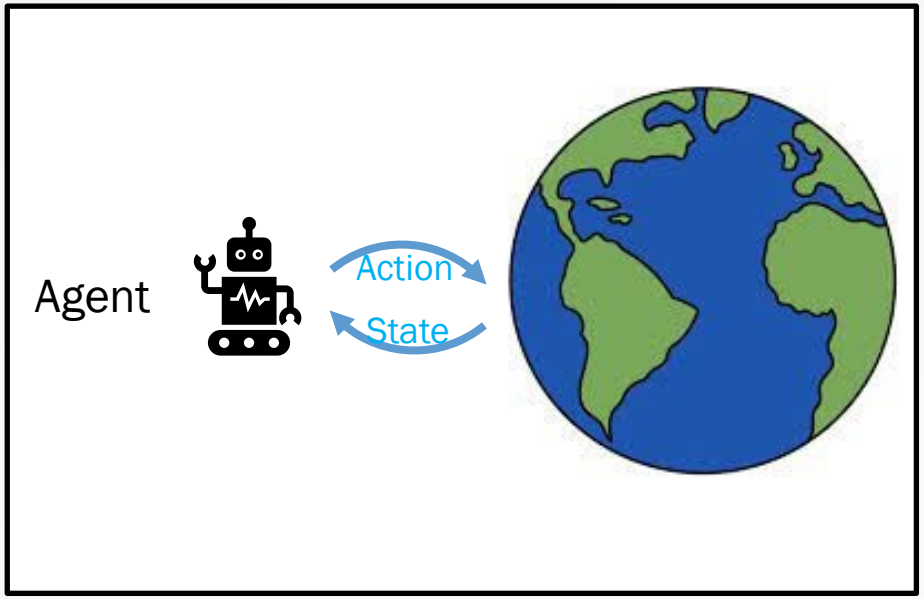
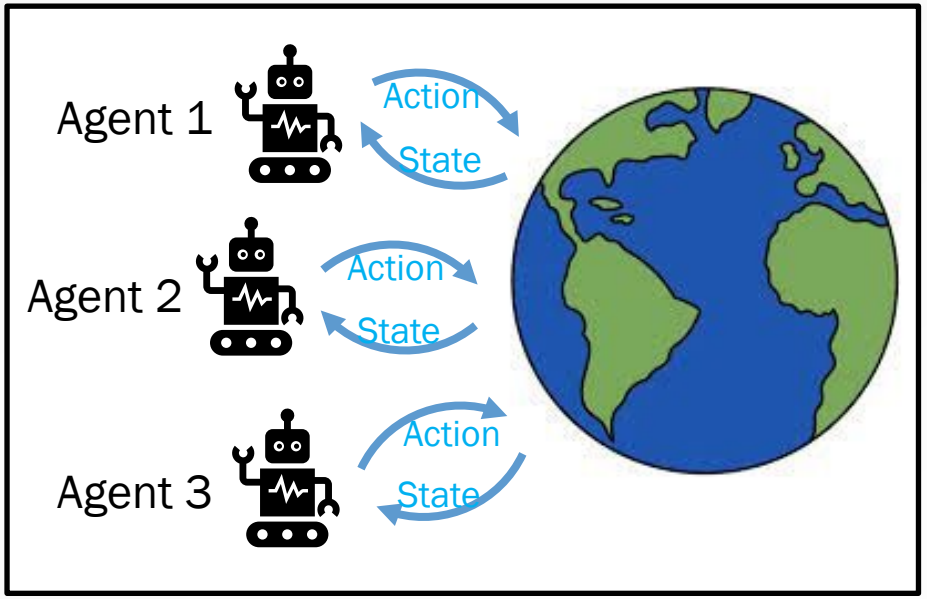
多智能体强化学习

朱圆恒
中国科学院自动化研究所

单智能体 vs 多智能体



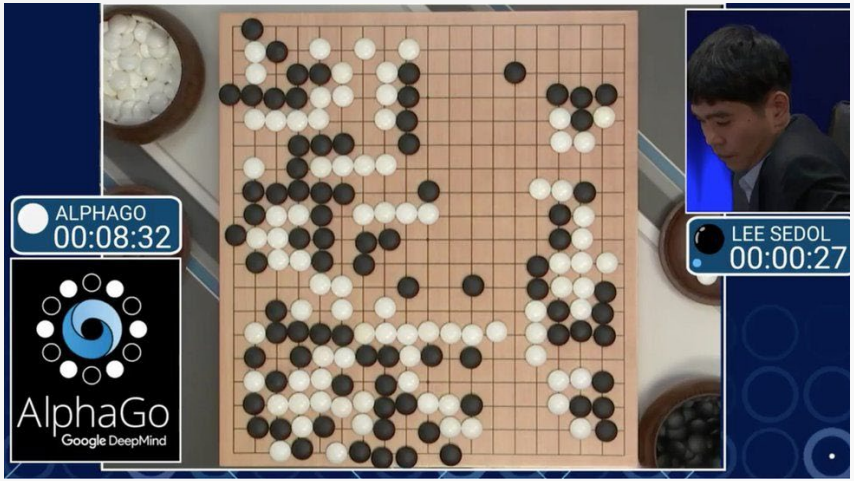
单智能体



多智能体

多智能体系统

- **多智能体系统：**在同一个环境中存在多个智能体相互交互的系统



ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

Article

Grandmaster level in StarCraft II using multi-agent reinforcement learning

https://doi.org/10.1038/s41586-019-1724-z
Received: 30 August 2019
Accepted: 10 October 2019
Published online: 30 October 2019

Oriol Vinyals[1,2]*, Igor Babuschkin[1,2], Wojciech M. Czarnecki[1,2], Michaël Mathieu[1,2], Andrew Dudzik[1,2], Junyoung Chung[1,2], David H. Choi[1,2], Richard Powell[1,2], Timo Ewalds[1,2], Petko Georgiev[1,2], Junhyuk Oh[1,2], Dan Horgan[1,2], Manuel Kroiss[1,2], Ivo Danihelka[1,2], Aja Huang[1,2], Laurent Sifre[1,2], Trevor Cai[1,2], John P. Agapiou[1,2], Max Jaderberg[1], Alexander S. Vezhnevets[1], Rémi Leblond[1], Tobias Pohlen[1], Valentin Dalibard[1], David Budden[1], Yury Sulsky[1], James Molloy[1], Tom L. Paine[1], Caglar Gulcehre[1], Ziyu Wang[1], Tobias Pfaff[1], Yuhuai Wu[1], Roman Ring[1], Dani Yogatama[1], Dario Wünsch[1], Katrina McKinney[1], Oliver Smith[1], Tom Schaul[1], Timothy Lillicrap[1], Koray Kavukcuoglu[1], Demis Hassabis[1], Chris Apps[1,3] & David Silver[1,3]*

多智能体系统

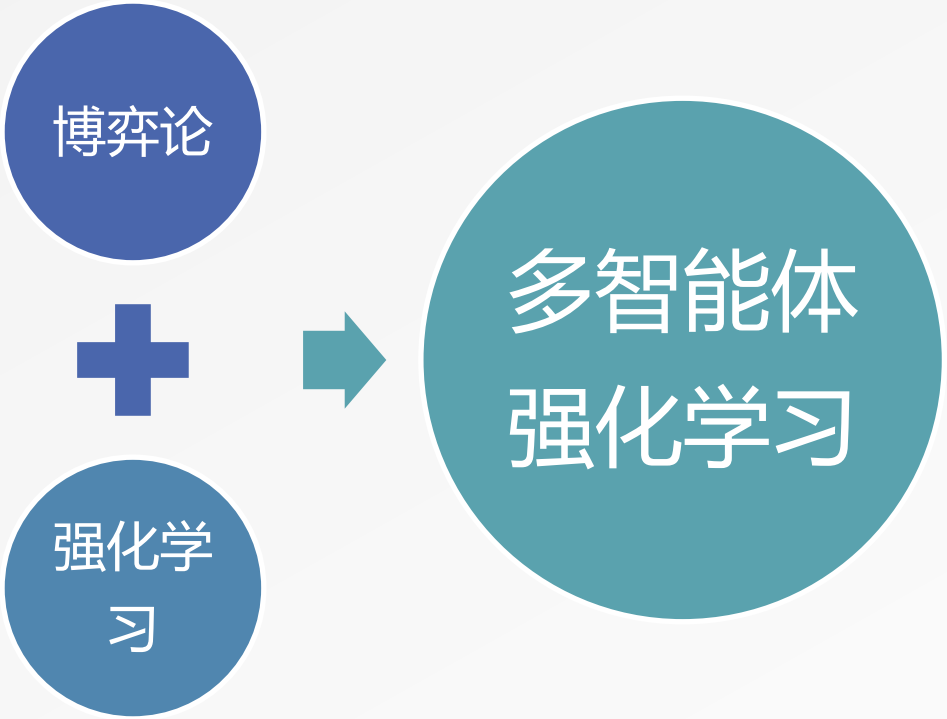
- **多智能体系统**：在同一个环境中存在多个智能体相互交互的系统

交通

股票市场

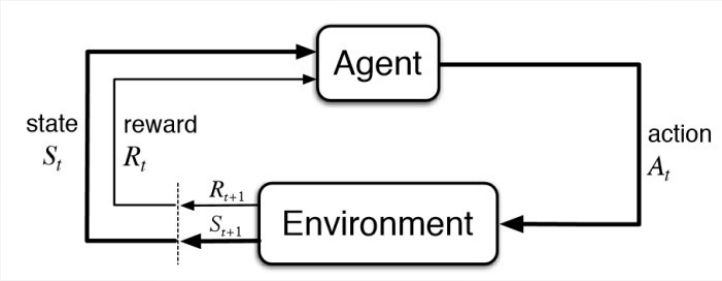
生产线

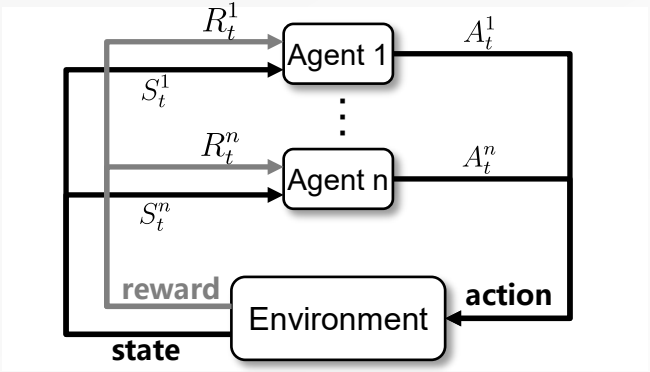
多智能体强化学习(1)



- 博弈论：研究多个自私的玩家在同一场景中的决策模型，决策内容会影响玩家各自的收益
- 强化学习：学习智能体从状态到动作映射的策略，能够最大化智能体获得的累加奖励
- 多智能体强化学习 Multi-Agent Reinforcement Learning, MARL

多智能体强化学习(2)





单智能体：马尔可夫决策过程

- 状态 $S_t, S_{t+1}, \ldots$
- 动作 $A_t, A_{t+1}, \ldots$
- 奖励 $R_t, R_{t+1}, \ldots$
- 转移 $S_{t+1} \sim P(S_t, A_t)$

多智能体：马尔可夫博弈过程

- 状态 $S_t, S_{t+1}, \ldots$
- 动作 $\{A_t^i\}, \{A_{t+1}^i\}, \ldots$
- 奖励 $\{R_t^i\}, \{R_{t+1}^i\}, \ldots$
- 转移 $S_{t+1} \sim P\left(S_t, \{A_t^i\}\right)$

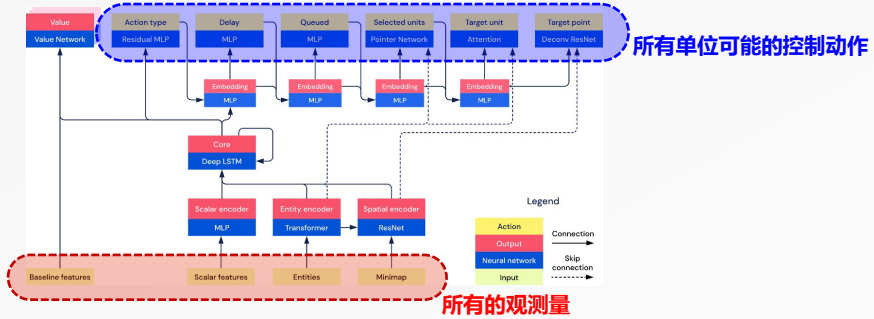
为什么要研究MARL

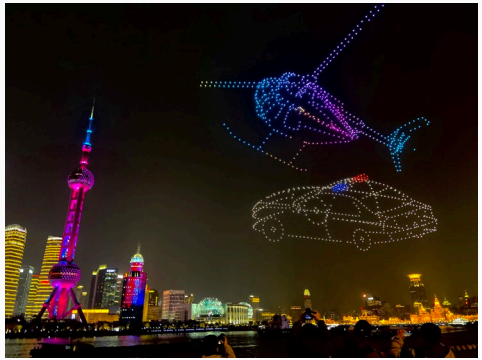
- 现实世界存在多智能体系统，需要解决多智能体系统决策问题
  - 交通，经济，市场，生产。。。
- 多智能体系统具有分布式特性，多个智能体可以共同协作完成同一任务，单个智能体计算量小，整个系统的灵活性、容错性和鲁棒性高
  - 蜂群、鱼群抵御天敌
- 多智能体通过通讯、示教、模仿等可以互相共享经验，比单个智能体学习效率更高
  - 1对1 教学 vs 班级教学

MARL分类(1)

- 集中式 centralized
  - 一个中心“大脑”，为所有智能体输出指令
  - AlphaStar
- 分布式 decentralized
  - 智能体各自学习和决策
  - 环境定义的通信约束
  - 无人机集群表演

Overview of the architecture of **AlphaStar**
[Vinyals et al. (2019) Nature]





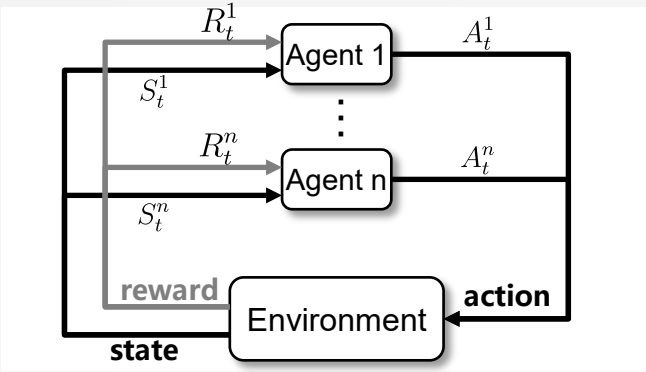
MARL分类(2)

- 对抗式：智能体互相竞争
  - **零和博弈** （围棋、德州扑克）
  - **个体奖励互为相反** $R^1 = -R^2$
- 合作式：智能体合作完成同一任务
  - **共享团队奖励** $R = R^1 = \cdots = R^n$ （生产线、无人机表演）
- 混合式：智能体需要合作与/或竞争才能实现收益最大化
  - **一般和博弈** （股票市场散户之间的合作与竞争）
  - **团队对抗：队内协作，队间竞争** （足球，Dota等团队竞技）

MARL分类(3)

- 智能体数量
  - 1个（单智能体）
  - 2个（零和，对抗）
  - 有限个
  - 无穷个

MARL挑战



多智能体

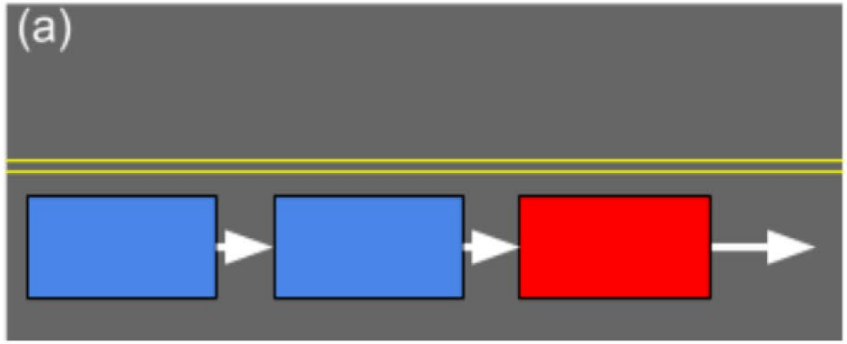
- 状态 $S_t, S_{t+1}, \dots$
- 动作 $\{A_t^i\}, \{A_{t+1}^i\}, \dots$
- 奖励 $\{R_t^i\}, \{R_{t+1}^i\}, \dots$
- 转移 $S_{t+1} \sim P\left(S_t, \{A_t^i\}\right)$

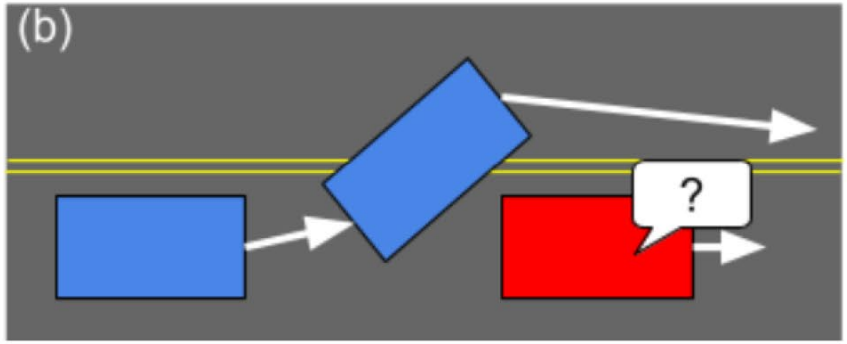
$\max R_t^i + \gamma R_{t+1}^i + \gamma^2 R_{t+2}^i + \cdots$

For all agent i

- 维数灾
  - 计算复杂性不光随智能体状态和动作维数指数增长，同时还随智能体数量呈指数增长 （$O(|S||A|^n)$）
- 学习目标
  - 智能体的回报是相互影响，无法独立优化各自目标
- 非静态性
  - 从智能体i的视角，当其它智能体$\{j\}$在学习时，包含智能体$\{j\}$的环境是在变化，那么智能体i的最优策略也是非固定的。学习环境具有非静态的特性。
- 协作学习
  - 智能体的行为会影响其它智能体，导致它人（或自己）学习过程混乱/振荡

挑战：非静态环境





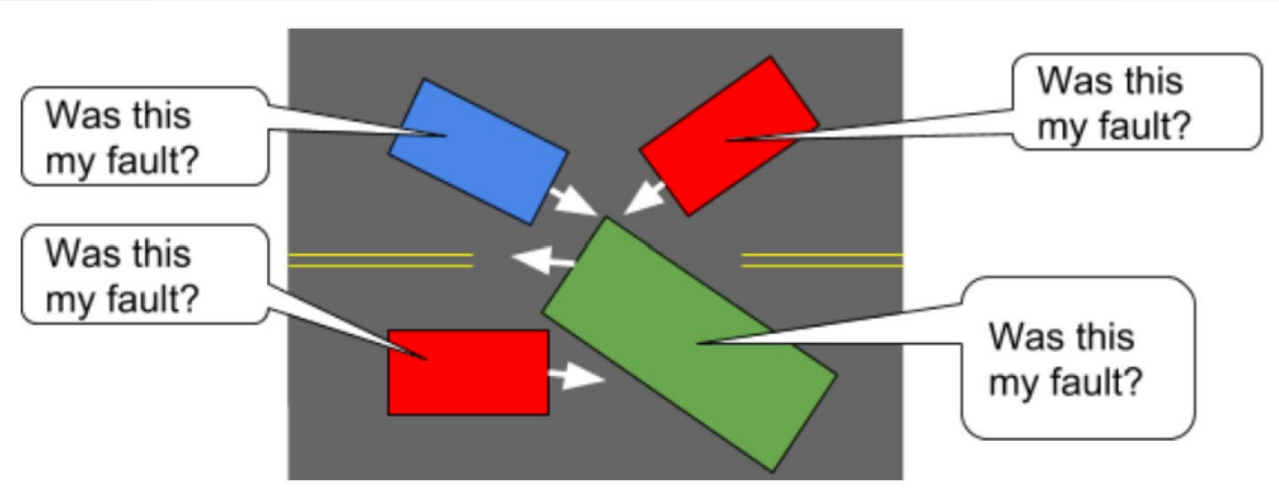
非静态环境：
初始时(a)，红色智能体学习到通过减速，调控整个交通流的速度。
一段时间后，蓝色智能体学到了超车，超过了红色智能体(b)。
红色智能体之前的学习经验在新场景下失效。

