由式(11)计算阈值v;

② $\hat{A}\leftarrow\left\{a'|\forall a'\in A,V_\theta\left(\tau_t^i,a'\right)\geqslant v\right\}$;

③ $\boldsymbol{P}\leftarrow(0,0,\ldots,0)$; /*初始化每个动作的概率为 0*/

④ for a' in $\hat{A}$

⑤ 由式(9)计算每个动作的概率$P(a')$;

⑥ end for

⑦ $a^i\sim\boldsymbol{P}$; /*根据概率分布$\boldsymbol{P}$采样一个动作a^i*/

⑧ 返回结果 a^i.

2.3 掼蛋扑克博弈编码框架

对状态信息进行编码是深度强化学习在扑克环境中进行应用的重要组成部分, 本节从 DouZero[8] 针对斗地主的编码方法出发, 根据掼蛋扑克游戏与斗地主的不同, 创新性地提出一种适用于掼蛋扑克游戏的编码框架.

对于掼蛋扑克牌局状态的编码应至少包含 3 部分: 私有信息、当前可出牌与公共信息. 私有信息一般则为自己的手牌; 当前可出牌包含所有可能出牌动作; 公共信息包含出牌记录与对局信息, 牌局信息指如掼蛋中的牌局等级信息、当前其余玩家手牌数量等当前对局的信息. 在此基础上可以添加额外的信息辅助深度网络进行训练.

状态的编码均采用 1 位有效编码的形式. 手牌信息根据游戏使用的牌数量构建不同大小的空矩阵, 若使用n副牌则构建$n\times(4\times13+2)$大小的矩阵, 其中$4\times13+2$表示 1 副牌, 4 为花色索引, 13 为点数索引, 2 表示大小王, 当拥有某张手牌时, 对应位置的矩阵数值置为 1. 对于标准掼蛋规则, 由于使用 2 副牌, 因此$n=2$, 具体卡牌编码如图 2“卡牌表示”部分所示.

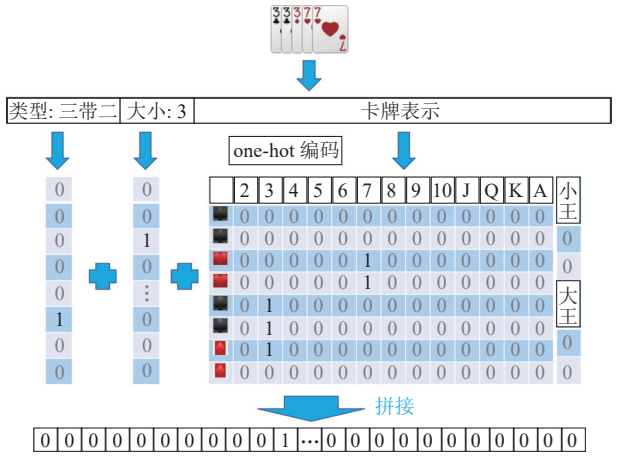


Fig. 2 An example of information encoding in GuanDan poker game

图 2 掼蛋扑克信息编码示例

出牌动作的编码类似手牌编码方式, 分别对出牌动作的类型、大小与所使用的牌进行编码. 对类型、大小的编码可解决相同出牌具有不同出牌类型与大小情况, 如掼蛋中部分具有逢人配的顺子等, 以及区分在编码出牌记录时无出牌记录的 0 填充编码与“过牌”(PASS)在编码时的区别, 并提供额外的信息辅助神经网络处理, 如图 2 所示.

对于出牌记录的处理主要包括对出牌动作的编码与出牌动作结构的组织. 对于 SDMC 与 DMC 等方法, 出牌记录信息会被输入到循环神经网络, 因此可采取序列式结构对出牌进行组织, 即从下家出牌到智能体自己出牌为止 1 轮的出牌编码进行拼接.

3 实验与结果分析

本节对本文提出的 SDMC 方法进行实验分析,

使用掼蛋扑克环境，衡量 SDMC 方法中软启动的加速训练效果，并分别与第 1 届、第 2 届“中国人工智能博弈算法大赛”的参赛算法对比，证明 SDMC 方法的有效性.