非静态问题：在多智能体系统中，智能体 i 面对的是动态的转移过程，原因是当其它玩家策略 $\pi^{-i} \neq \bar{\pi}^{-i}$ 时

$$P_{\pi^{-i}}(s'|s,a^i) \neq P_{\bar{\pi}^{-i}}(s'|s,a^i)$$

此时智能体i 的样本不再具有马尔可夫性

挑战：高方差估计



如上交通堵塞，无法判定是哪个智能体行为导致当前局面的产生。同样当堵塞解除时，也无法将一个全局的奖励归功于是哪个智能体的贡献

随机不确定性来源：

- s_t在状态转移时的随机性，P
- $\{a_t^i\}$在所有智能体动作选择时的随机性，$\{\pi^i\}$
- r_t^i在奖励生成时的随机性，R

挑战总结

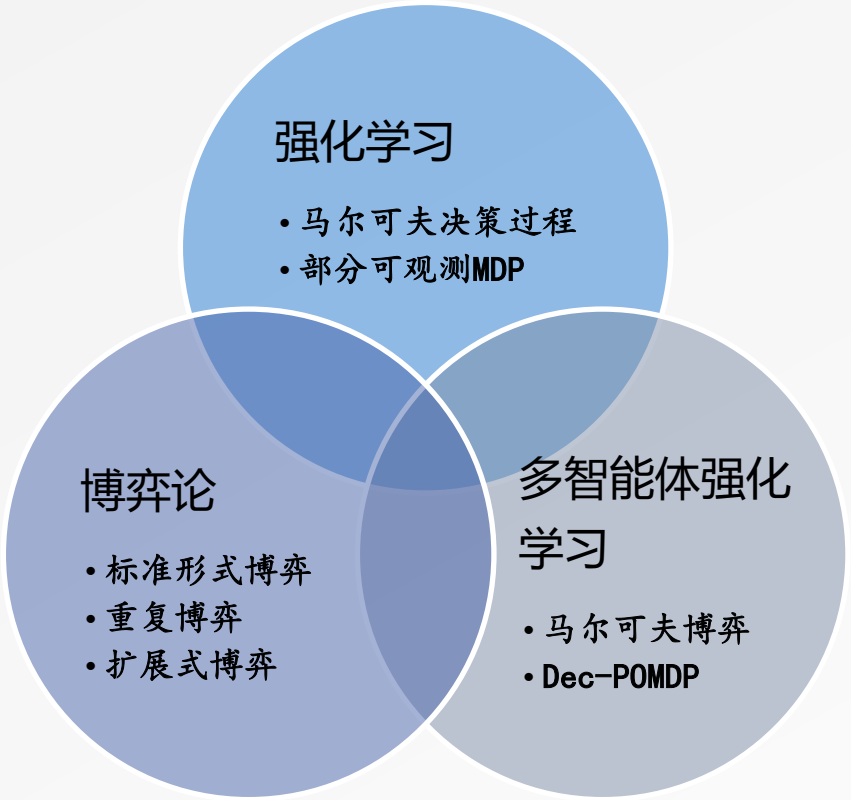
- 单智能体RL，智能体只需要根据自身的动作和对环境的影响，调整自己的策略。在多智能体MARL中，智能体要适应其它智能体的动作和学习；智能体在同一状态执行同一动作会获得不同的奖励
- MARL中智能体有可能无法获得环境全局信息（分布式性），即使是能获取全局信息，也无法预测其它智能体的行为以及环境的变化
- 信誉分配问题：难以确定是哪个智能体要对任务成功或失败负责。在智能体间如何分配奖励，以及局部奖励和全局奖励如何结合，能够加快学习速率或是收敛到全局最优解，是MARL难题

博弈模型

博弈模型

- 博弈论 Game Theory
  - 标准形式博弈 normal form game
  - 重复博弈 repeated game
- 多智能体强化学习 Multi-agent Reinforcement Learning
  - 随机博弈/马尔可夫博弈 Stochastic/Markov game
  - 分布式部分可观测马尔可夫过程 Decentralized Partially-Observable MDP

博弈模型



	Single Agent	Multiple (≥2) Agents
Large Problems	Reinforcement Learning	Multiagent Reinforcement Learning
Small Problems	Approximate Dynamic Programming	Game Theory

标准形式博弈
(矩阵博弈)

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

- 一组玩家 $i \in \mathcal{N} = \{1, 2, \cdots, n\}$
- 每个玩家的动作集 $\mathcal{A}_i \in \{a_1, a_2, \ldots\}$
- 所有玩家的联合动作 $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$
- 收益函数 $u : \mathcal{N} \times \mathcal{A} \to U \subseteq \Re$

- 每个玩家目标：
  - 最大化自身期望收益
  - 受其它玩家策略影响

$$\pi_i \in \Delta(\mathcal{A}_i),\ \text{maximize}\ \mathbb{E}_{a\sim\pi}[u_i(a)]$$

联合策略 $\pi = (\pi_1, \ldots, \pi_n)$, $\sum_{a\in\mathcal{A}} \pi_1(a_1) * \cdots \pi_n(a_n) * u_i(a)$

最佳响应
Best Response

$-i = \mathcal{N} - \{i\}$

- 假定我们是玩家 i，同时其它玩家策略已知且固定 π_{-i}

$$\pi_i \in \Delta(\mathcal{A}_i),\ \text{maximize}\ \mathbb{E}_{a\sim\pi}[u_i(a)]$$

$$\Downarrow$$

$$\pi_i \in BR(\pi_{-i}) \Leftrightarrow u_i(\pi_i, \pi_{-i}) = \max_{\pi_i'} \mathbb{E}_{a\sim(\pi_i', \pi_{-i})}[u_i(a)]$$

- π_i 称为 π_{-i} 的**最佳响应策略**

举例：石头-剪刀-布

- 2个玩家
- 3个动作
- 最佳响应：
  - 石头 → 剪刀 → 布 → 石头

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

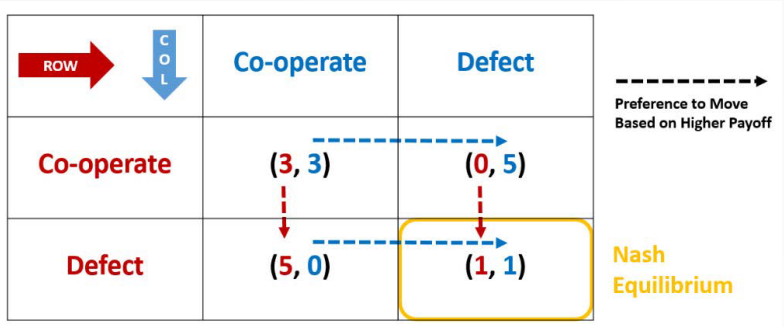
纳什均衡
Nash equilibrium

- 如果一组联合策略 π，任何一位玩家在此联合策略下单方面改变自己的策略（其他玩家策略不变）都不会提高自身的收益，称为纳什均衡
  - 每个玩家的策略是其它玩家策略的最佳响应

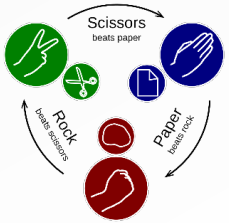
$$\forall i \in \mathcal{N}, \pi_i \in BR(\pi\_i)$$

确定性策略 vs 随机性策略

- 确定性纳什均衡策略：纯策略
  - 囚徒困境
- 随机性纳什均衡策略：混合策略
  - 石头-剪子-布 (1/3, 1/3, 1/3)



	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0



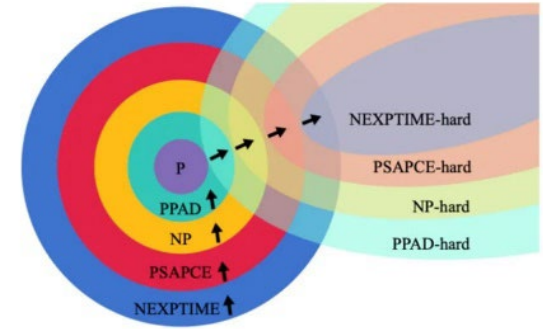
纳什均衡
Nash equilibrium

Boy \ Girl	Ballet	Football
Ballet	1, 4	0, 0
Football	0, 0	4, 1

- 在有限博弈中纳什均衡永远存在
  - 描述了多智能体博弈中玩家策略的一种均衡态(没人愿意主动离开)
  - 可能存在多个纳什均衡点，如 battle of sexes
  - 两人零和博弈中存在唯一的纳什均衡

	Boy	Girl
Equilibrium1	Ballet	Ballet
Equilibrium2	Football	Football
Equilibrium3	0.2*Ballet, 0.8*Football	0.8*Ballet, 0.2*Football

- 计算纳什均衡计算复杂度是 PPAD-Complete (⊆NP)
  - 一种思路是求解（可行的）子问题 $\pi_i \in \epsilon\text{-}BR(\pi_{-i})$
  - 另一种是计算近似纳什均衡解



近似纳什均衡

- 纳什均衡：
  - 每个玩家策略是其它玩家策略的最佳响应

$$\pi_i \in BR(\pi_{-i}) \Leftrightarrow u_i(\pi_i, \pi_{-i}) = \max_{\pi_i'} \mathbb{E}_{a \sim (\pi_i', \pi_{-i})}[u_i(a)]$$

- 近似纳什均衡：
  - 每个玩家策略是其它玩家策略的 ϵ-近似最佳响应

$$\pi_i \in \epsilon\text{-}BR(\pi_{-i}) \Leftrightarrow \quad u_i(\pi_i, \pi_{-i}) \geq \max_{\pi_i'} \mathbb{E}_{a \sim (\pi_i', \pi_{-i})}[u_i(a)] - \epsilon$$

纳什均衡
Nash equilibrium

- 在有限博弈中纳什均衡永远存在
- 计算纳什均衡计算复杂度是 PPAD-Complete (⊆NP)
  - 一种思路是求解（可行的）子问题
  - 另一种是计算近似纳什均衡解 $\pi_i \in \epsilon\text{-}BR(\pi_{-i})$
- 假定玩家都是理性（自私）的
- 假定已知博弈信息：
  - 收益/奖励函数
  - 理性假设是常识（common knowledge）

两人零和博弈

- 硬币匹配博弈：两人掷硬币 $u_1(\cdot) = -u_2(\cdot)$
  - 相同时行玩家赢1，列玩家输1
  - 不同时行玩家输1，列玩家赢1
- 纳什均衡满足
  - 行玩家最大化 u_1
  - 列玩家最大化 u_2 ⇔ 最小化 u_1
- 行玩家的纳什均衡是在列玩家最小化u_1下最大化u_1

列玩家

行玩家		A	B
$\pi(a)$	a	1, -1	-1, 1
$\pi(b)$	b	-1, 1	1, -1

动作概率

列玩家动作A下的期望收益

列玩家动作B下的期望收益

$$\max V$$
$$\pi(a) - \pi(b) \geq V \quad (\text{vs. A})$$
$$-\pi(a) + \pi(b) \geq V \quad (\text{vs. B})$$
$$\pi(a) + \pi(b) = 1$$
$$0 \leq \pi(a), \pi(b) \leq 1$$

两人零和博弈

- 多项式时间求解纳什均衡
  - 基于现成的线性规划求解器
- 找到混合（随机）纳什均衡解
  - 纯策略容易被对手针对/利用，混合策略可以有更低的可利用性 exploitability
- 硬币匹配博弈：
  - 纳什均衡策略 $\pi(a) = \pi(b) = \frac{1}{2}, V = 0$

最小最大定理

John von Neumann 1928

最大-最小：玩家1寻找 π_1 使得

$$v_1 = \max_{\pi_1} \min_{\pi_2} u_1(\pi_1, \pi_2)$$

最小-最大：玩家2寻找 π_2 使得

$$v_1 = \min_{\pi_2} \max_{\pi_1} u_1(\pi_1, \pi_2)$$

在两人零和场景，上述过程是等价的

$$v^* = \max_{\pi_1} \min_{\pi_2} u_1(\pi_1, \pi_2) = \min_{\pi_2} \max_{\pi_1} u_1(\pi_1, \pi_2)$$

---- 最小最大定理 Minimax Theorem

重复博弈
Repeated Game

- 标准形式博弈是单回合的，没有经验可言
  - 玩家无法根据多轮回合的结果自适应调整策略
- 重复博弈会进行多轮博弈，结果作为经验更新玩家策略
- 各个玩家经过多轮更新后，可能会收敛到纳什均衡解
  - 近似求解纳什均衡

马尔可夫博弈
Markov Game

- 多智能体/多玩家+马尔可夫过程 $\mathcal{MG} = (\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \gamma, \rho_0)$
  - 一组玩家 $i \in \mathcal{N} = \{1, 2, \cdots, n\}$
  - （全局）状态集 $\mathcal{S} = \{s_1, s_2, \ldots\}$
  - 每个玩家的动作集 $\mathcal{A}^i = \{a_1, a_2, \ldots\}$
  - 状态转移函数 $s_{k+1} \sim \mathcal{P}(s_k, \mathbf{a}_k)$
  - 奖励函数 $r_k^i \sim \mathcal{R}^i(s_k, \mathbf{a}_k)$

$\mathbf{a}_k = (a_k^1, \ldots, a_k^n)$

联合动作

- 也称为随机博弈
  - L. S. Shapley 1953: Stochastic Games

马尔可夫博弈
Markov Game

- 每个玩家策略 $\pi^i : \mathcal{S} \to \mathcal{A}^i$ ，n个玩家联合策略 $\boldsymbol{\pi} = (\pi^i)_{i \in \mathcal{N}}$
- 通常我们会用参数化的策略表示：$\pi(a^i|s;\theta^i)$
- 不同智能体之间策略参数是可以共享的/同构： $\theta^1 = \theta^2 = \cdots = \theta^n$
  - 同一厂家的智能驾驶汽车使用相同的策略
- 也可以非共享的/异构： $\theta^i \neq \theta^j$
  - 足球运动员踢不同的位置：前锋，后卫，守门员

马尔可夫博弈
Markov Game

- 每个玩家策略 $\pi^i : \mathcal{S} \to \mathcal{A}^i$ ，n个玩家联合策略 $\boldsymbol{\pi} = (\pi^i)_{i \in \mathcal{N}}$
- 在联合策略下每个玩家的回报(sum of rewards):

$$J^i(s_0|\boldsymbol{\pi}) = \sum_{k=0}^{\infty} r_k^i \qquad \mathbf{a}_k \sim \boldsymbol{\pi}(s_k), r_k^i \sim \mathcal{R}^i(s_k, \mathbf{a}_k)$$

  - 回报依赖于：所有的未来状态 $\{s_0, s_1, s_2, \dots\}$，以及所有智能体的所有未来动作 $\{a_0^i, a_1^i, a_2^i, \dots\}$, for all $i = 1, \dots, n$

马尔可夫博弈
Markov Game

- 每个玩家策略 $\pi^i : \mathcal{S} \to \mathcal{A}^i$ ，n个玩家联合策略 $\boldsymbol{\pi} = (\pi^i)_{i \in \mathcal{N}}$
- 价值函数 (期望回报)

$$V^i(s_0|\boldsymbol{\pi}) = \mathbb{E}\left[\sum_{k=0}^{\infty} r_k^i\right]$$

  - 期望是由所有可能的轨迹（未来状态，未来动作）得到
  - $a_t^j \sim \pi^j(\cdot|s_t),\ \forall j = 1, \ldots, n$
  - 因此 V^i 与 $\pi^1, \ldots, \pi^n$ 均有关
  - 任何一个智能体的策略改变后，其它所有人的 $V^1, \ldots, V^n$ 都要跟着改变
  - 例如：足球比赛中，其他队友不变但前锋变强，所有队友的 value 都提升，对方球员的 value 都下降

马尔可夫博弈
Markov Game

- 纳什均衡：每个玩家的策略都是其它玩家策略的最佳响应

$$\boldsymbol{\pi}_* = (\pi_*^i)_{i\in\mathcal{N}} \quad \Leftrightarrow \quad V^i(\pi_*^i, \boldsymbol{\pi}_*^{-i}) \geq V^i(\pi^i, \boldsymbol{\pi}_*^{-i}), \forall \pi^i \in \Pi^i$$

  - 任何一个玩家都不会主动改变自己的策略
  - 最佳响应

$$\pi_*^i = \arg\max_{\pi^i} V^i(\pi^i, \boldsymbol{\pi}_*^{-i})$$

NashConv

- NashConv描述了一组玩家的联合策略 $(\pi^1, \ldots, \pi^n)$，与纳什均衡之间的误差

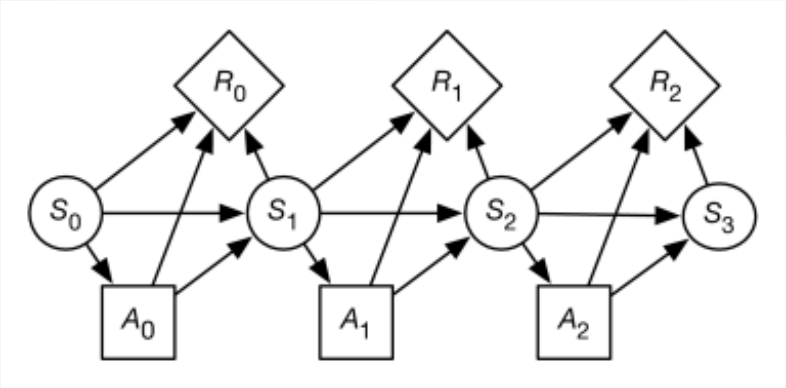
$$\begin{aligned}\text{NashConv}(\boldsymbol{\pi}) &= \sum_i \left(\max_\pi u^i(\pi, \boldsymbol{\pi}^{-i}) - u^i(\pi^i, \boldsymbol{\pi}^{-i}) \right) \\ &= \sum_i \left(\max_\pi \sum_s \rho(s) V^i(s|\pi, \boldsymbol{\pi}^{-i}) - \sum_s \rho(s) V^i(s|\pi^i, \boldsymbol{\pi}^{-i}) \right)\end{aligned}$$

  - 描述每个玩家 π^i 在面对其它玩家 π^{-i} 时与最佳响应之间的差值

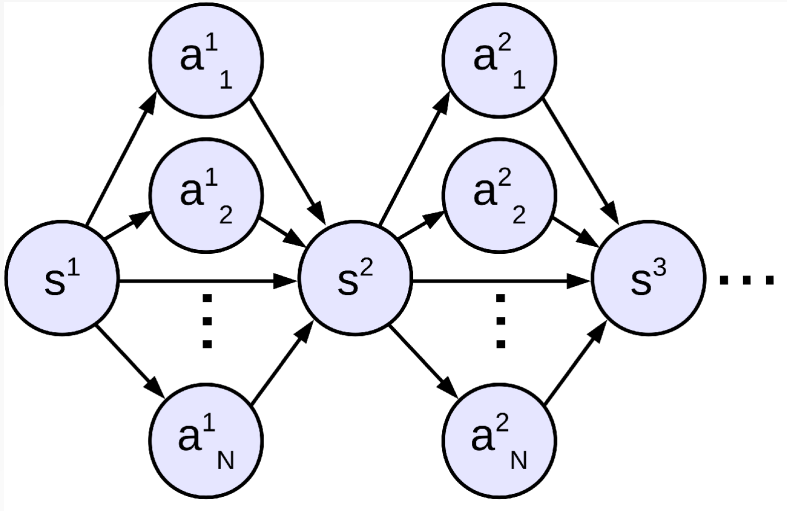
$$\text{NashConv}(\boldsymbol{\pi}) = \sum_i \left(u^i(BR^i(\pi^{-i}), \boldsymbol{\pi}^{-i}) - u^i(\pi^i, \boldsymbol{\pi}^{-i}) \right)$$

- 纳什均衡的$NashConv(\pi_*^1, \ldots, \pi_*^n) = 0$
  - 纳什均衡点上每个玩家策略都是其它玩家策略的最佳响应

MDP vs MG

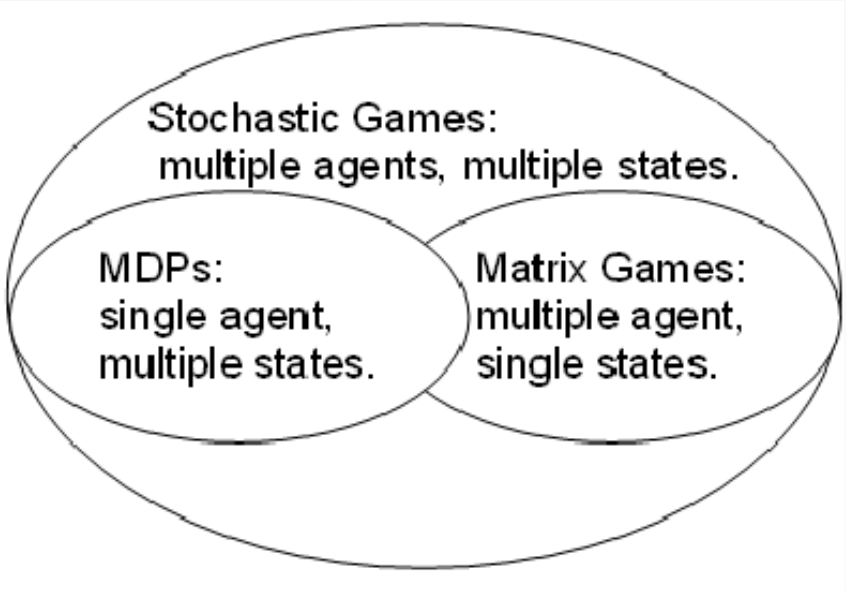


马尔可夫决策过程



马尔可夫博弈

MDP vs NFG vs MG



动作-价值函数

- (状态)价值函数

$$V^i(s_0|\boldsymbol{\pi}) = \mathbb{E}\left[\sum_{k=0}^{\infty} r_k^i\right]$$

- 动作-价值函数/Q函数

$$Q^i(s, a_1, \ldots, a_n) = \mathbb{E}\left[\mathcal{R}^i(s, a_1, \ldots, a_n) + \gamma \sum_{s'} \mathcal{P}(s'|s, a_1, \ldots, a_n) V^i(s'|\pi)\right]$$

- 纳什Q函数 （纳什策略下的价值函数）

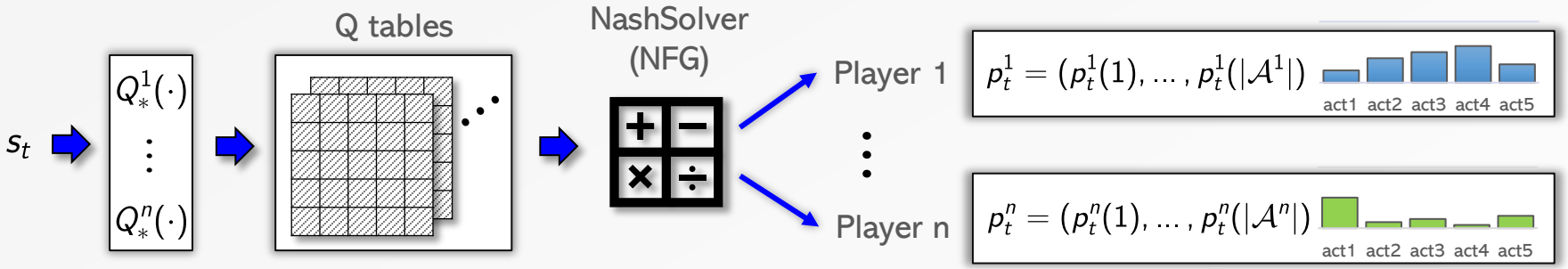
$$Q_*^i(s, a_1, \ldots, a_n) = \mathbb{E}\left[\mathcal{R}^i(s, a_1, \ldots, a_n) + \gamma \sum_{s'} \mathcal{P}(s'|s, a_1, \ldots, a_n) V_*^i(s'|\pi)\right]$$

动作-价值函数

- 在纳什Q函数 $(Q_*^i)_{i\in\mathcal{N}}$ 已知的前提下，n个玩家在每一时刻的决策变成了一个标准形式博弈问题
  - 一组玩家 $i \in \mathcal{N} = \{1, 2, \cdots, n\}$
  - 每个玩家的动作集 $\mathcal{A}_i \in \{a_1, a_2, \dots\}$
  - 所有玩家的联合动作 $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$
  - 收益函数 $u^i(\dots) = Q_*^i(s, \dots)$

动作-价值函数

- 以当前状态下的纳什Q表作为玩家在标准形式博弈的收益矩阵，计算纳什均衡动作概率分布



$$(\pi_*^1(s), \ldots, \pi_*^n(s)) = NashSolver(Q_*^1(s, \cdot), \ldots, Q_*^n(s, \cdot))$$

两人零和MG

- 两个玩家 $\mathcal{N} = \{1, 2\}$
- 两人的目标完全相反
  - 奖励 $\mathcal{R}^1(s, \mathbf{a}) = -\mathcal{R}^2(s, \mathbf{a})$
  - 回报 $J^1(s|\pi^1, \pi^2) = -J^2(s|\pi^1, \pi^2)$
  - 价值函数 $V^1(s|\pi^1, \pi^2) = -V^2(s|\pi^1, \pi^2)$
  - Q函数 $Q^1(s, a^1, a^2|\pi^1, \pi^2) = -Q^2(s, a^1, a^2|\pi^1, \pi^2)$
- 纳什均衡一定存在 (π_*^1, π_*^2)，且满足最小最大条件

$$V^1(\pi_*^1, \pi^2) \geq V^1(\pi_*^1, \pi_*^2) \geq V^1(\pi^1, \pi_*^2)$$

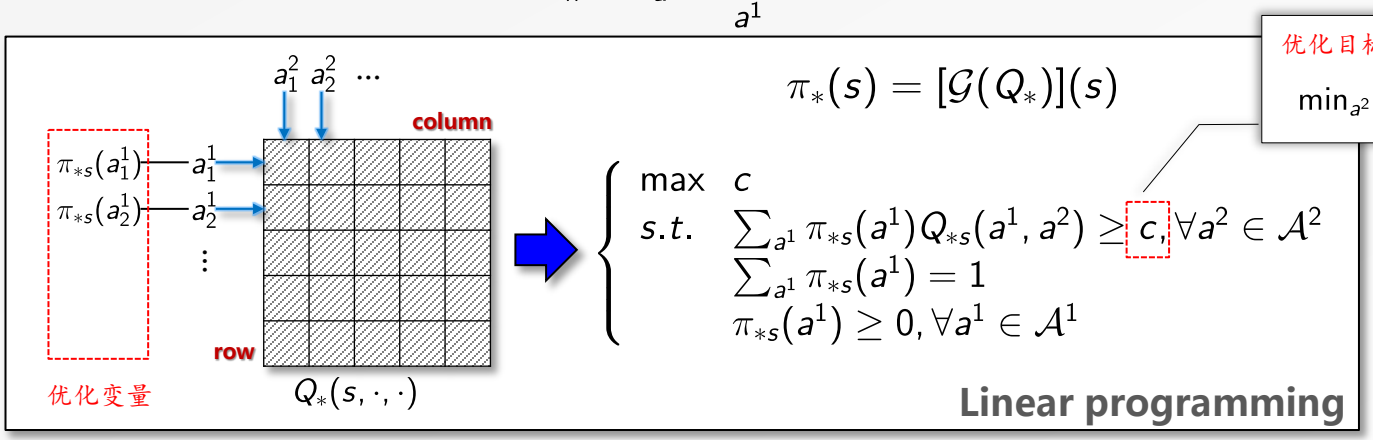
According to Sharpley theory [Shapley (1953) PNAS], there always exists a **unique Nash value** to two-player zero-sum game

$$V_*(s) = \max_{\pi^1 \in \Pi^1} \min_{\pi^2 \in \Pi^2} V^1(s|\pi^1, \pi^2)$$

两人零和MG

- 两人零和马尔可夫博弈的纳什均衡策略

$$\pi_*^1(s) = \arg\max_{\pi^1} \min_{a^2} \sum_{a^1} \pi^1(a^1|s) Q_*(s, a^1, a^2)$$



- 已知纳什 Q_* 函数，在状态 s下的 $Q_*(s,\cdot,\cdot)$ 组成两人零和标准形式博弈
- 线性规划在多项式时间求解两人零和的纳什均衡动作概率分布

贝尔曼最小最大方程

Bellman minimax eq

- 价值函数贝尔曼最小最大方程

$$V_*(s) = \max_{\pi^1 \in \Pi^1} \min_{\pi^2 \in \Pi^2} V^1(s|\pi^1, \pi^2)$$

$$V_*(s) = \max_{\pi^1(s) \in \Pi^1} \min_{a^2 \in \mathcal{A}^2} \sum_{a^1 \in \mathcal{A}^1} \pi^1(a^1|s) \left[\mathcal{R}(s, a^1, a^2) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a^1, a^2) V_*(s') \right]$$

(Bellman principle)

- Q函数的贝尔曼最小最大方程

$$Q_*(s_t, a_t^1, a_t^2) = \mathcal{R}(s_t, a_t^1, a_t^2) + \gamma \sum_{s_{t+1}} \mathcal{P}(s_{t+1}|s_t, a_t^1, a_t^2) \max_{\pi^1} \min_{a_t^2} \sum_{a_{t+1}^1} \pi^1(a_{t+1}^1|s_{t+1}) Q^*(s_{t+1}, a_{t+1}^1, a_{t+1}^2)$$

- 挑战：max min 非线性算子 （动态规划求解！）

基于价值函数的强化学习

价值迭代

贝尔曼最小最大方程

$$Q_*(s_t, a_t^1, a_t^2) = \mathcal{R}(s_t, a_t^1, a_t^2) + \gamma \sum_{s_{t+1}} \mathcal{P}(s_{t+1}|s_t, a_t^1, a_t^2) \max_{\pi^1} \min_{a_t^2} \sum_{a_{t+1}^1} \pi^1(a_{t+1}^1|s_{t+1}) Q^*(s_{t+1}, a_{t+1}^1, a_{t+1}^2)$$

$$\begin{aligned} Q_{k+1}(s_t, a_t^1, a_t^2) &= \mathcal{T}(Q_k)(s_t, a_t^1, a_t^2) \\ &= \mathcal{R}(s_t, a_t^1, a_t^2) + \gamma \sum_{s_{t+1}} \mathcal{P}(s_{t+1}|s_t, a_t^1, a_t^2) \max_{\pi^1} \min_{a_{t+1}^2} \sum_{a_{t+1}^1} \pi^1(a_{t+1}^1|s_{t+1}) Q_k(s_{t+1}, a_{t+1}^1, a_{t+1}^2) \end{aligned}$$

$k \leftarrow k+1$

- 价值迭代
  - 给定一个初始 $Q_0(=0)$
  - 将当前 Q_k 代入价值迭代算子 $\mathcal{T}()$ 得到新的 Q_{k+1}
  - 价值迭代算子 $\mathcal{T}()$ 满足 γ-收缩性，所以 $\lim_{k\to\infty} Q_k = Q_*$
  - 对比：2p0sMG每一迭代求解 Max Min (线性规划)，1pMDP求解Max 或 Min (枚举)

策略迭代

- 策略评估：给定玩家1的策略 π_k，计算在最坏对手策略下的价值

$$Q_k(s_t, a_t^1, a_t^2) = \mathcal{R}(s_t, a_t^1, a_t^2) + \gamma \sum_{s_{t+1}} \mathcal{P}(s_{t+1}|s_t, a_t^1, a_t^2) \min_{a_{t+1}^2} \sum_{a_{t+1}^1} \pi_k^1(a_{t+1}^1|s_{t+1}) Q_k(s_{t+1}, a_{t+1}^1, a_{t+1}^2)$$

  - 看作是玩家2在已知玩家1策略π_k下，求最优化(min)的MDP过程

$$Q_*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s'} \mathcal{P}(s'|s, a) \min_{a'} Q_*(s', a')$$

  - 玩家2使用动态规划(VI/PI)计算最坏对手策略，实现对玩家1的策略评估

- 策略提升：

$$\pi_{k+1}^1(s) = \arg\max_{\pi^1} \min_{a^2} \sum_{a^1} \pi^1(a^1|s) Q_k(s, a^1, a^2)$$

策略迭代

- 策略迭代得到Q函数序列 $\{Q_1, Q_2, \dots\}$是单调递增，且有上限纳什Q_*函数，所以收敛且收敛到纳什均衡解
- 对比
  - 2p0sMG：策略评估涉及玩家2 (非线性)贝尔曼最优方程的求解
    策略提升需要求解 Max Min (线性规划)
  - 1pMDP：策略评估涉及智能体(线性)贝尔曼期望方程的求解
    策略提升需要求解 Max 或 Min (枚举)

在线学习

- 价值迭代依赖模型

$$Q_{k+1}(s_t, a_t^1, a_t^2) = \mathcal{R}(s_t, a_t^1, a_t^2) + \gamma \sum_{s_{t+1}} \mathcal{P}(s_{t+1}|s_t, a_t^1, a_t^2) \max_{\pi^1} \min_{a_{t+1}^2} \sum_{a_{t+1}^1} \pi^1(a_{t+1}^1|s_{t+1}) Q_k(s_{t+1}, a_{t+1}^1, a_{t+1}^2)$$

- 回顾：
  - 1pMDP 价值迭代 (基于模型)

$$Q_{k+1}(s, a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \max_{a'} Q_k(s', a')$$

  - Q学习 （无模型）

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$$

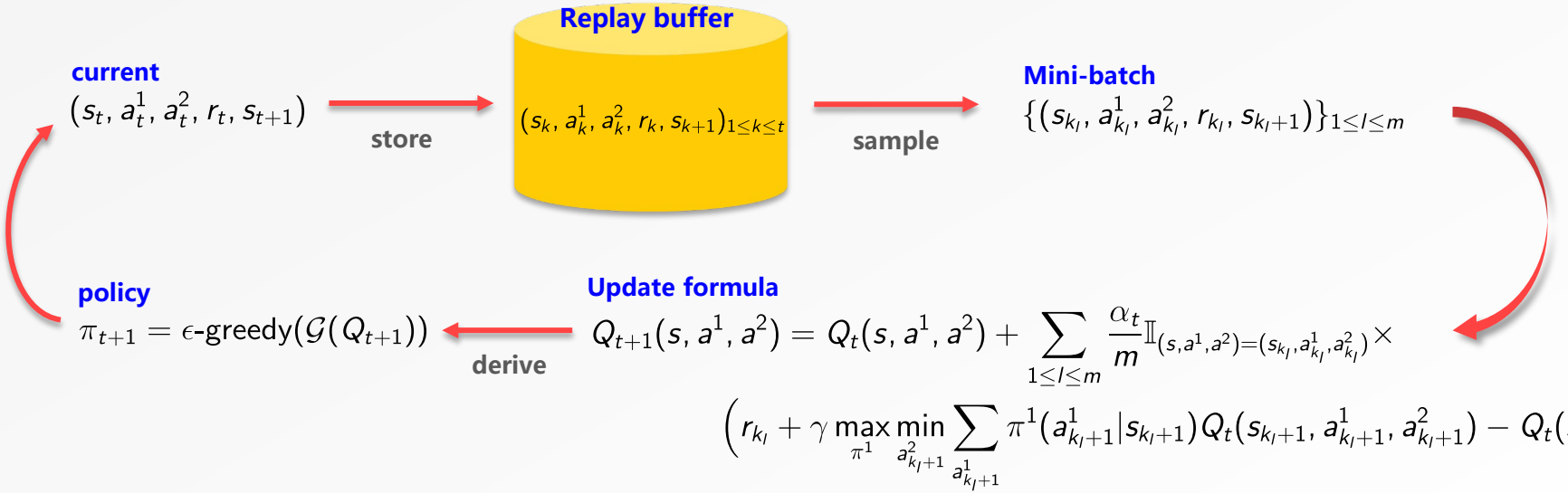
Minimax-Q [Littman (1994) MLP]

$$Q_{t+1}(s_t, a_t^1, a_t^2) = Q_t(s_t, a_t^1, a_t^2) + \alpha_t \left[r_t + \gamma \max_{\pi^1} \min_{a_{t+1}^2} \sum_{a_{t+1}^1} \pi^1(a_{t+1}^1 | s_{t+1}) Q_t(s_{t+1}, a_{t+1}^1, a_{t+1}^2) - Q_t(s_t, a_t^1, a_t^2) \right]$$

- 每一时刻根据在线观测 $(s_t, a_t^1, a_t^2, r_t, s_{t+1})$，更新Q表
- 针对有限2p0sMG，在无限探索、无穷时刻收敛为贪心策略(GLIE)条件下，Minimax-Q算法收敛到纳什Q_*

经验回放

- Q学习/Minimax-Q 算法每一时刻的观测只使用一次，数据利用率不高
- 经验回放：将在线观测数据存储在经验池，每次更新时从经验池选择多个样本更新



经验回放

- Q学习/Minimax-Q 算法每一时刻的观测只使用一次，数据利用率不高
- 经验回放：将在线观测数据存储在经验池，每次更新时从经验池选择多个样本更新
  - 解耦相邻时刻在线观测的相关性
  - Minibatch更新比stochastic更新收敛性更好

足球博弈
Soccer Game

- 状态集：A position (4*5) * B position (4*5) * Ball possession (Boolean)
  - 4*5 网格
  - A player，B player，Ball
  - Ball：跟随A或B
- A/B动作集: 上、下、左、右、停止
- 状态转移：
  - 每一时刻，随机概率区分谁先执行动作，谁后执行动作
  - 如果玩家x执行动作后会进入另一玩家y所在位置，球权归另一玩家y，并且玩家x动作不执行
- 奖励：
  - 左右两侧分别是A/B的球门
  - 当Ball进入x方球门，玩家x获 -1，玩家y获+1

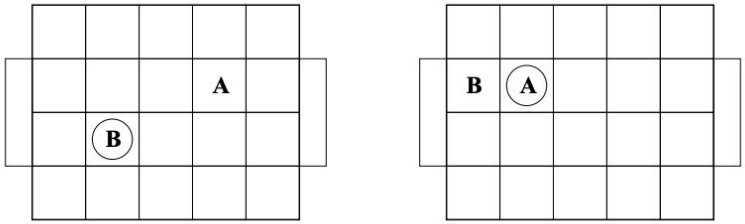
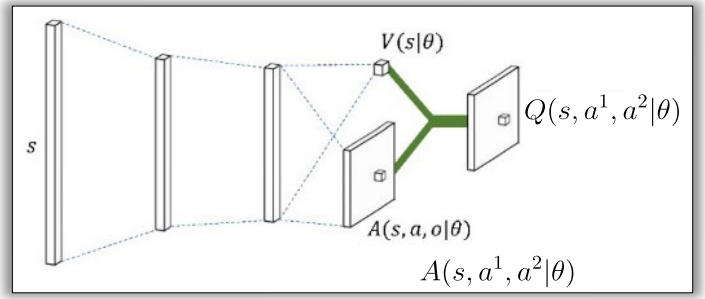


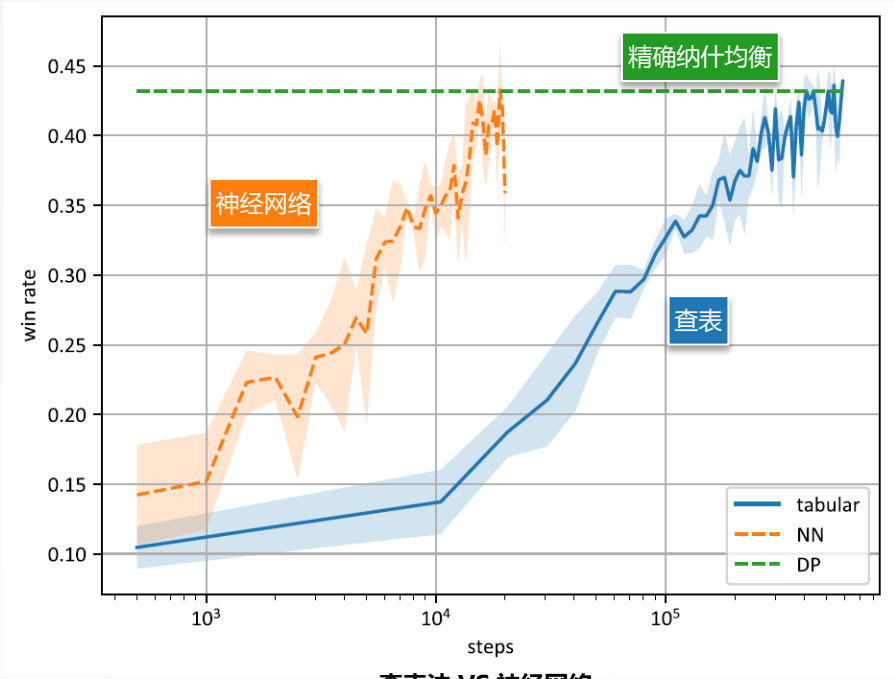
Figure 2: An initial board (left) and a situation requiring a probabilistic choice for A (right).