3.1 掼蛋扑克介绍

掼蛋是国内一种广泛流传的扑克类博弈. 掼蛋博弈使用 2 副扑克牌，共 108 张牌，采取 2 对 2 的模式对抗，其中每个博弈玩家与对家为 1 支队伍进行对抗. 本文采用第 2 届“中国人工智能博弈算法大赛”中的掼蛋规则，下面简要介绍.

1) 等级. 1 局掼蛋博弈可以分为若干小局，每小局根据双方队伍等级中最高的决定当前牌局等级，并依据小局胜负情况更新双方的等级. 掼蛋对局中初始双方等级为 2，之后依次为 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A.

2) 升级. 每小局对战结束后，仅第 1 个出完牌的玩家(称为上游)所在队伍可以升级，升级数依据队友出完牌的顺序决定：若队友第 2 个出完牌(称为二游)，则升 3 级；第 3 个出完牌(称为三游)，则升 2 级；若队友最后 1 个出完牌(称为下游)，则只升 1 级.

3) 获胜条件. 当一方队伍到达等级 A，并且队伍中一人获得上游，另一人获得二游或者三游.

4) 特殊牌. 掼蛋中和当前牌局等级相同的牌为级牌，其中红桃级牌称为逢人配，可当作任意牌与其他花色牌组合使用.

5) 牌型. 掼蛋中牌型如下：单张、对子、三连对、三同张、二连三、三带二、顺子、同花顺、炸弹、天王炸. 其中炸弹张数多者为大，同样张数按照点数排序，同花顺大于任意不超过 5 张的炸弹，天王炸为 2 张大王和 2 张小王为最大牌型.

6) 牌点大小. 掼蛋中牌点从大到小依次为：大王、小王、级牌、A、K、Q、J、10、9、8、7、6、5、4、3、2. A 在搭配成三连对、二连三、顺子、同花顺时，可视作 1.

7) 进贡、还贡. 从第 2 小局开始，由上一轮的下游向上游进贡，挑选 1 张除红心级牌之外最大的牌给上游. 上游选择 1 张不大于 10 的牌给下游还贡. 如果上一局出现一方队伍获得三游和下游，则队伍 2 人均向对方队伍分别进贡和接受还贡. 如进贡方有 2 个大王，则可以不进贡.

掼蛋由于使用 2 副牌，因此具有更高的求解复杂度，对算法训练效率提出了更大挑战. 仅第 1 轮发牌后的信息集的数量级约为 10^{20}，远超斗地主等扑克博弈(斗地主的第 1 轮发牌后数量级约 10^{14}，如考虑去除斗地主的花色影响约为 10^{8}). 同时考虑到各个玩家拥有更多的手牌产生的指数级增长的可选动作以及等级、逢人配等因素，掼蛋实际信息集数量与大小远超斗地主等扑克博弈.

3.2 实现细节

本节主要描述了基于 SDMC 的掼蛋智能体实现细节，包含整体架构、状态编码方式与奖励设计 3 个方面.

1) 整体架构细节

基于 SDMC 的掼蛋智能体主要由 2 部分组成：掼蛋贡、还牌规则决策模块与出牌的 SDMC 决策模块. 其中所使用的掼蛋贡、还牌规则决策模块采用第 1 届“中国人工智能博弈算法大赛”的冠军规则. 出牌的 SDMC 决策模块中，式(11)中 $\omega' = \frac{1}{500}$.

SDMC 的网络结构与文献 [8] 相似，均采用 LSTM 网络处理历史动作，通过 6 层全连接网络输出动作评估值.

2) 状态编码方式

掼蛋编码方式采用 2.3 节编码框架，每一部分的编码大小如表 1 所示.

Table 1 State Representation of GuanDan Games

表 1 掼蛋环境状态编码

类型	含义	one-hot 编码长度
出牌动作	卡牌的矩阵表示	108
	类型的矩阵表示	10
	大小的矩阵表示	15
私有信息	手牌的卡牌矩阵表示	108
	手牌中逢人配数量	3
公共信息	当前等级	13
	其他玩家剩余手牌	108
	上家已出卡牌矩阵表示	108
	对家已出卡牌矩阵表示	108
	下家已出卡牌矩阵表示	108
	最近一次的出牌动作	133
	上家最近一次出牌动作	133
	对家最近一次出牌动作	133
	下家最近一次出牌动作	133
	上家剩余手牌数量	28
	对家剩余手牌数量	28
	下家剩余手牌数量	28
	最近 4 轮出牌的联合表示	532

3) 奖励设计

如 3.1 节介绍的，掼蛋胜负取决于队伍积分是否超过 A，因此通过判断大局胜利的方式给予 2 队智能

体奖励, 可能会导致较差的小局内策略获得正奖励, 因此在掼蛋环境训练过程中, 当小局结束时, 对双方队伍给定奖励, 奖励分配方式如表 2 所示, 1—1—2—2 表示完牌顺序分别为队伍 1、队伍 1、队伍 2、队伍 2 的选手.