足球博弈
Soccer Game

- 基于神经网络的Q函数



[Mnih et al. (2015) Nature]



查表法 VS 神经网络

n玩家马尔可夫博弈

- 对n玩家马尔可夫博弈，构造（每个玩家的）纳什Q函数

$$Q_*^i(s,a^1,\ldots,a^n) = r^i(s,a^1,\ldots,a^n) + \beta \sum_{s' \in S} p(s'|s,a^1,\ldots,a^n) v^i(s',\pi_*^1,\ldots,\pi_*^n)$$

[Hu & Wellman (2003)]

  - $\boldsymbol{\beta}$ 衰减因子
  - $(\boldsymbol{\pi}_*^1,\ldots,\boldsymbol{\pi}_*^n)$ 纳什均衡策略

- 与Q学习类似，设计在线学习算法，学习n玩家的纳什Q函数

Nash-Q算法

[Hu & Wellman (2003)]

Initialize:

Let $t = 0$, get the initial state s_0.

Let the learning agent be indexed by i.

For all $s \in S$ and $a^j \in A^j$, $j = 1, \ldots, n$, let $Q_t^j(s, a^1, \ldots, a^n) = 0$.

Loop

Choose action a_t^i.

Observe $r_t^1, \ldots, r_t^n; a_t^1, \ldots, a_t^n$, and $s_{t+1} = s'$

Update Q_t^j for $j = 1, \ldots, n$

$Q_{t+1}^j(s, a^1, \ldots, a^n) = (1 - \alpha_t) Q_t^j(s, a^1, \ldots, a^n) + \alpha_t [r_t^j + \beta NashQ_t^j(s')]$

where $\alpha_t \in (0, 1)$ is the learning rate, and $NashQ_t^k(s')$ is defined in (7)

Let $t := t + 1$.

- $NashQ_t^i(s') = \pi^1(s') \cdots \pi^n(s') \cdot Q_t^i(s')$ ：根据 $(Q_t^1(s'), \ldots, Q_t^n(s'))$ 构造标准形式博弈，经过计算得到的纳什均衡策略，以及相应的期望收益

小结

- 基于价值函数的博弈强化学习
  - 学习纳什均衡点的价值函数，价值函数推出纳什均衡策略
  - 尤其两人零和问题，纳什均衡解一定存在，策略可由LP在多项式时间求解
  - 基于价值的特点很容易和 value-based RL （动态规划、Q学习）技术结合

小结

- 基于价值函数的博弈强化学习
  - 学习纳什均衡点的价值函数，价值函数推出纳什均衡策略
  - 尤其两人零和问题，纳什均衡解一定存在，策略可由LP在多项式时间求解
  - 基于价值的特点很容易和 value-based RL (动态规划、Q学习) 技术结合

  - 多人 (n>2) 博弈问题很难由价值函数求解纳什均衡策略 (Nash solver $\pi(s) = [\mathcal{G}(Q)](s)$)
  - Q函数包含所有玩家的动作 $Q(s, a^1, a^2, \dots, a^n)$，复杂度随动作数量和玩家数量的增加指数型增长（维数灾）

基于策略优化的强化学习

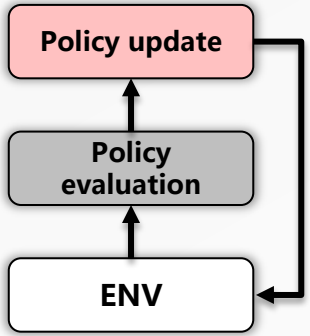
策略强化学习

policy-based RL

- 单智能体RL：明确定义策略函数，使用策略更新公式训练

Policy update formula

$$\begin{cases} \mathbb{E}[Q(s,a)\nabla \log \pi(s,a)] & \text{A3C/TRPO/PPO} \\ \mathbb{E}[\nabla_a Q(s,a)\nabla \pi(a|s)] & \text{DDPG} \end{cases}$$

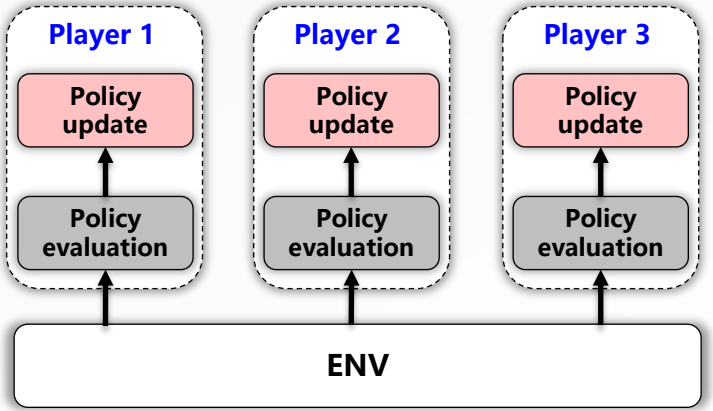


- 多智能体/多玩家：independent learning + policy RL

Player 1 update $\Delta\pi^1 \leftarrow \mathbb{E}_{s,a^1\sim\mathcal{MG}(\pi^{-1})}[Q(s,a^1)\nabla \log \pi^1(s,a^1)]$

Player 2 update $\Delta\pi^2 \leftarrow \mathbb{E}_{s,a^2\sim\mathcal{MG}(\pi^{-2})}[Q(s,a^2)\nabla \log \pi^2(s,a^2)]$

$\vdots$

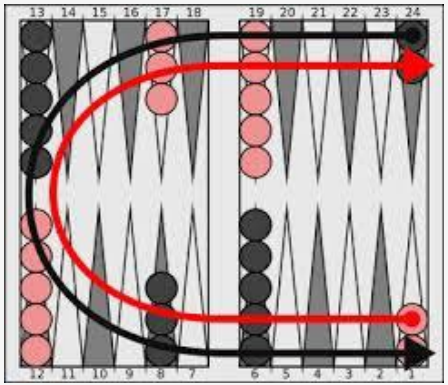


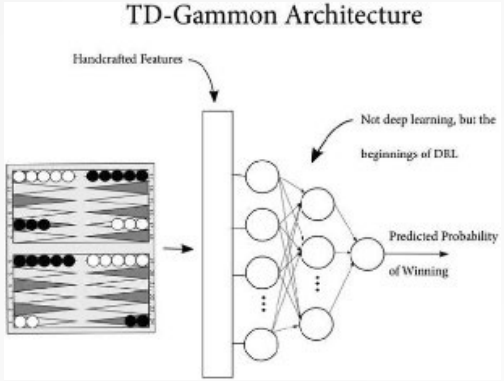
- 独立学习下由于其它玩家的策略变化，导致环境是动态变化，policy update不稳定，所有玩家容易出现策略震荡

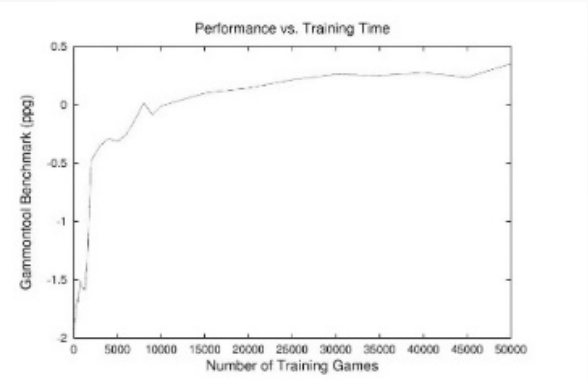
自我博弈
Self Play

- 和自己的策略博弈，在博弈过程中学习如何提升，适用于两人零和对称symmetric博弈

西洋双陆棋 [Gerald Tesauro, TD-Gammon, 1992]





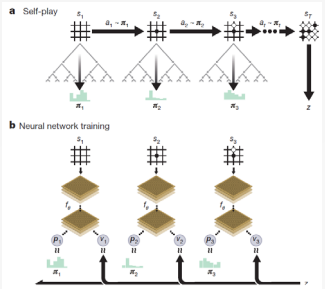


TD-Gammon: neural network that can learn to play backgammon solely by playing against itself and learning from the results.

自我博弈
Self Play

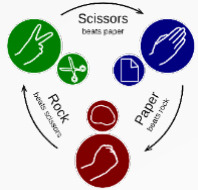
- 和自己的策略博弈，在博弈过程中学习如何提升，适用于两人零和对称symmetric博弈

AlphaGo Zero [David Silver, et al, Nature, 2017]



"The program plays a game s1, ..., sT against itself"

自我博弈容易出现"策略转圈 strategic circle"，导致训练过程振荡不收敛



石头 ← 布 ← 剪刀 ← 石头 。。。

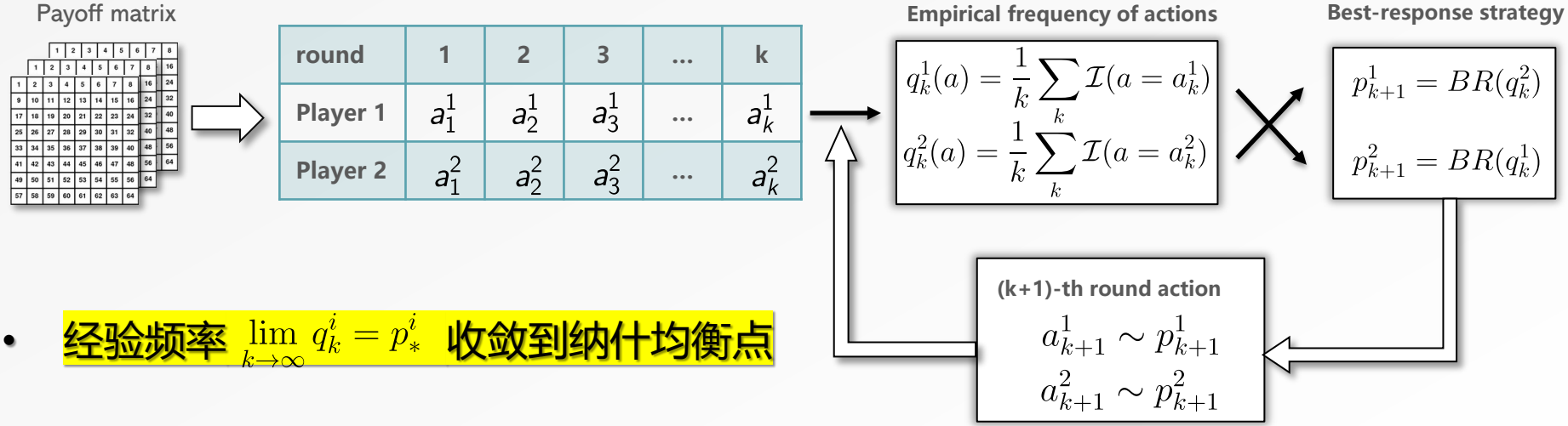
AlphaGo Zero解决办法：保存历史最强的策略作为对手，只有当前策略提升到以55%胜率打败最强对手时，才替换成最强策略

If the new player wins by a margin of > 55% then it becomes the best player, and is subsequently used for self-play generation, and also becomes the baseline for subsequent comparisons.

虚拟博弈

Fictitious Play [Brown (1951) AAPA]

- 标准形式+重复博弈
  - 玩家从任意策略出发
  - 每一轮玩家会根据其它玩家在历史回合中选择策略的经验频率 (empirical frequency)，选择对应的最佳响应 (best response)并在下一轮执行

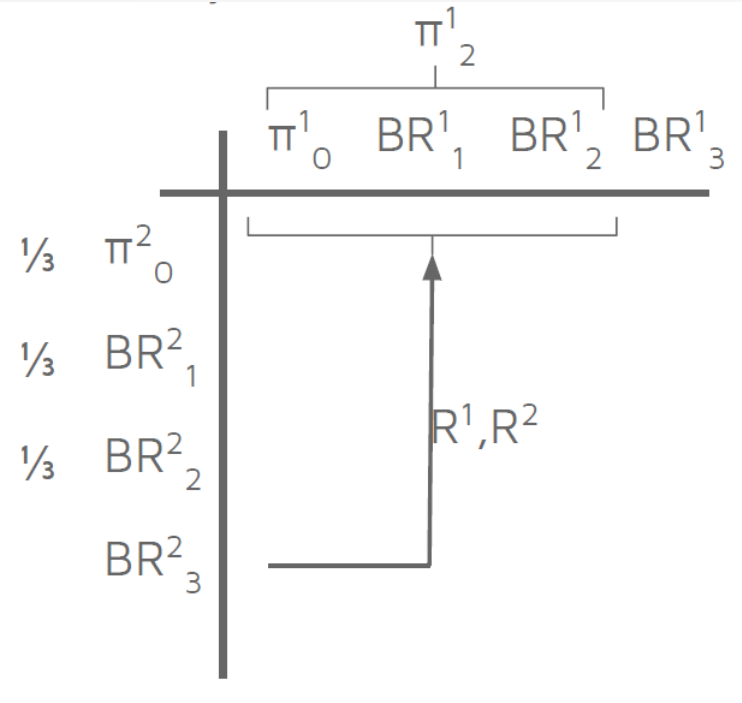


- 经验频率 $\lim_{k\to\infty} q_k^i = p_*^i$ 收敛到纳什均衡点

虚拟博弈
Fictitious Play

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

- 标准形式+重复博弈 （RPS博弈）



– 从玩家任意策略出发 $(\boldsymbol{\pi_0^1}, \boldsymbol{\pi_0^2})$

– 每一轮针对对手在历史回合的平均策略，计算最优响应并在下一轮执行

虚拟博弈
Fictitious Play

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

- 标准形式+重复博弈 （RPS博弈）

	R
R	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)

虚拟博弈
Fictitious Play

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

- 标准形式+重复博弈 （RPS博弈）

	R	P
R	0	1
P	-1	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - (½, ½, 0), (½, ½, 0)

虚拟博弈
Fictitious Play

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

- 标准形式+重复博弈 （RPS博弈）

	R	P	P
R	0	1	1
P	-1	0	0
P	-1	0	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - (½, ½, 0), (½, ½, 0)
- Iteration 2:
  - BR^1_2, BR^2_2 = P, P
  - (⅓, ⅔, 0), (⅓, ⅔, 0)

虚拟博弈
Fictitious Play

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

- 标准形式+重复博弈 （RPS博弈）

	R	P	P	S
R	0	1	1	-1
P	-1	0	0	1
P	-1	0	0	1
S	1	-1	-1	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - (½, ½, 0), (½, ½, 0)
- Iteration 2:
  - BR^1_2, BR^2_2 = P, P
  - (⅓, ⅔, 0), (⅓, ⅔, 0)
- Iteration 3:
  - BR^1_3, BR^2_3 = S, S
  - (¼,½,¼), (¼,½,¼)

虚拟博弈
Fictitious Play

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

- 标准形式+重复博弈 （RPS博弈）

	R	P	P	S	S
R	0	1	1	-1	-1
P	-1	0	0	1	1
P	-1	0	0	1	1
S	1	-1	-1	0	0
S	1	-1	-1	0	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - (½, ½, 0), (½, ½, 0)
- Iteration 2:
  - BR^1_2, BR^2_2 = P, P
  - (⅓, ⅔, 0), (⅓, ⅔, 0)
- Iteration 3:
  - BR^1_3, BR^2_3 = S, S
  - (¼,½,¼), (¼,½,¼)

虚拟博弈
Fictitious Play

- 马尔可夫博弈的虚拟博弈过程
  - 保存关于其它玩家的虚拟策略 ：统计玩家的历史经验行为（平均策略）

$$\tilde{\pi}_k^j(s,a) = \frac{1}{k}\left(\pi_1^j(s,a) + \ldots \pi_k^j(s,a)\right)$$

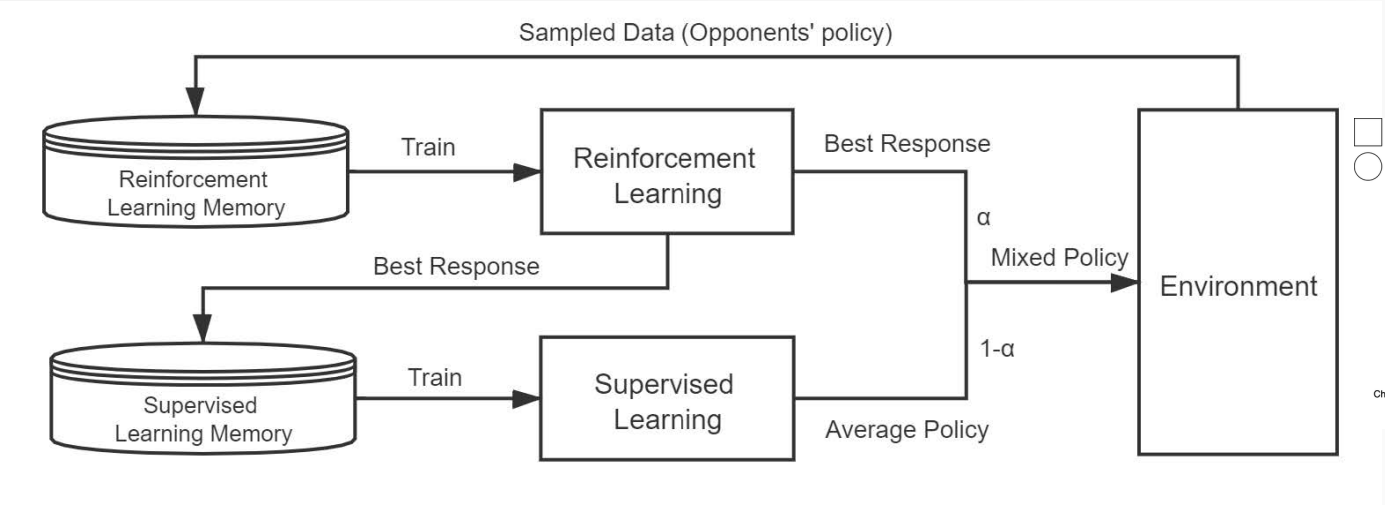
    - 根据玩家历史行为(s_t, a_t)监督训练一个平均策略网络（average policy net）
  - 下一轮博弈的策略采用针对其它玩家经验行为的最佳响应

$$\pi_{k+1}^i = BR(\tilde{\pi}_k^{-i})$$

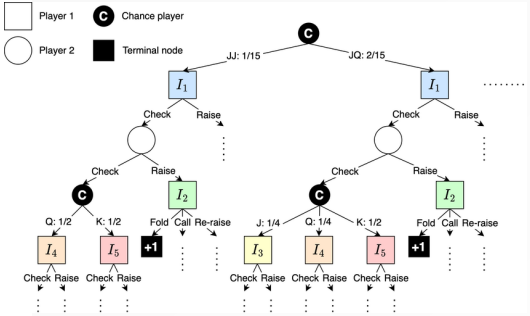
    - 强化学习最佳响应 $\max \sum \gamma^t r_t^i$， against $\tilde{\pi}_k^{-i}$

神经网络虚拟自我博弈

Neural Fictitious Self Play [Heinrich & Silver (2016)]



Leduc Hold'em

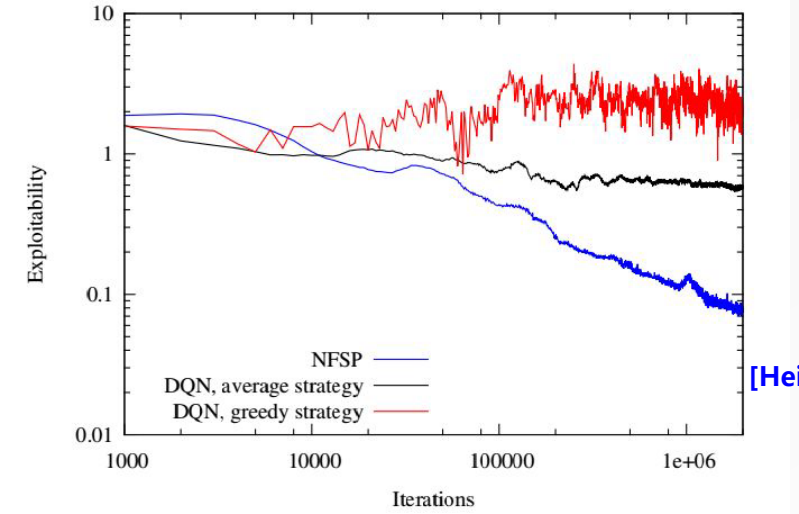


- 每个玩家以一定概率执行best response或average policy
- 所有数据都进入RL Memory用于DQN训练BR
- BR策略执行的动作进入SL Memory训练average policy

神经网络虚拟自我博弈

Neural Fictitious Self Play

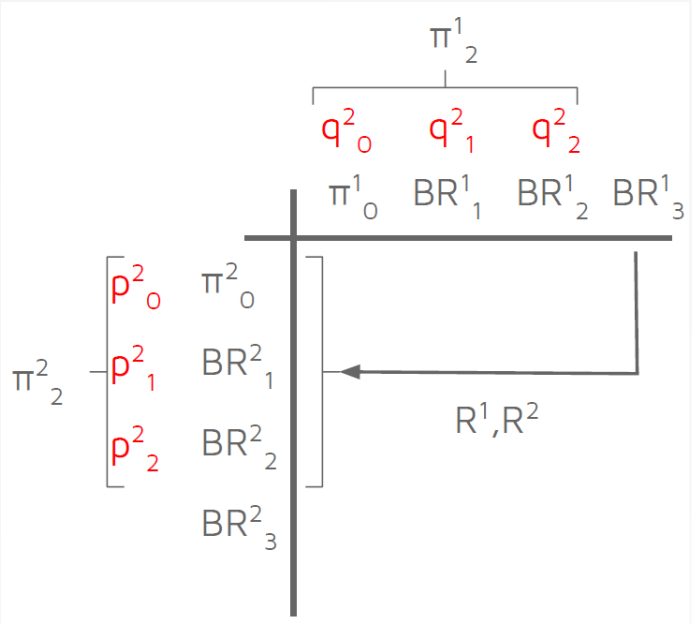
- Leduc Hold’ em poker experiments:



[Heinrich & Silver (2016)]

- 1st scalable end-to-end approach to learn approximate Nash equilibria w/o prior domain knowledge
  - Competitive with superhuman computer poker programs when it was released

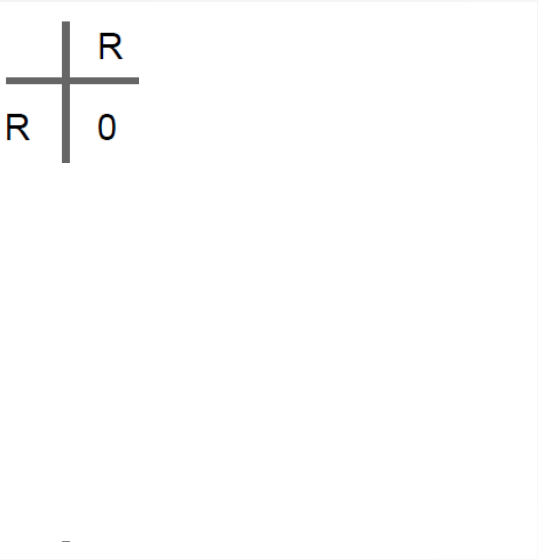
Double Oracle [HB McMahan 2003]



- 标准形式的重复博弈
- 玩家从任意策略出发(π_0^1, π_0^2)
  - 统计各玩家已有策略相互之间的 payoff，形成 (empirical) payoff matrix
  - 根据(empirical) payoff matrix 计算已有策略组合的纳什均衡概率 (p^n, q^n)
  - 针对组合策略 (p^n, q^n) 计算各玩家的最佳响应 (BR_n^1, BR_n^2)

Double Oracle

- 收敛次数更快



- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)

Double Oracle

- 收敛次数更快

	R	P
R	0	1
P	-1	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - Solve the game : (0, 1, 0), (0, 1, 0)

Double Oracle

- 收敛次数更快

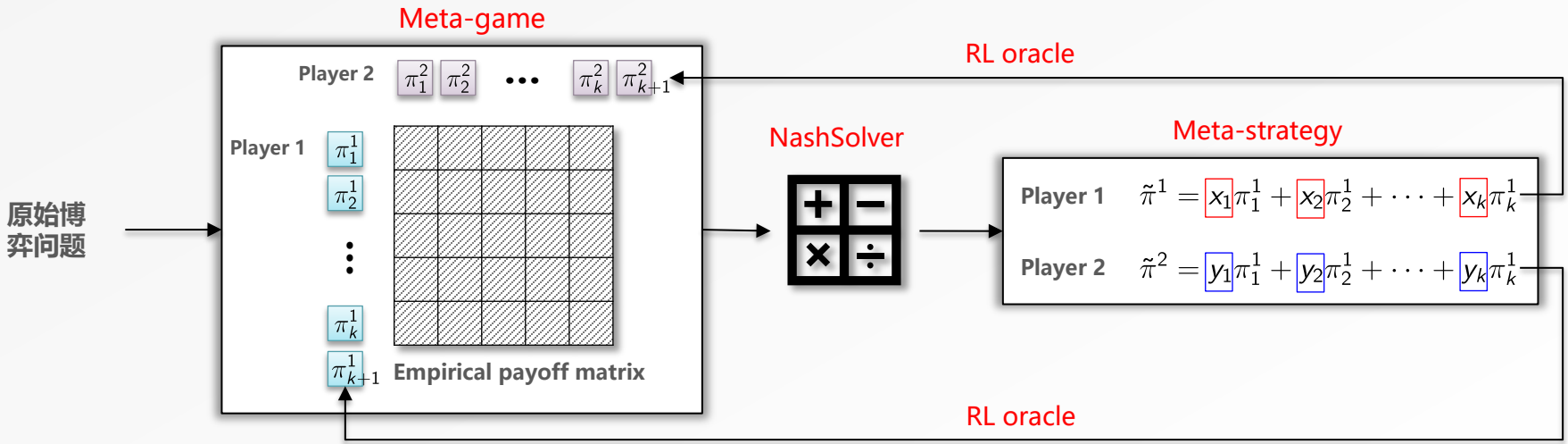
	R	P	S
R	0	1	-1
P	-1	0	1
S	1	-1	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - Solve the game : (0, 1, 0), (0, 1, 0)
- Iteration 2:
  - BR^1_2, BR^2_2 = S, S
  - (⅓, ⅓, ⅓), (⅓, ⅓, ⅓) 收敛!

PSRO

policy space response oracle [Lanctot et al. (2017) NIPS]

- **标准形式/马尔可夫博弈**
- 根据当前玩家已有策略统计 empirical payoff matrix （meta-game）
- 计算已有策略组合构成纳什均衡的概率
- 针对纳什策略组合（meta-strategy）分别计算各自的最佳响应
  - 马尔可夫问题需要RL oracle计算 best response/approximate best response



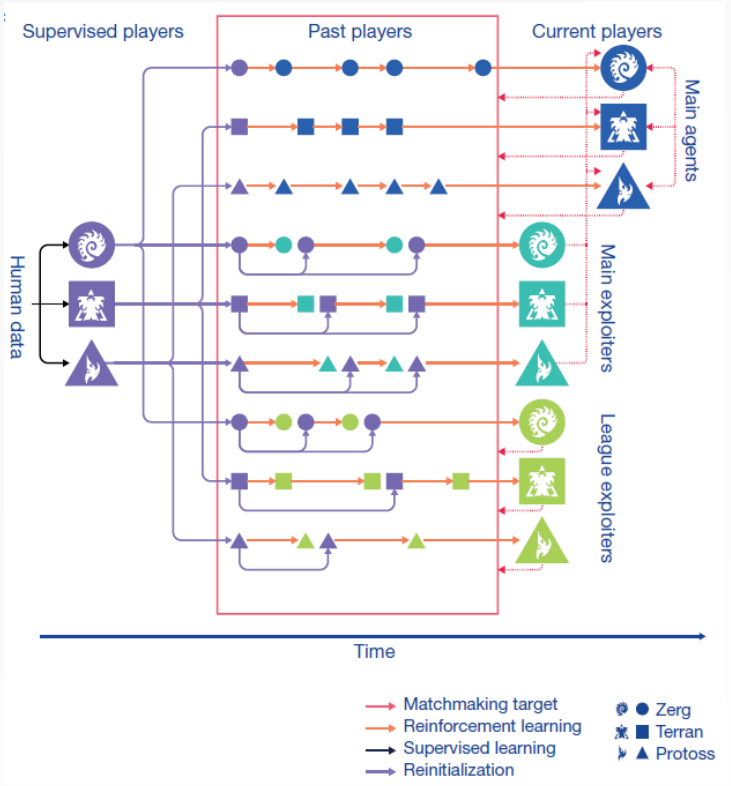
PSRO

policy space response oracle

- 如果 $x_1 = x_2 = \cdots = x_k = \frac{1}{k}$ ， PSRO变成 fictitious play
- 如果 $x_1 = \cdots = x_{k-1} = 0, x_k = 1$ ， PSRO变成 iterated best response

AlphaStar

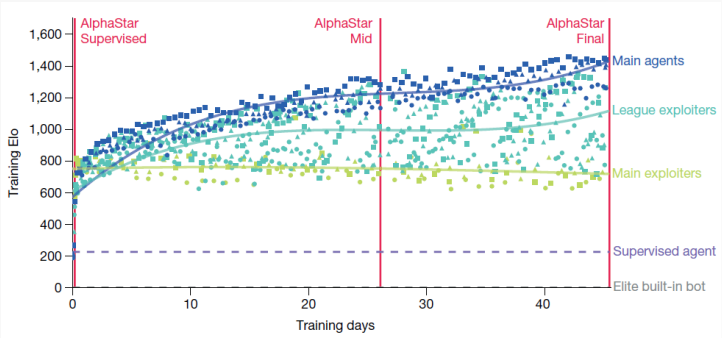
- PSRO的思路在于构建策略池 policy pool，从中选取适合于作对手的策略
- 将新学习的策略加入到策略池，不断补充和完善策略集合
- AlphaStar policy pool (league) 包含三种 past agents:
  - **Main agents** 主策略
  - **Main exploiters** 发现当前Main agents的缺陷
  - **League exploiters** 发现 league 中的缺陷



[Vinyals et al Nature AlphaStar (2019)]

AlphaStar

- 训练模式：
  - 当前的Main agents 与整个 league 以及自己对抗训练
  - 当前的Main exploiters 与当前的 Main agents 对抗训练
  - 当前的League exploiters 与整个 league 对抗训练



- 策略池对手选择概率根据打败难易程度决定 (Prioritized Fictitious Self Play)

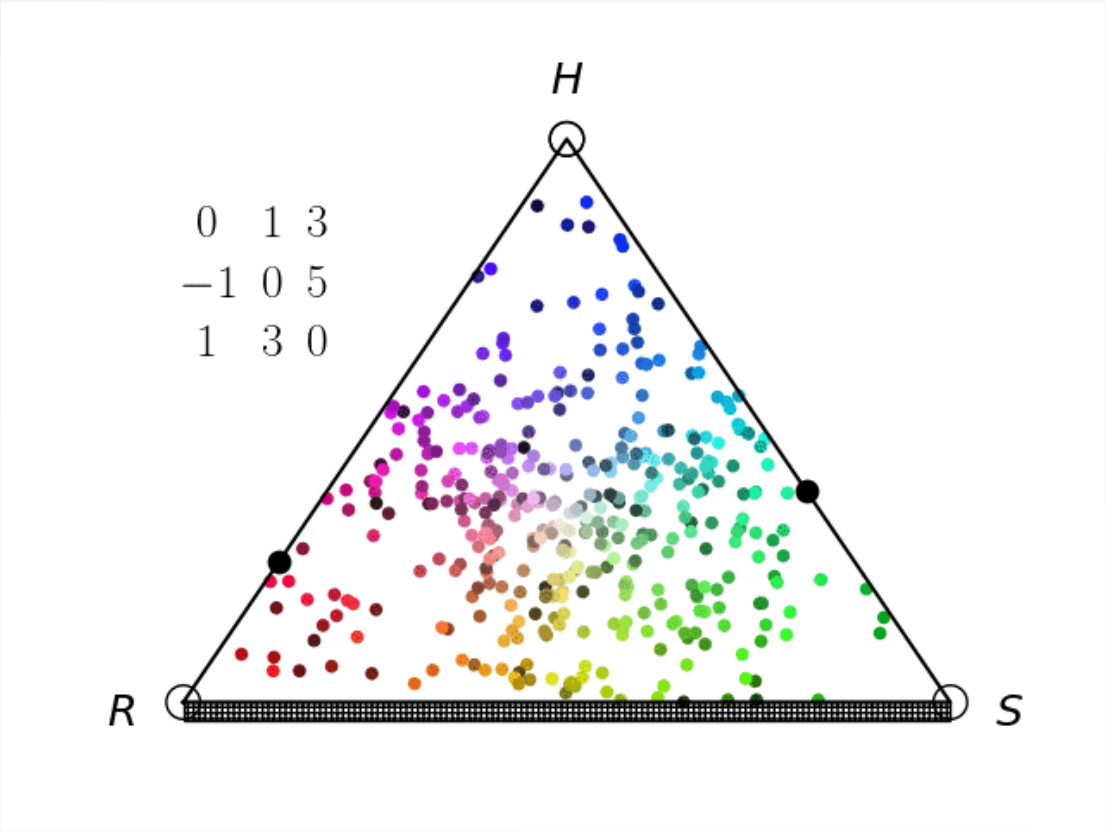
$$\frac{f(\mathbb{P}[A \text{ beats } B])}{\sum_{C \in \mathcal{C}} f(\mathbb{P}[A \text{ beats } C])}$$

  - $f_{\text{hard}}(x) = (1-x)^p$ focuses on hardest players (default weighting)
  - $f_{\text{var}}(x) = x(1-x)$ preferentially plays against opponents around its own level (main exploiters and struggling main agents)

演化博弈理论

Evolutionary Game Theory

- 由一群玩家（个体）组成的种群 (population) 在策略层面的交互过程
- 每个玩家分配一种策略，策略来源可以是
  - 1）由上一代父辈玩家遗传
  - 2）模仿其它玩家策略
- 每一代个体随机与种群其它个体配对，进行两人博弈
- 博弈结果决定个体策略的遗传概率
  - 收益越高越会遗传给下一代
  - 收益越低越会模仿其它玩家策略


H
R
S
0 1 3
−1 0 5
1 3 0

演化博弈理论
Evolutionary Game Theory

- 演化过程中同一策略的个体数量占种群的比例满足

Growth rate of a

$\sum \pi_t(a)u(a,\pi_t)$

$$\dot{\pi}_t(a) = \pi_t(a)\left[u(a,\boldsymbol{\pi}_t) - \bar{u}(\boldsymbol{\pi}_t)\right]$$

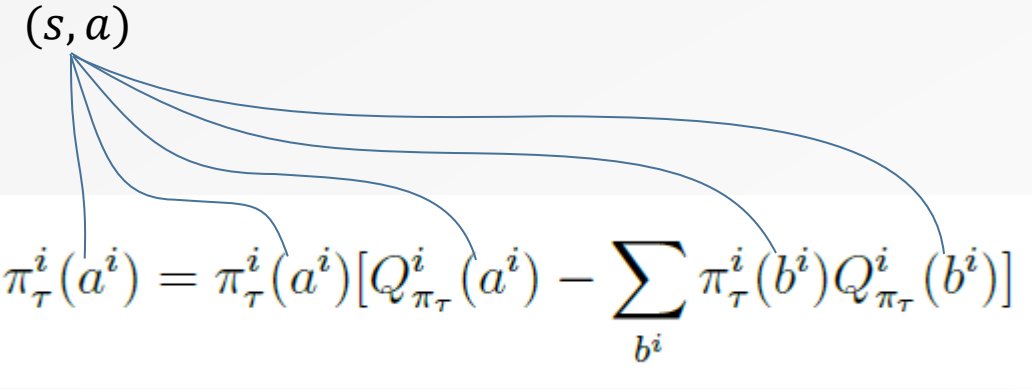
Current frequency of strategy
（多少a的个体可以进入下一代）

Relative fitness compared to the average
（a个体的繁衍速率有多快）

[Replicator Dynamics. Taylor & Jonker (1978) Mathematical Biosciences]