Table 2 Reward Functions Designed in GuanDan Games

表 2 掼蛋环境训练中的奖励设计

完牌顺序	队伍 1 获得奖励	队伍 2 获得奖励
1—1—2—2	+3	−3
1—2—1—2	+2	−2
1—2—2—1	+1	−1
2—1—1—2	−1	+1
2—1—2—1	−2	+2
2—2—1—1	−3	+3

设计的奖励方式基本与掼蛋晋级方式相同, 但当掼蛋遇到等级 A 时有所不同, 由于当一方队伍到达等级 A 时想要获胜至少要有一个队友第 1 个完牌并且另一个队友不能最后一个完牌, 因此若以队伍 1 达到等级 A 为例, 完牌顺序 1—1—2—2 与完牌顺序 1—2—1—2 相同, 剩余 4 种完牌顺序奖励也相同. 考虑到 SDMC 方法会尽量选择高评估值的动作, 因此不对奖励函数进行修正很大程度上并不会影响训练策略的正确性, 故智能体在训练过程中并未对等级 A 进行特殊处理.

3.3 实验结果

本节通过与不同算法的对比验证 SDMC 的效果. 具体而言, 分别与第 1 届“中国人工智能博弈算法大赛”的冠军(1st Champion)与前 2 届 16 强(1st Top 16 和 2nd Top 16)进行比较. 实验中选取 2 种算法作为 2 支队伍, 采取 2 种评估指标, 分别评估对战双方团队的胜利与净胜小分情况.

我们首先验证了经过 30 天训练的 SDMC 与其他方法的最终胜率与净胜小分, 每场对战均进 500 次, 并重复验证 5 次, 最终胜率与标准差如表 3 所示. 表 3 中的数据表示算法 1 对阵算法 2 时的胜率, 如 SDMC 对抗 2nd Top 16 胜率为 97.5%, 算法的排名顺序按照击败(胜率>50%)其他算法的顺序进行排序.

我们看到 SDMC 对战第 1 届比赛的冠军以及其他算法胜率均大于 90%, 对抗 2nd Top 16 和 1st Top 16 时胜率甚至分别大于 97.5% 和 100%, 因此可以认为 SDMC 的效果显著高于其他算法. 同时我们也对比了不使用 SAS 的 SDMC(SDMC-无 SAS)和使用 SAS 的 SDMC 的效果, 对于测试的, 使用 SAS 能够提升一定的 SDMC 的效果.

同时对于净胜小分, 我们详细列出了各种算法之间对战过程中双方小分的获得情况与净胜分, 如表 4 所示. 可以看到 SDMC 在对战其他参赛方法的时候具有很高的 3 分得分率, 即在掼蛋中有很高的双上(队伍分别以上游和二游完牌)概率, 在对战 1st Champion 时达到 44.3%, 对战 2nd Top 16 时达到 48.4%, 对战 1st Top 16 时达到 68.4%, 因此对战过程中的净胜分非常高, 对战 1st Champion 时平均每局净胜达到 4 955.6 分. 同时观察到 SDMC 与 SDMC-无 SAS 在对抗过程中无论是 3 分、2 分、1 分的占比还是净胜分均是 SDMC 更胜一筹, 且 SDMC 在对战除 2nd Top 16 的对手时, 净胜分均高于 SDMC-无 SAS.