Replicator Dynamics

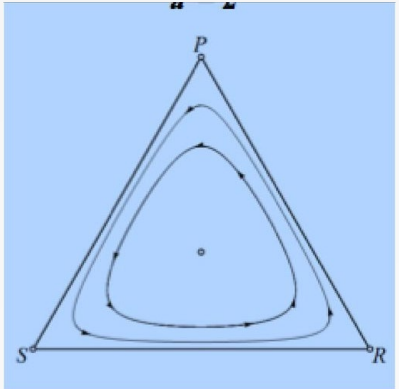
- 马尔可夫/序贯博弈问题



  - 策略：给定状态下的动作概率
  - Advantage：给定状态下的动作优势（相比于策略的期望），决定策略/动作概率的演化

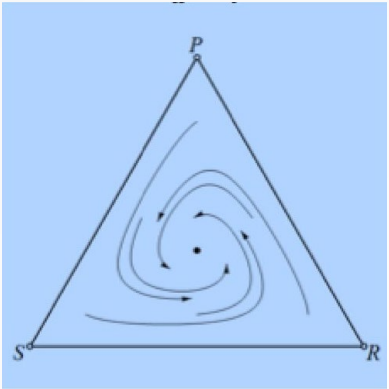
Replicator Dynamics

	R	P	S
R	0	-1	1
P	1	0	-1
S	-1	1	0



(π_R, π_P, π_S) 出现 cycling

	R	P	S
R	0	-1	2
P	2	0	-1
S	-1	2	0



(π_R, π_P, π_S)收敛到

(1/3, 1/3, 1/3)

Replicator Dynamics

- RD is convergent in time average
  - 演化过程中的平均策略收敛到纳什均衡
- DeepNash：Reward transformation [Perolat et al. (2022) Science]

$$r^i(\pi^i, \pi^{-i}, a^i, a^{-i}) = r^i(a^i, a^{-i}) - \eta \log\left(\frac{\pi^i(a^i)}{\pi^i_{\text{reg}}(a^i)}\right) + \eta \log\left(\frac{\pi^{-i}(a^{-i})}{\pi^{-i}_{\text{reg}}(a^{-i})}\right)$$

  - 根据给定的正则化策略 π_{reg}，构造 reward transformation
  - 利用RD求解正则化的博弈问题 （a unique fixed point and guaranteed convergence）
  - 新得到的策略作为下一轮迭代的正则化策略 π_{reg}，多次迭代后收敛到原始纳什均衡解

DeepNash

A

		Player 2	
		Head: H	Tail: T
Player 1	Head: H	1	-1
	Tail: T	-1	1

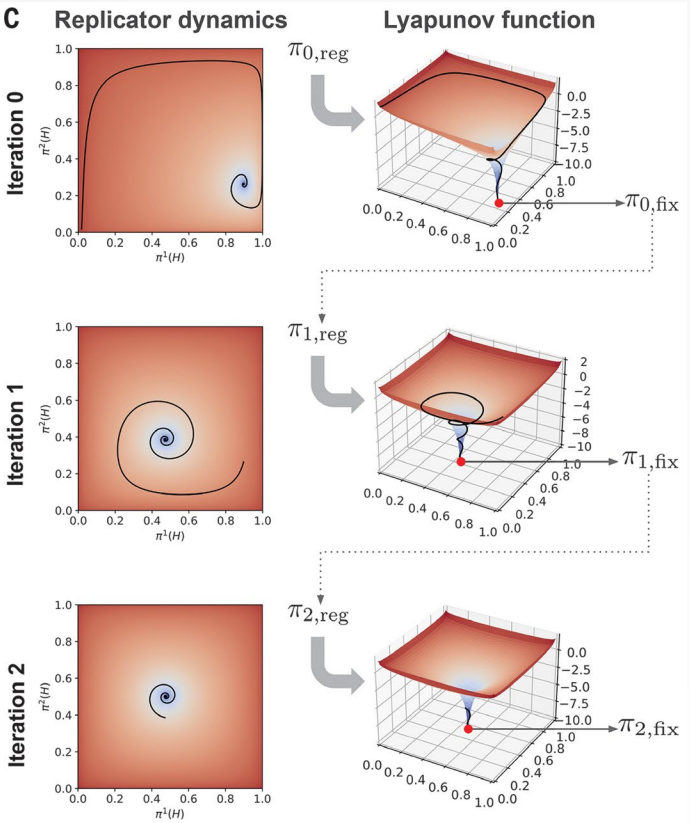
B

R-NaD Iteration

Start with an arbitrary regularization policy: $\pi_{0,\text{reg}}$

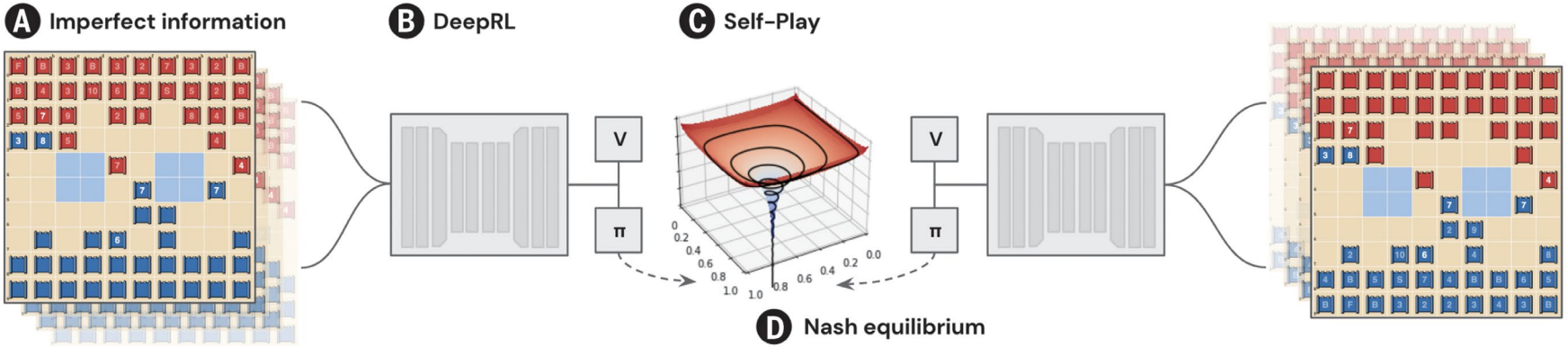
1. Reward transformation: Construct the transformed game with: $\pi_{m,\text{reg}}$
2. Dynamics: Run the replicator dynamics until convergence to: $\pi_{m,\text{fix}}$
3. Update: Set the regularization policy:
$$\pi_{m+1,\text{reg}} = \pi_{m,\text{fix}}$$

Repeat stages until convergence



[Perolat et al. (2022) Science]

DeepNash



Replicator dynamics: $\frac{d}{d\tau}\pi_\tau^i(a^i) = \pi_\tau^i(a^i)\left[Q_{\pi_\tau}^i(a^i) - \sum_{b^i}\pi_\tau^i(b^i)Q_{\pi_\tau}^i(b^i)\right]$

Reward transformation: $r^i(\pi^i, \pi^{-i}, a^i, a^{-i}) = r^i(a^i, a^{-i}) - \eta\log\left(\frac{\pi^i(a^i)}{\pi_{\text{reg}}^i(a^i)}\right) + \eta\log\left(\frac{\pi^{-i}(a^{-i})}{\pi_{\text{reg}}^{-i}(a^{-i})}\right)$

DeepNash beats existing state-of-the-art AI methods in Stratego and achieved a yearly (2022) and all-time top-3 rank on the Gravon games platform, competing with human expert players.

连续时间学习动态

Zhu et al. (2022) Empirical policy optimization for n-player Markov games

■ **Continuous-time learning dynamics (CTLD)**

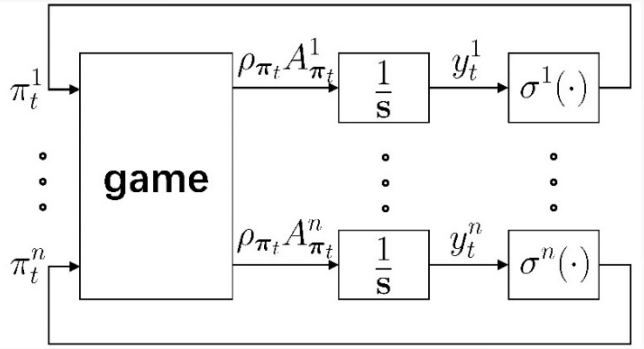
Score function

State visiting freq

Advantage function

$$\begin{cases} \dot{y}_t^i(s,a) = \eta\left[\rho_{\pi_t}(s) A_{\pi_t}^i(s,a) - y_t^i(s,a)\right] \\ \pi_t^i(s) = \arg\max_{\pi} \sum_a \pi(s,a) y_t^i(s,a) - \epsilon \mathcal{H}(\pi(s)) \end{cases}$$

Gibbs entropy function



CTLD框架

– 定义能量函数，使用 LaSalle’s invariance principle (拉塞尔不变性原理) 证明系统稳定性

  • Fenchel-coupling function $F^i(\pi^i, y^i) = \max_{\pi \in \Pi^i} \sum_s \left(\sum_{a^i} y^i(s,a^i)\pi(s,a^i) - h^i(\pi(\cdot|s)) \right) - \sum_s \left(\sum_{a^i} y^i(s,a^i)\pi^i(s,a^i) - h^i(\pi^i(\cdot|s)) \right)$

  • 能量函数随时间呈 non-increasing $\dot{\mathcal{V}}(\mathbf{y}_t) \le -\frac{1}{2}\eta(K-2\mu)\left\|\boldsymbol{\sigma}(\mathbf{y}_t) - \boldsymbol{\sigma}(\bar{\mathbf{y}})\right\|^2$

– 当博弈收益函数是 concave，系统稳定点与近似纳什均衡点一致

– 结论对**任意数量**玩家的博弈问题成立

协作多智能体强化学习
Cooperative MARL

协作多智能体强化学习

- 完全协作的多智能体系统
  - 所有智能体共享同一个奖励函数

$$R^1 = R^2 = \cdots = R^n = R$$

  - 所有智能体目标是最大同一个期望累加奖励/价值函数

集中式的MDP

$$Q(s_t, \boldsymbol{a}_t) \leftarrow Q(s_t, \boldsymbol{a}_t) + \alpha[r_t + \gamma \max_{\boldsymbol{a}_{t+1} \in \boldsymbol{\mathcal{A}}} Q(s_{t+1}, \boldsymbol{a}_{t+1}) - Q(s_t, \boldsymbol{a}_t)]$$

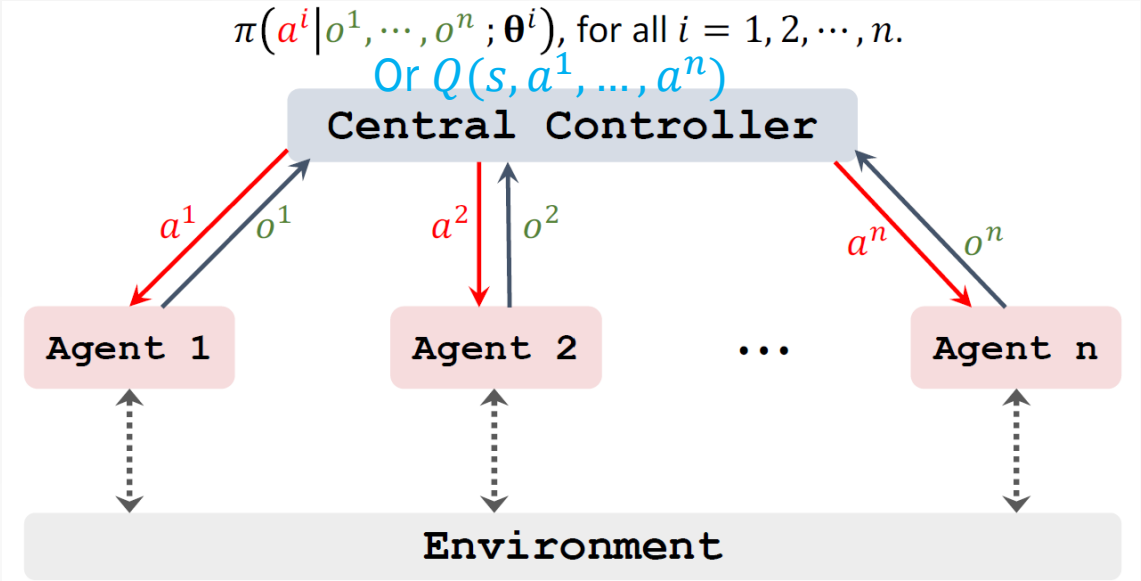
全局状态　联合动作

$$\boldsymbol{a}_t = \arg\max Q(s_t, \boldsymbol{a}_t)$$

    - 优势：所有单智能体RL算法都能用

Cooperative-MARL的典型结构

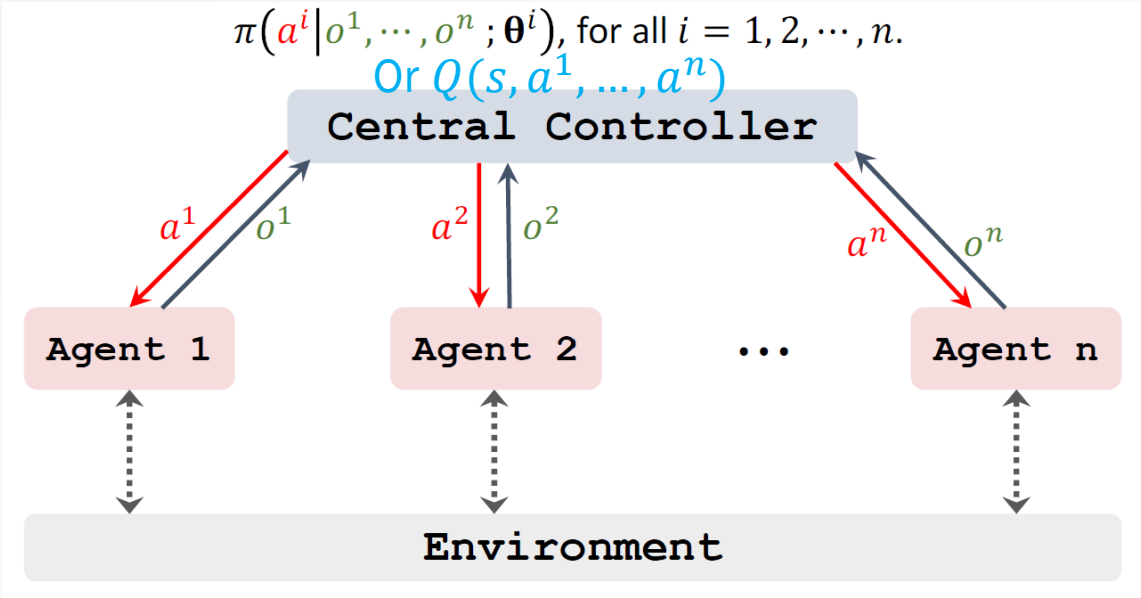
- 完全集中式 fully centralized



  - 所有智能体将观测发送给中央控制器，计算所有动作后分发给各个智能体
  - 通信和同步消耗时间，不利于实时决策

Cooperative-MARL的典型结构

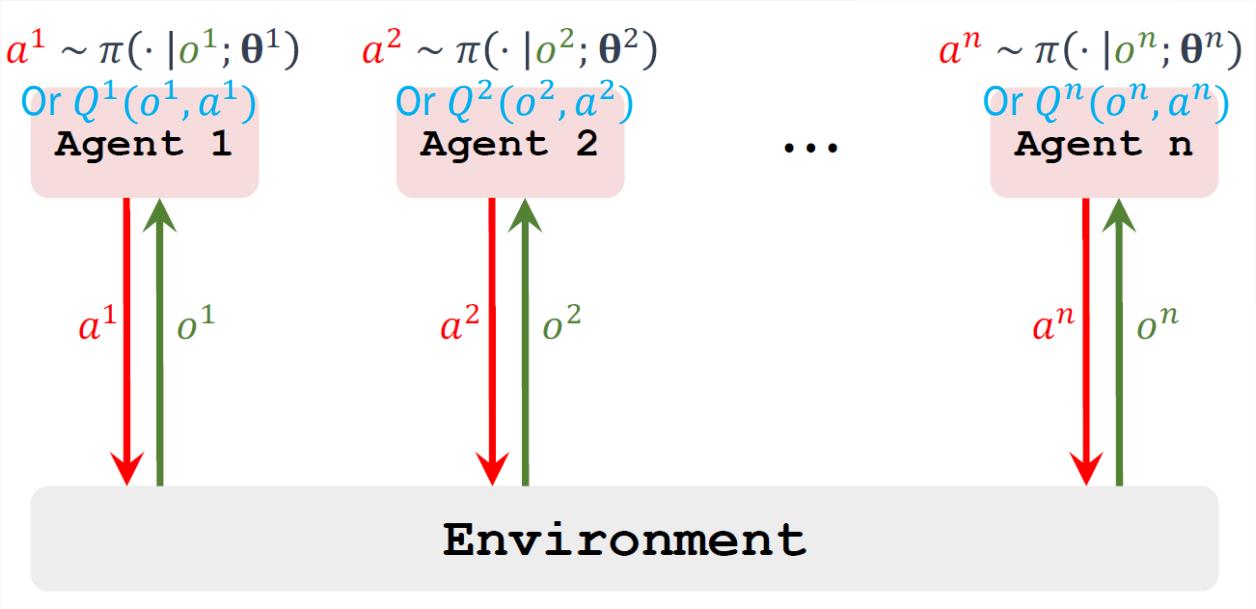
- 完全集中式 fully centralized



  - 算法复杂度随智能体数量呈指数增长 (维数灾) $Q(\mathcal{O}^1\times\cdots\times\mathcal{O}^n,\mathcal{A}^1\times\cdots\times\mathcal{A}^n)$
  - 通常智能体只基于自身观测独立决策，无法集中式选择联合最优动作 $\pi_i:\Omega^i\to\Delta(\mathcal{A}^i)$

Cooperative-MARL的典型结构

- 完全分布式 fully decentralized



  - 每个智能体按照 single-agent RL 训练各自的value/policy networks
  - 效果不好（非静态环境、高方差、易发散）

Cooperative-MARL的典型结构

- 完全分布式 fully decentralized
  - e.g. Independent Q learning

$$Q_i(s_t, a_t^i) \leftarrow Q_i(s_t, a_t^i) + \alpha[r_{t+1} + \gamma \max_{a_{t+1}^i \in \mathcal{A}^i} Q_i(s_{t+1}, a_{t+1}^i) - Q_i(s_t, a_t^i)].$$

$$\pi_i(a_t^i|s_t) = \mathbb{1}\{a_t^i = \arg\max_{a_t^i} Q_i(s_t, a_t^i)\}.$$

Tan (1993) Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents

  - 缺陷：非静态问题 non-stationary

$$P_{\pi^{-i}}(s'|s, a^i) \neq P_{\bar{\pi}^{-i}}(s'|s, a^i)$$

重要性采样

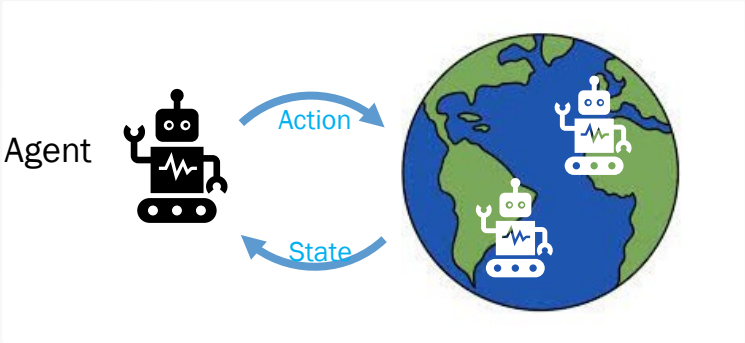
当前时刻其它智能体策略概率

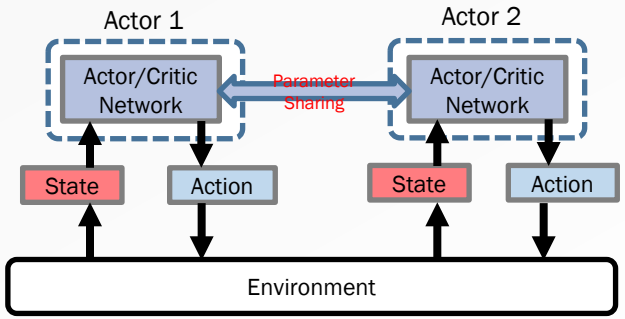
$$\mathcal{L}(\theta_i) = \frac{\boldsymbol{\pi}_{-i}^{t_c}(\boldsymbol{a}^{-i}|\boldsymbol{o}^{-i})}{\boldsymbol{\pi}_{-i}^{t_i}(\boldsymbol{a}^{-i}|\boldsymbol{o}^{-i})}[(y_i^{\mathrm{Q}} - Q_i(o^i, a^i; \theta_i))^2],$$

样本采集时其它智能体策略概率

Cooperative-MARL的典型结构

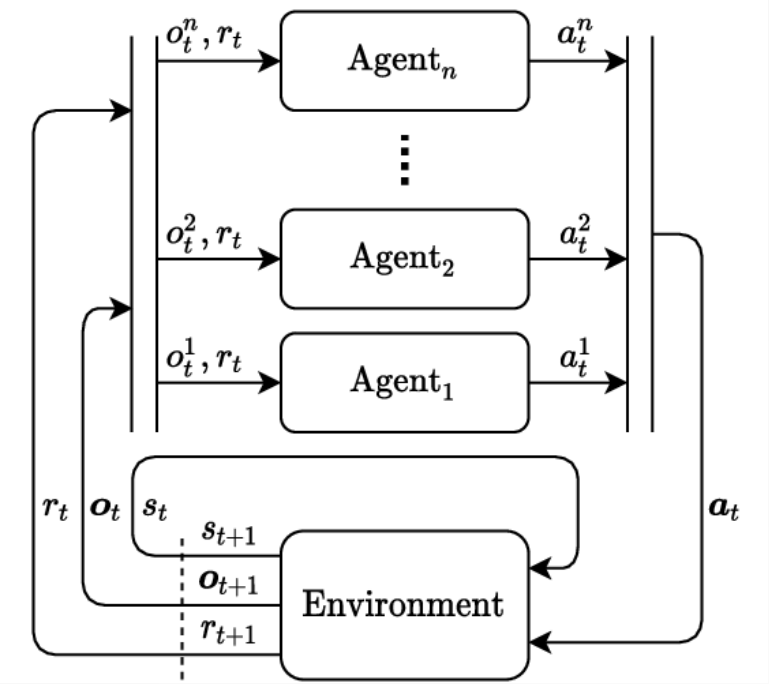
- 完全分布式 fully decentralized
  - 优势：
    - 将其它智能体看作环境一部分
    - 每个智能体的决策和训练是分布/独立的，易扩展
    - 每个智能体使用single-agent RL学习
    - 同构智能体策略/价值网络可共享参数





部分可观测
Partial observability

- Dec-POMDP (decentralized partially-observable MDP)



- 局部观测o_t^i 是全局状态s_t 的部分信息
- 分布式策略依赖局部观测决策 $a_t^i \sim \pi^i(o_t^i)$
- 但会缺失关键状态信息，性能受限
- 依靠局部观测的历史去决策

$$a_t^i \sim \pi^i(\dots o_{t-2}^i, o_{t-1}^i, o_t^i)$$

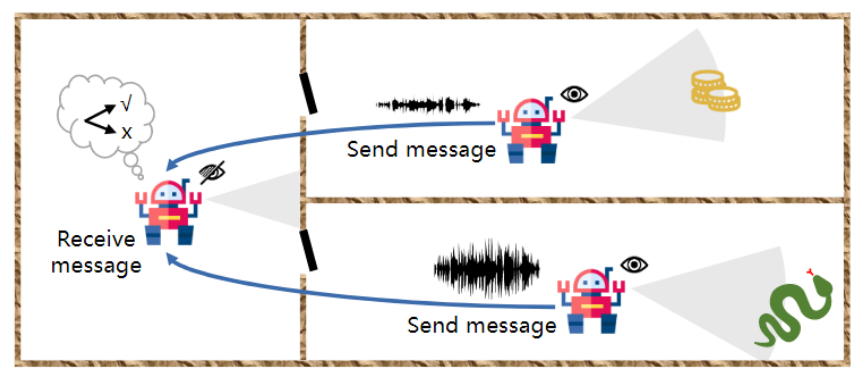
  - 从观测历史推断全局状态
  - e.g. 从LiDAR观测相对距离$\dots, \Delta d_{t-1}, \Delta d_t$推断相对速度$\Delta v_t \approx \frac{\Delta d_t - \Delta d_{t-1}}{\Delta t}$
- 递归神经网络RNN, LSTM, GRU，序列建模神经网络 Transformer

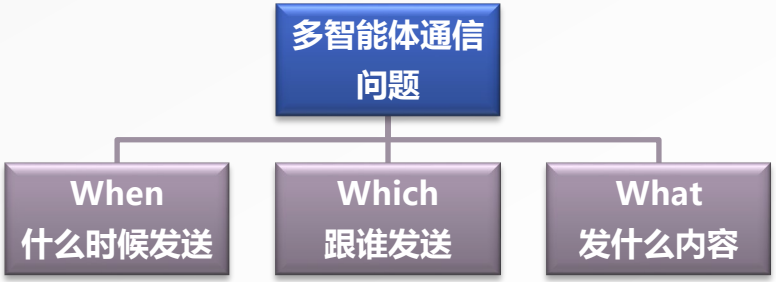
多智能体通信

- 智能体在分布式设计下部分可观测
  - 无法获得全局信息，影响决策（$\mathcal{O}^i \in \mathcal{S}$）
  - 局部样本不具有马尔可夫性，影响训练（$P_{\pi^{-i}}(s'|s,a^i) \neq P_{\overline{\pi}^{-i}}(s'|s,a^i)$）
- 多智能体通信：智能体通过通信获得其它智能体观测，扩大局部观测视野，提升决策和训练效果

$$a^i \sim \pi^i\left(o^i, \{m^j\}_{j\in\mathcal{N}_i}\right)$$

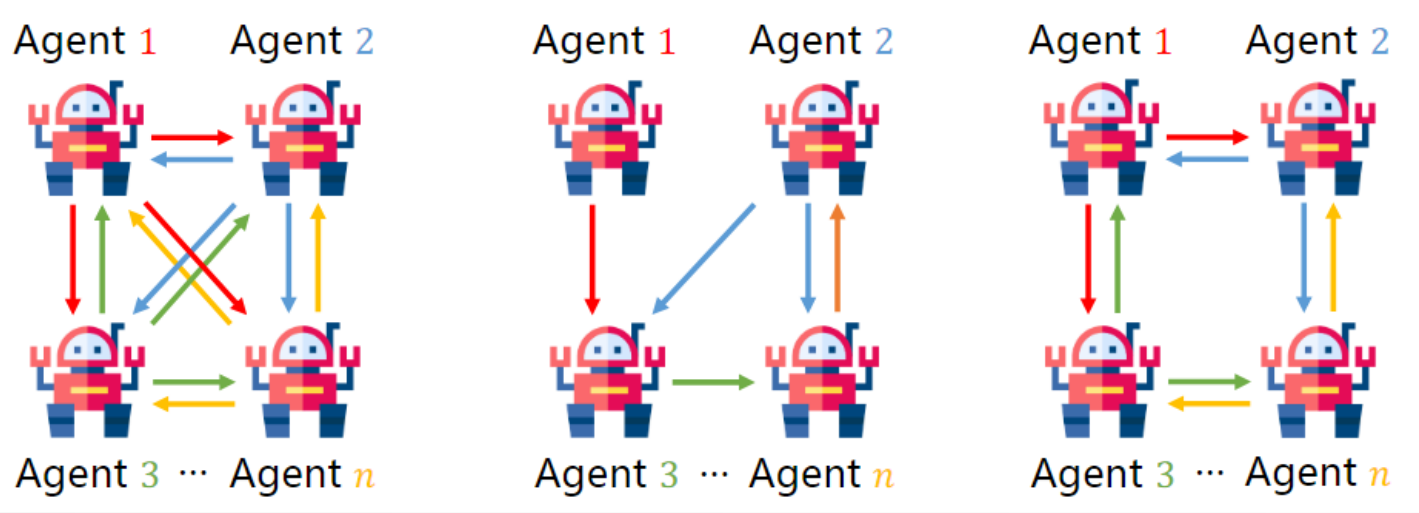
  - $\mathcal{N}_i$代表与智能体i 建立通信的智能体集合
  - m^j代表智能体 j 的通信内容





多智能体通信

- 典型通信拓扑结构



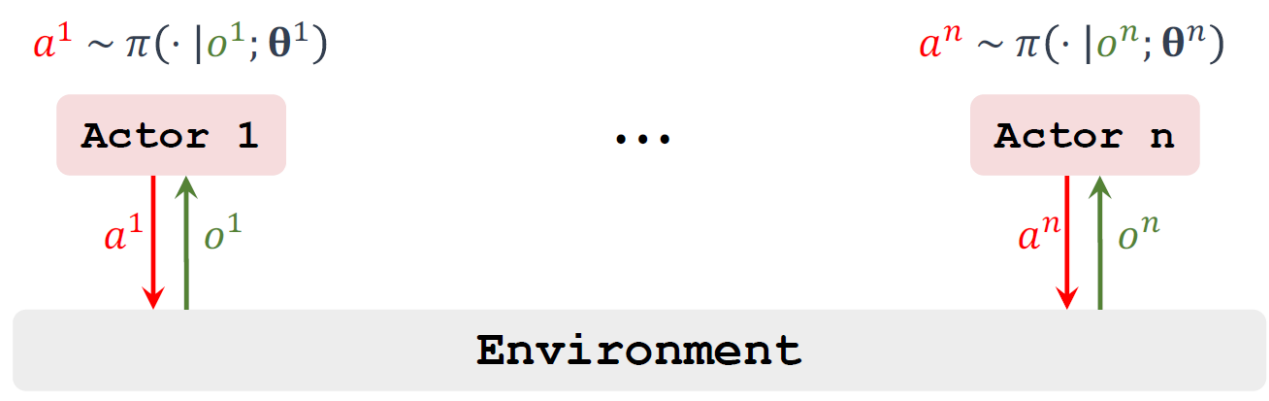
(a) 广播式：消息发送给所有人

(b) 目标式：消息发送给指定人

(c) 邻域式：消息发送给周围一定范围人

Cooperative-MARL的典型结构

- 集中式训练-分布式执行结构

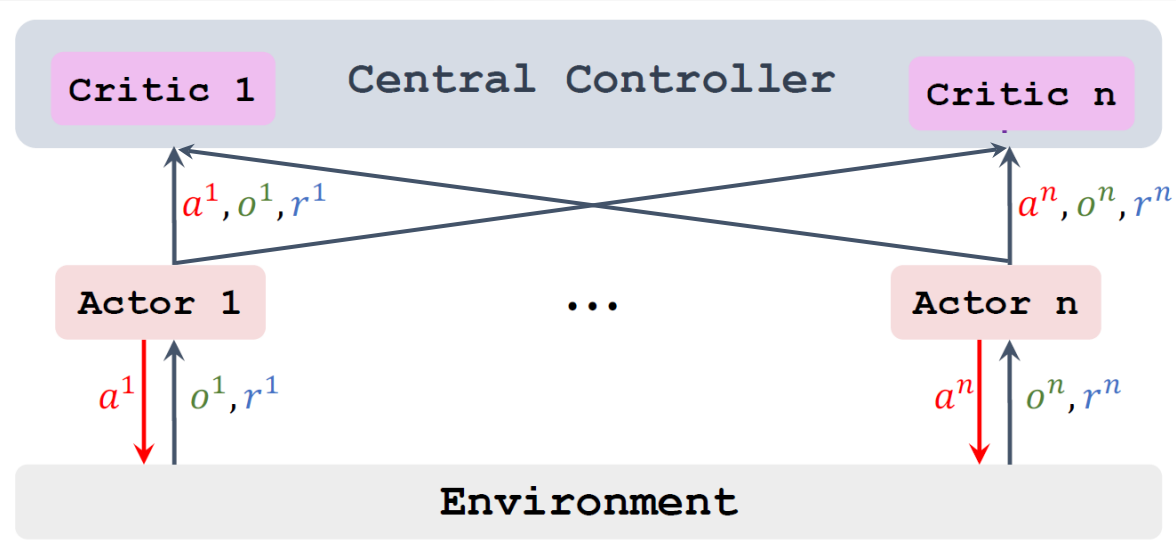


– 每个智能体拥有独立的策略网络 actor：

$$\pi_1(a^1|o^1,\theta^1),\pi_2(a^2|o^2,\theta^2),\cdots,\pi_n(a^n|o^n,\theta^n).$$

Cooperative-MARL的典型结构

- 集中式训练-分布式执行结构

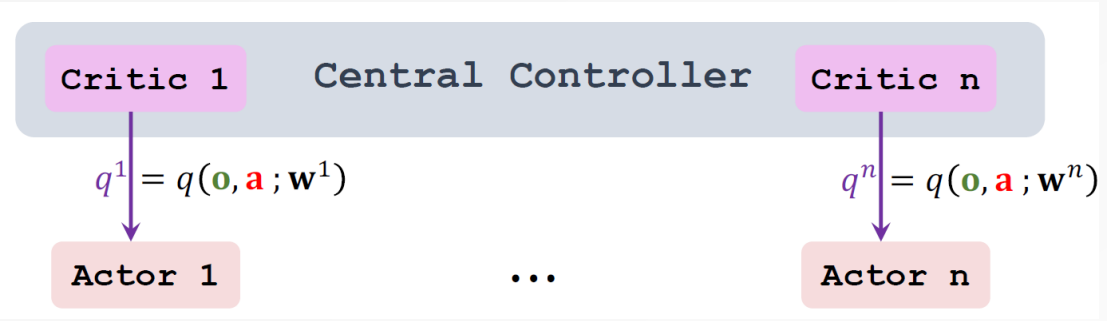


– 中央控制器拥有 n 个价值网络 (critics)： $q(\boldsymbol{o}, \boldsymbol{a}; w^i)$

– 分别近似各自智能体的目标： $J^i(\theta^1, \ldots, \theta^n) = \mathbb{E}_{\boldsymbol{\pi}}[V^i(s)]$

Cooperative-MARL的典型结构

- 集中式训练-分布式执行结构



– 对每个智能体策略参数沿梯度上升方向更新

$$\theta^i \leftarrow \theta^i + \alpha^i \nabla_{\theta^i} J(\theta^1, \cdots, \theta^n).$$

协同多智能体策略梯度定理:

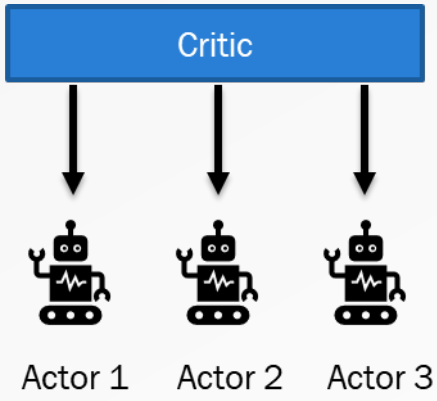
$$\boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{o}, \boldsymbol{\theta}) = \pi_1(a^1|o^1; \theta^1) \times \cdots \times \pi_n(a^n|o^n; \theta^n)$$

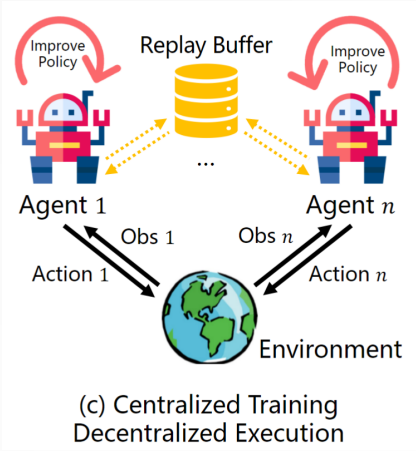
$$\nabla_{\theta^i} J(\theta^1, \cdots, \theta^n) = \mathbb{E}[\nabla_{\theta^i} \log \pi_i(a^i|o^i, \theta^i) \cdot (Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) - b)].$$

对数似然梯度　　联合Q函数　　基线baseline

集中式训练-分布式执行

- 集中式训练-分布式执行
  - Centralized Training Decentralized Execution (CTDE)
  - 集中式训练：智能体在训练策略时利用其它智能体和全局信息（评估精确）
  - 分布式执行：智能体在采样动作时只基于自身局部信息　　（执行方便）
- 目前cooperative MARL领域的热点方向

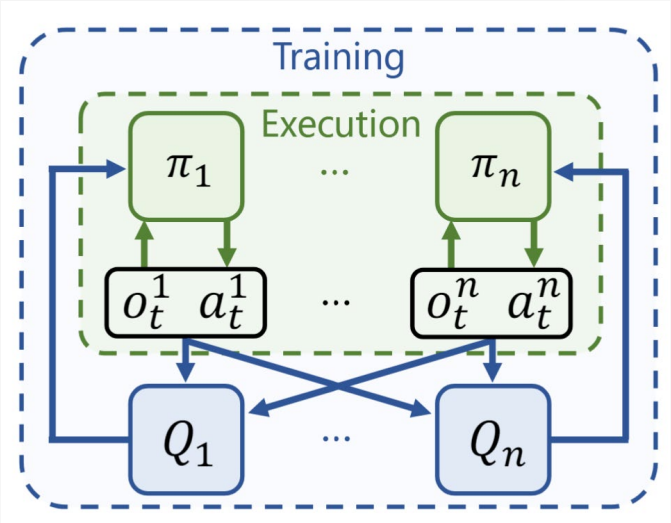




(c) Centralized Training Decentralized Execution

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Lowe, et al.(2017) Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments



Framework of MADDPG

每个智能体对应各自的联合Q_i函数，因此MADDPG即可解决cooperative，也可以解决competitive问题

Algorithm 1 MADDPG Algorithm

for $episode = 1$ **to** M **do**
  Initialize random exploration noise $\mathcal{N}$
  Obtain initial state s
  for $t = 1$ **to** maximum trajectory length **do**
    Each agent explores based on the current policy, obtaining actions: $a^i = \pi_i(o^i;\theta^i) + \mathcal{N}_t$
    All agents execute joint actions $\boldsymbol{a} = (a^1, \cdots, a^n)$, obtaining rewards r and new state s'
    Store $(s, \boldsymbol{a}, r, s')$ in the experience pool $\mathcal{D}$
    $s \leftarrow s'$
    for agent $i = 1$ **to** n **do**
      Sample samples $\{s_j, \boldsymbol{a}_j, r_j, s'_j\}_{j=1}^{bs}$ from experience pool $\mathcal{D}$
      Optimize Critic network according to Equation 15
      Optimize Actor network according to Equation 16
    end for
    Update target neural networks: $\theta^i_{\_} = \tau\theta^i + (1-\tau)\theta^i_{\_}$
  end for
end for

$$y_i^Q = r + \gamma Q_i(s', \boldsymbol{a}'|\phi^i)|_{a'^j=\pi'_j(o'^j;\theta^j_{\_})},$$

$$\mathcal{L}(\phi^i) = \frac{1}{2}(y_i^Q - Q_i(s, \boldsymbol{a}; \phi^i))^2.$$

$$\nabla_{\theta^i} J(\theta^i) = \nabla_{\theta^i}\pi_i(o^i;\theta^i)\nabla_{a^i}Q_i(s,\boldsymbol{a};\phi^i)|_{a^j=\pi_j(o^j|\theta^j)}$$

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Lowe, et al.(2017) Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments

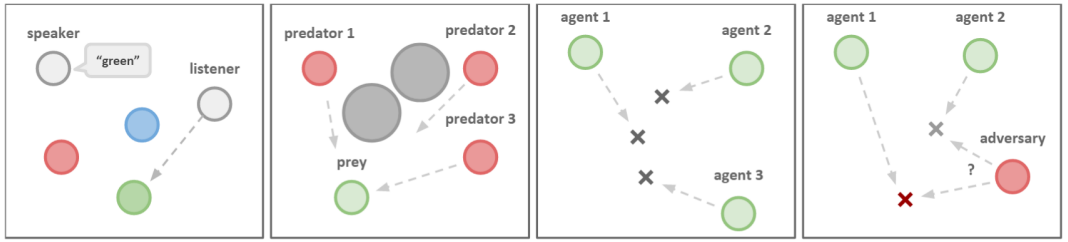


Figure 2: Illustrations of the experimental environment and some tasks we consider, including a) *Cooperative Communication* b) *Predator-Prey* c) *Cooperative Navigation* d) *Physical Deception*. See webpage for videos of all experimental results.