最后为了验证本文提出的软启动方法的实验效果, 我们比较了 DMC 方法、已有策略预训练的方法(基于策略启动的 DMC)以及我们提出的采用软启动的 SDMC 方法对抗 1st Top 16 时的胜率与平均每小局净胜分曲线, 如图 3 所示, 其纵坐标分别为对抗的胜率与平均每小局净胜分, 横坐标为训练所用的时间步. 对于每种方法, 每次测试记录 200 局与 1st Top 16 的胜率与每小局净胜分结果, 测试 3 次, 图 3 绘制了测试的平均值与标准差. 其中基于策略启动的 DMC

Table 3 Winning Percentage of Different Algorithms Against Each Other

表 3 不同算法对抗胜率 %

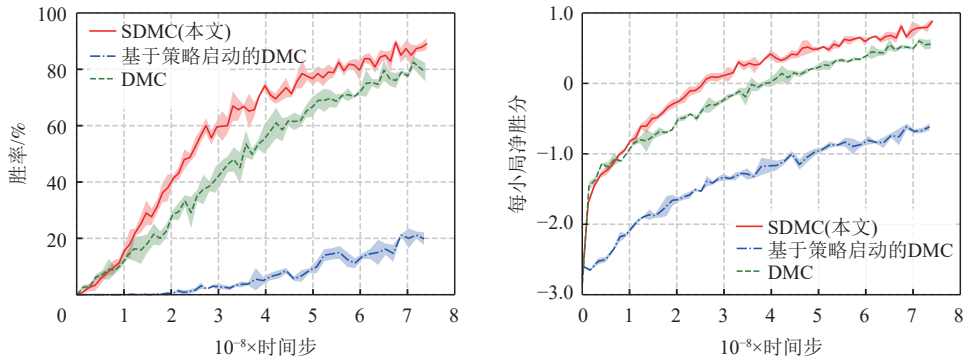
算法 1	算法 2				
	SDMC(本文)	SDMC-无 SAS	1st Champion	2nd Top 16	1st Top 16
SDMC(本文)	\	51.0±1.0	92.1±1.0	97.5±0.7	100.0±0.0
SDMC-无 SAS	49.0±1.0	\	91.8±0.9	97.2±0.9	100.0±0.0
1st Champion	7.9±1.0	8.2±0.9	\	62.2±1.0	97.3±0.6
2nd Top 16	2.5±0.7	2.8±0.9	37.8±1.0	\	94.9±0.8
1st Top 16	0.0±0.0	0.0±0.0	2.7±0.6	5.1±0.8	\

注: “\”表示 2 种相同算法之间不对抗, 没有对抗胜率.

Table 4 Score of Different Algorithms Against Each Other

表 4 不同算法对抗时得分

算法	对战算法	3 分占比/%	2 分占比/%	1 分占比/%	净胜小分（500 局）
SDMC-无 SAS	SDMC（本文）	23.0±0.7	9.8±0.3	13.7±0.6	−95.6±233.9
1st Champion		12.3±0.2	6.0±0.4	13.9±0.6	−495.6±126.5
2nd Top 16		8.5±0.7	5.7±0.6	12.8±0.4	−5869.4±96.7
1st Top 16		2.0±0.2	1.7±0.2	7.5±0.2	−7415.2±47.1
SDMC（本文）	SDMC-无 SAS	23.4±0.6	11.3±0.4	15.6±0.8	95.6±233.9
1st Champion		12.4±0.4	6.2±0.4	13.8±0.3	−4907.0±80.0
2nd Top 16		8.8±0.4	5.7±0.2	12.2±0.4	−5885.4±93.8
1st Top 16		1.9±0.2	1.8±0.2	7.4±0.3	−7387.4±14.9
SDMC（本文）	1st Champion	44.3±1.1	9.6±0.2	13.8±0.7	4955.6±126.5
SDMC-无 SAS		44.2±0.8	9.8±0.3	13.7±0.6	4907.0±80.0
2nd Top 16		23.3±0.7	7.2±0.3	16.3±0.6	−1213.0±128.4
1st Top 16		8.7±0.4	3.8±0.4	14.4±0.7	−5997.4±140.3
SDMC（本文）	2nd Top 16	48.4±0.4	12.0±0.3	12.6±0.3	5869.4±96.7
SDMC-无 SAS		48.8±0.7	12.0±0.2	12.6±0.4	5885.4±93.8
1st Champion		30.2±0.4	9.2±0.4	13.7±0.4	1213.0±128.4
1st Top 16		10.1±0.3	5.1±0.3	15.7±0.5	−5367.6±63.4
SDMC（本文）	1st Top 16	68.4±0.5	13.5±0.6	7.0±0.1	7415.2±47.1
SDMC-无 SAS		68.1±0.5	13.9±0.5	7.0±0.3	7387.4±14.9
1st Champion		50.5±1.2	11.5±0.4	11.0±0.4	5997.4±140.3
2nd Top 16		42.6±0.7	14.2±0.4	12.4±0.3	5367.6±63.4



(a) 训练过程中的胜率变化趋势 (b) 训练过程中的净胜分变化趋势

Fig. 3 Performance of three methods against baseline in training stage

图 3 3 种方法在训练过程中对抗基准算法时的表现

采取与 DMC 和 SDMC 相同的神经网络架构与训练方式，不同之处在于基于策略启动的 DMC 每次自博弈生成轨迹是依据已有策略进行ϵ-贪心采样，训练基于策略启动的 DMC 模型. 基于策略启动的 DMC 所采用的已有策略与 SDMC 相同，均为 1st Champion.

从图 3(a)中可以看到 SDMC 相较于 DMC 在训练初始阶段取得了较高的胜率提升速度，如在胜率达到 60% 时，SDMC 需要约 2.7×10^8个时间步，而 DMC 则需要约 4×10^8个时间步，对于 60% 胜率，SDMC 仅需 DMC 的 68% 训练开销. 同样地，在图 3(b)中对于净胜小分，SDMC 仅需 2.5×10^8个时间步即可达到净胜分大于 0，而 DMC 需要约 3.8×10^8个时间步，SDMC 降低了约 35% 的时间需求.

同时可以发现，相比于 SDMC 和 DMC，基于策略启动的 DMC 训练效果不佳，原因可能正如 2.1 节中讨论的，仅通过已有策略生成数据在训练过程中

由于掼蛋的动作空间过于庞大,因此无法很好地拟合未探索动作的值,因此存在过估计问题. 而 DMC 和 SDMC 因为存在通过神经网络去决策的步骤,当存在高评估值的动作时,由于倾向于选择高评估值动作,可以有效地对于这个动作的评估值进行验证,从而更好地评估.

4 总 结

本文提出了一种针对掼蛋扑克博弈的软深度蒙特卡洛 SDMC 方法. SDMC 方法在学习过程中不仅采用了软启动方法,结合已有策略,加速模型训练过程,同时采取软动作采样,在实际对战过程中,保证选择的策略在当前模型下的评估值变化不大的情况下对动作进行采样,降低训练过程中方差带来的影响,并增加被对手利用的难度. 在掼蛋环境下的实验表明,本文所提方法 SDMC 相较于现有方法有着更高的对战胜率与净胜得分. 之后,拟从软动作采样的角度出发,依据现有模型的动作评估值,结合子博弈求解方法提升在实战环境下的策略强度,致力于得到在团体对战情况下的团队最大最小均衡等博弈论角度下的最优策略,最终实现在掼蛋等扑克博弈环境下战胜人类的职业选手.

作者贡献声明:葛振兴提出选题、研究内容,设计技术方案,撰写论文;向帅设计技术方案,采集和整理实验数据,修订论文;田品卓设计技术方案,提出指导性建议;高阳提出指导性建议,指导论文写作.

参 考 文 献

[1] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484−489

[2] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354−359

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

[4] Brown N, Sandholm T. Superhuman AI for multiplayer poker[J]. Science, 2019, 365(6456): 885−890

[5] Li Junjie, Koyamada S, Ye Qiwei, et al. Suphx: Mastering mahjong with deep reinforcement learning[J]. arXiv preprint, arXiv: 2003. 13590, 2020

[6] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning[J]. Nature, 2019, 575(7782): 350−354

[7] Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning[J]. arXiv preprint, arXiv: 1912. 06680, 2019

[8] Zha Daochen, Xie Jingru, Ma Wenye, et al. DouZero: Mastering DouDiZhu with self-play deep reinforcement learning [C] //Proc of the 38th Int Conf on Machine Learning. New York: ACM, 2021: 12333−12344

[9] Zinkevich M, Johanson M, Bowling M, et al. Regret minimization in games with incomplete information [C] //Proc of the 21st Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2007: 1729−1736

[10] Lanctot M, Waugh K, Zinkevich M, et al. Monte Carlo sampling for regret minimization in extensive games [C] //Proc of the 23rd Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2009: 1078−1086

[11] Farina G, Kroer C, Sandholm T. Online convex optimization for sequential decision processes and extensive-form games [C] //Proc of the 33rd AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2019: 1917−1925

[12] Johanson M, Bard N, Lanctot M, et al. Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization [C] //Proc of the 11th Int Conf on Autonomous Agents and Multiagent Systems. Berlin: Springer, 2012: 837−846

[13] Burch N. Time and space: Why imperfect information games are hard [D]. Edmonton: University of Alberta, 2018

[14] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529−533

[15] Qi Faxin, Tong Xiangrong, Yu Lei. Agent trust boost via reinforcement learning DQN[J]. Journal of Computer Research and Development, 2020, 57(6): 1227−1238 (in Chinese)
(亓法欣, 童向荣, 于雷. 基于强化学习 DQN 的智能体信任增强[J]. 计算机研究与发展, 2020, 57(6): 1227−1238)

[16] Zheng Bolong, Ming Lingfeng, Hu Qi, et al. Dynamic ride-hailing route planning based on deep reinforcement learning[J]. Journal of Computer Research and Development, 2022, 59(2): 329−341 (in Chinese)
(郑渤龙, 明岭峰, 胡琦, 等. 基于深度强化学习的网约车动态路径规划 [J]. 计算机研究与发展, 2022, 59(2): 329−341)

[17] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning [C] //Proc of the 33rd Int Conf on Machine Learning. New York: ACM, 2016: 1928−1937

[18] Küttler H, Nardelli N, Lavril T, et al. TorchBeast: A pytorch platform for distributed rl[J]. arXiv preprint, arXiv: 1910. 03552, 2019

[19] Brown N, Sandholm T. Strategy-based warm starting for regret minimization in games [C] //Proc of the 30th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2016: 432−438

[20] Sandholm T. Abstraction for solving large incomplete-information games [C] //Proc of the 29th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2015: 4127−4131

[21] Bard N, Foerster J N, Chandar S, et al. The Hanabi Challenge: A new frontier for AI research[J]. Artificial Intelligence, 2020, 280: 103216

[22] Foerster J, Song F, Hughes E, et al. Bayesian action decoder for deep multi-agent reinforcement learning [C] //Proc of the 36th Int Conf on Machine Learning. New York: ACM, 2019: 1942−1951

[23] Hu Hengyuan, Foerster J. Simplified action decoder for deep multi-agent reinforcement learning[J]. arXiv preprint, arXiv: 1912. 02288, 2019

[24] You Yang, Li Liangwei, Guo Baisong, et al. Combinatorial Q-learning for Dou Di Zhu [C] //Proc of the 16th AAAI Conf on Artificial Intelligence and Interactive Digital Entertainment. Palo Alto, CA: AAAI, 2020: 301−307

[25] Jiang Qiqi, Li Kuangzheng, Du Boyao, et al. DeltaDou: Expert-level Doudizhu AI through self-play [C] //Proc of the 28th Int Joint Conf on Artificial Intelligence. San Francisco: Morgan Kaufmann, 2019: 1265−1271

[26] Zahavy T, Haroush M, Merlis N, et al. Learn what not to learn: Action elimination with deep reinforcement learning [C] // Proc of the 32nd Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2018: 3566−3577

Ge Zhenxing, born in 1998. PhD candidate. Student member of CCF. His main research interests include game theory, multi-agent systems, and reinforcement learning.

葛振兴, 1998 年生. 博士研究生. CCF 学生会员. 主要研究方向为博弈论、多智能体系统、强化学习.

Xiang Shuai, born in 1997. Master candidate. Student member of CCF. His main research interests include game theory, reinforcement learning, and multi-agent systems.

向　帅, 1997 年生. 硕士研究生. CCF 学生会员. 主要研究方向为博弈论、强化学习、多智能体系统.

Tian Pinzhuo, born in 1991. PhD, lecturer. His main research interests include machine learning, meta-learning, and transfer learning. (pinzhuo@shu.edu.cn)

田品卓, 1991 年生. 博士, 讲师. 主要研究方向为机器学习、元学习、迁移学习.

Gao Yang, born in 1972. PhD, professor, PhD supervisor. Committee member of CCF. His main research interests include reinforcement learning, multi-agent systems, computer vision, and big data analysis.

高　阳, 1972 年生. 博士, 教授, 博士生导师. CCF 委员会委员. 主要研究方向为强化学习、多智能体系统、计算机视觉、大数据分析.