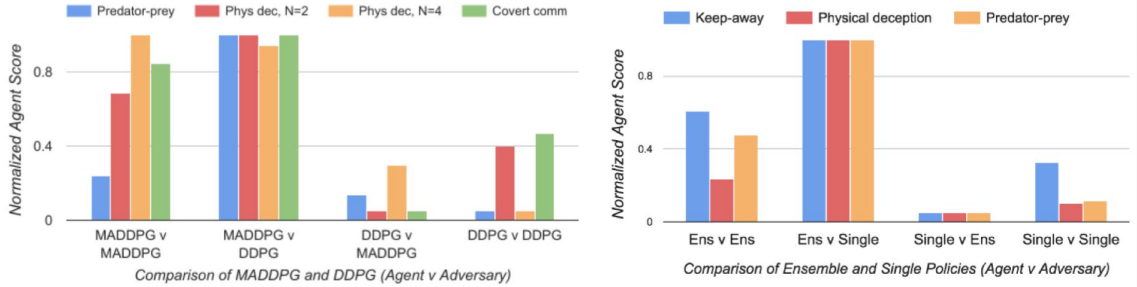


Figure 3: Comparison between MADDPG and DDPG (left), and between single policy MADDPG and ensemble MADDPG (right) on the competitive environments. Each bar cluster shows the 0-1 normalized score for a set of competing policies (agent v adversary), where a higher score is better for the agent. In all cases, MADDPG outperforms DDPG when directly pitted against it, and similarly for the ensemble against the single MADDPG policies. Full results are given in the Appendix.

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Lowe, et al.(2017) Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments

https://github.com/openai/multiagent-particle-envs

(回顾)强化学习分类

- 基于价值
  - 学习价值函数 $Q(s,a)$
  - 策略直接从价值提取 $a^* = \arg\max Q(s,\cdot)$
- 基于策略
  - 学习策略函数 $\pi(a|s)$
  - 基于轨迹优化策略 $\Delta\pi = \nabla J(\pi)$
- Actor-Critic
  - 学习价值函数 $Q(s,a)$
  - 学习策略优化 $\pi \leftarrow \mathbb{E}[\log \pi(s,a)\, Q(s,a)]$

学习效率高
离散动作

学习难度大
离散和连续动作

值函数分解 value decomposition

独立全局最大化原理 individual-global-max principle：多智能体系统根据联合Q_{tot}选择最优联合动作，等价于每个智能体根据个体Q_i选择最优动作

[Rashid et al. (2018)]

$$\arg\max_{\boldsymbol{a}} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a^1} Q_1(\tau^1, a^1) \\ \vdots \\ \arg\max_{a^n} Q_n(\tau^n, a^n) \end{pmatrix}$$

轨迹输入 $\tau_t^i = (o_1, a_1, ..., a_{t-1}, o_t) \in \mathcal{T}^i$ 减轻部分观测的挑战

- 联合Q_{tot}函数：反映系统整体评价指标，计算训练损失（e.g. 时间差分误差）

$$\delta_t = r_t + \gamma \max_{\boldsymbol{a}_{t+1}} Q_{tot}(\boldsymbol{\tau}_{t+1}, \boldsymbol{a}_{t+1}) - Q_{tot}(\boldsymbol{\tau}_t, \boldsymbol{a}_t)$$

- 个体Q_i函数：智能体分布式执行最优动作

$$\arg\max_{a^i} Q_i(\tau^i, a^i)$$

值函数分解 value decomposition

$$\arg\max_{\boldsymbol{a}} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a^1} Q_1(\tau^1, a^1) \\ \vdots \\ \arg\max_{a^n} Q_n(\tau^n, a^n) \end{pmatrix}$$

- VDN **Sunehag et al.(2018) Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward**
  - 联合Q函数是个体Q函数的加和

$$Q_{\text{tot}}^{\text{VDN}}(\boldsymbol{\tau}, \boldsymbol{a}) = \sum_{i=1}^{n} Q_i\left(\tau^i, a^i\right).$$

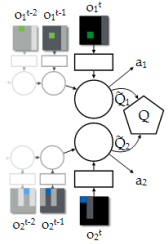


Figure 2: Value-decomposition individual architecture showing how local observations enter the networks of two agents over time (three steps shown), pass through the low-level linear layer to the recurrent layer, and then a dueling layer produces individual "values" that are summed to a joint Q-function for training, while actions are produced independently from the individual outputs.

- QMIX Rashid et al. (2018) QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning
  - 混合网络：输入是个体Q值，输出是联合Q值
  - 混合网络结构和权重保证输出对输入偏导不小于0

$$Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a}) = Mixing(s, Q_1(\tau^1, a^1), ..., Q_n(\tau^n, a^n)),$$
$$\forall i \in \mathcal{N}, \frac{\partial Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q_i\left(\tau^i, a^i\right)} \geq 0.$$

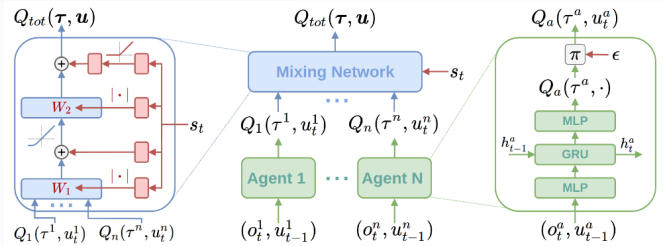


Figure 2. (a) Mixing network structure. In red are the hypernetworks that produce the weights and biases for mixing network layers shown in blue. (b) The overall QMIX architecture. (c) Agent network structure. Best viewed in colour.

QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning

混合网络多层感知机（MLP）结构，每层权重由超网络$h(\cdot)$决定，权重取绝对值：

$$[|W_1|, b_1, |W_2|, b_2] = h(s)$$

IGM条件 $\frac{\partial Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q_i(\tau^i, a^i)} \geq 0$

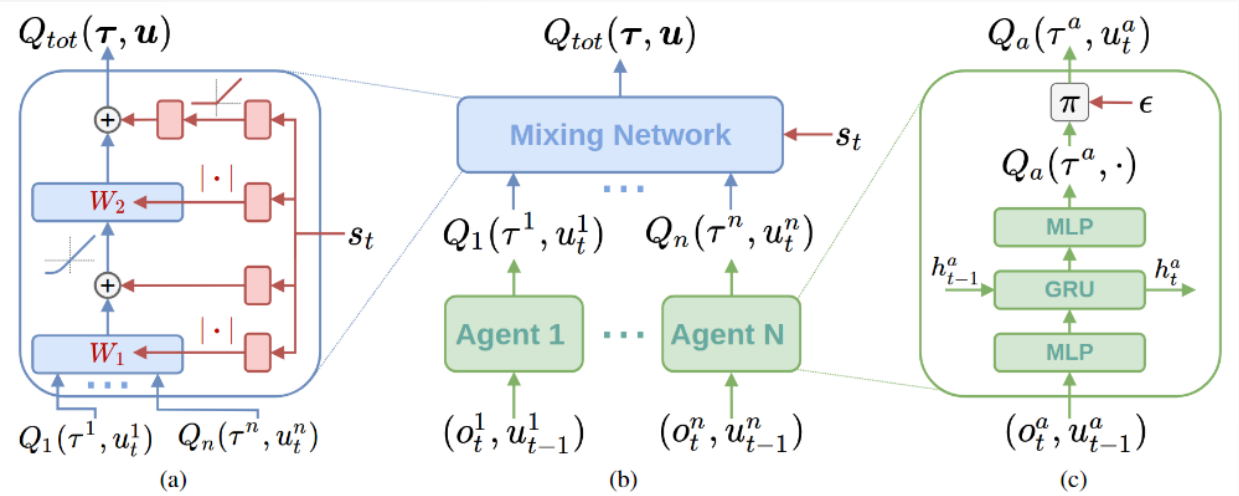
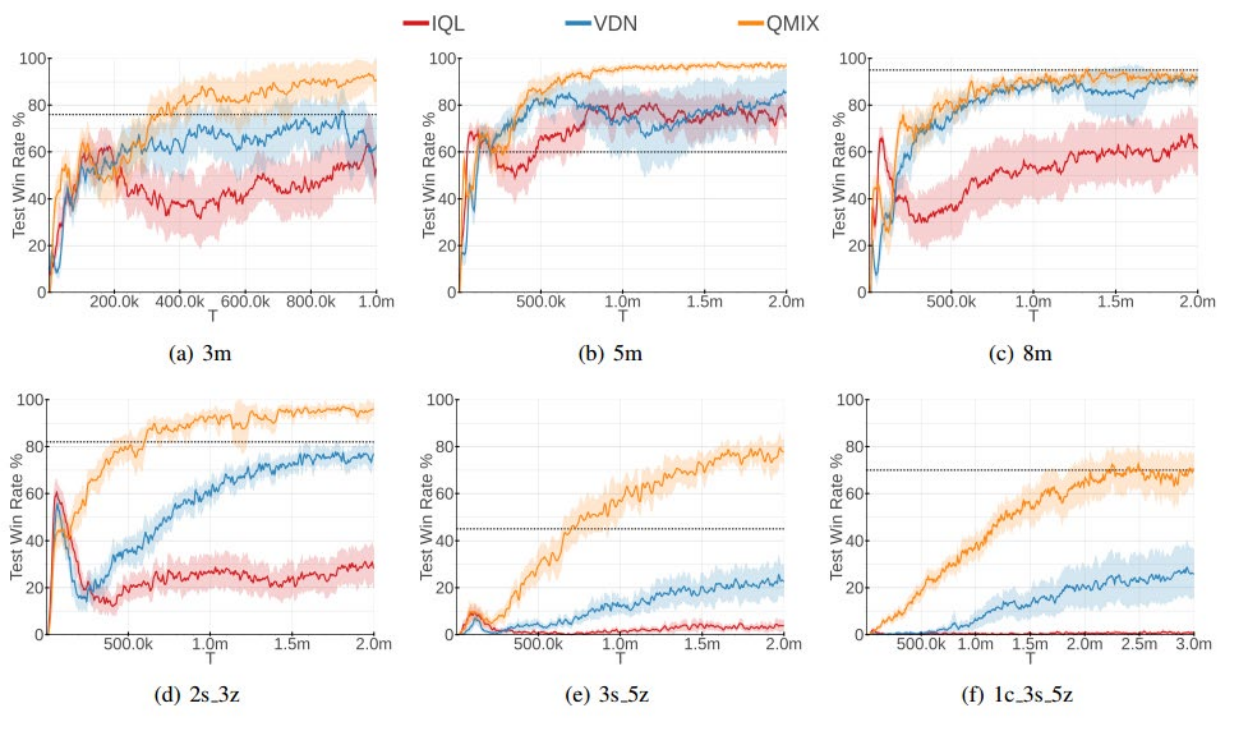


Figure 2. (a) Mixing network structure. In red are the hypernetworks that produce the weights and biases for mixing network layers shown in blue. (b) The overall QMIX architecture. (c) Agent network structure. Best viewed in colour.

训练损失 $\mathcal{L}_{\text{QMIX}} = (Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a}) - y_{\text{tot}})^2, \quad y_{\text{tot}} = r + \gamma \max_{\boldsymbol{a}'} Q_{\text{tot}}^{\text{QMIX},-}(\boldsymbol{\tau}', \boldsymbol{a}')$

QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning

- 星际争霸II微操实验



QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning

- 星际争霸II微操实验

https://github.com/starry-sky6688/StarCraft

The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games

Chao Yu[1♯,*], **Akash Velu**[2♮,*], **Eugene Vinitsky**[2♭], **Jiaxuan Gao**[1], **Yu Wang**[1♭], **Alexandre Bayen**[2], **Yi Wu**[13♭]
[1] Tsinghua University [2] University of California, Berkeley [3] Shanghai Qi Zhi Institute
♯zoeyuchao@gmail.com, ♮akashvelu@berkeley.edu

Abstract

Proximal Policy Optimization (PPO) is a ubiquitous on-policy reinforcement learning algorithm but is significantly less utilized than off-policy learning algorithms in multi-agent settings. This is often due to the belief that PPO is significantly

PPO vs MAPPO

- **PPO loss**

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

  - $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$
  - $A_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V, \quad \delta_t^V = r_t + \gamma V(s_{t+l}) - V(s_t)$

    (generalized advantage estimator)

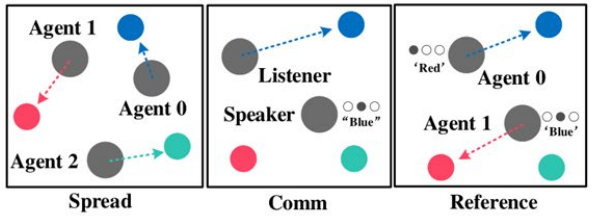
- **MAPPO loss**

$$L^{MAPPO(CLIP)}(\theta) = \mathbb{E}_t[\min(r_t^{(k)}(\theta)A_t^{(k)}, clip(r_t^{(k)}(\theta), 1-\epsilon, 1+\epsilon)A_t^{(k)})]$$

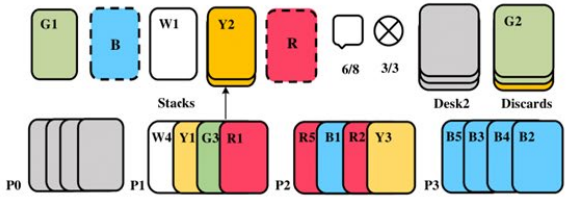
  - $r_t^{(k)}(\theta) = \frac{\pi_\theta(a_t^{(k)}|o_t^{(k)})}{\pi_{\theta_{old}}(a_t^{(k)}|o_t^{(k)})}$

Experiment scenarios

- Multi-agent particle-world environment (MPE)
- StarCraft multi-agent challenge (SMAC)
- Google Research Football (GRF)
- Hanabi Challenge



(a) MPE scenarios



(b) 4-player Hanabi-Full

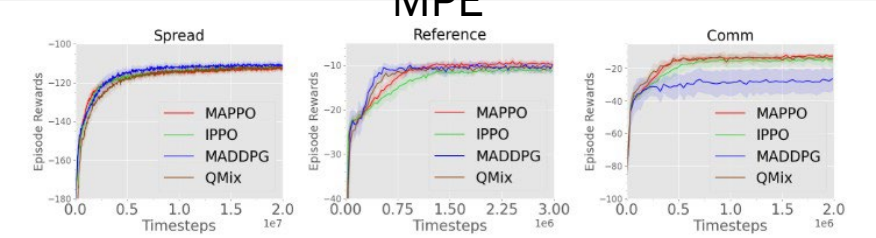
(c) SMAC corridor

(d) SMAC 2c_vs_64zg

(e) GRF academy 3 vs 1 with keeper

Results and Performance

MPE



SMAC

Map	MAPPO(FP)	MAPPO(AS)	IPPO	QMix	RODE*	MAPPO*(FP)	MAPPO*(AS)
2m vs_1z	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.0)	**95.3**(5.2)	/	*100.0*(0.0)	*100.0*(0.0)
3m	**100.0**(0.0)	**100.0**(1.5)	**100.0**(0.0)	96.9(1.3)	/	*100.0*(0.0)	*100.0*(1.5)
2svs1sc	**100.0**(0.0)	**100.0**(0.0)	**100.0**(1.5)	96.9(2.9)	*100.0*(0.0)	*100.0*(0.0)	*100.0*(0.0)
2s3z	**100.0**(0.7)	**100.0**(1.5)	**100.0**(0.0)	95.3(2.5)	*100.0*(0.0)	96.9(1.5)	96.9(1.5)
3svs3z	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.0)	**96.9**(12.5)	/	*100.0*(0.0)	*100.0*(0.0)
3svs4z	**100.0**(1.3)	**98.4**(1.6)	**99.2**(1.5)	**97.7**(1.7)	/	*100.0*(2.1)	*100.0*(1.5)
so many baneling	**100.0**(0.0)	**100.0**(0.7)	**100.0**(1.5)	96.9(2.3)	/	*100.0*(1.5)	96.9(1.5)
8m	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.7)	97.7(1.9)	/	*100.0*(0.0)	*100.0*(0.0)
MMM	**96.9**(0.6)	93.8(1.5)	**96.9**(0.0)	**95.3**(2.5)	/	*93.8*(2.6)	*96.9*(1.5)
1c3s5z	**100.0**(0.0)	96.9(2.6)	**100.0**(0.0)	96.1(1.7)	*100.0*(0.0)	*100.0*(0.0)	96.9(2.6)
bane vs bane	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.0)	**100.0**(0.0)	*100.0*(46.4)	*100.0*(0.0)	*100.0*(0.0)
3svs5z	**100.0**(0.6)	**99.2**(1.4)	**100.0**(0.0)	**98.4**(2.4)	78.9(4.2)	*98.4*(5.5)	*100.0*(1.2)
2cvs64zg	**100.0**(0.0)	**100.0**(0.0)	98.4(1.3)	92.2(4.0)	*100.0*(0.0)	*96.9*(3.1)	95.3(3.5)
8mvs9m	**96.9**(0.6)	**96.9**(0.6)	**96.9**(0.7)	92.2(2.0)	/	*84.4*(5.1)	*87.5*(2.1)
25m	**100.0**(1.5)	**100.0**(4.0)	**100.0**(0.0)	85.9(7.1)	/	*96.9*(3.1)	*93.8*(2.9)
5mvs6m	**89.1**(2.5)	**88.3**(1.2)	**87.5**(2.3)	75.8(3.7)	*71.1*(9.2)	*65.6*(14.1)	*68.8*(8.2)
3s5z	**96.9**(0.7)	**96.9**(1.9)	**96.9**(1.5)	88.3(2.9)	*93.8*(2.0)	71.9(11.8)	53.1(15.4)
10mvs11m	**96.9**(4.8)	**96.9**(1.2)	**93.0**(7.4)	**95.3**(1.0)	*95.3*(2.2)	81.2(8.3)	*89.1*(5.5)
MMM2	**90.6**(2.8)	**87.5**(5.1)	**86.7**(7.3)	**87.5**(2.6)	*89.8*(6.7)	51.6(21.9)	28.1(29.6)
3s5zvs3s6z	**84.4**(34.0)	63.3(19.2)	**82.8**(19.1)	**82.8**(5.3)	*96.8*(25.11)	*75.0*(36.3)	18.8(37.4)
27mvs30m	**93.8**(2.4)	85.9(3.8)	69.5(11.8)	39.1(9.8)	*96.8*(1.5)	*93.8*(3.8)	*89.1*(6.5)
6hvs8z	**88.3**(3.7)	**85.9**(30.9)	**84.4**(33.3)	9.4(2.0)	*78.1*(37.0)	*78.1*(5.6)	*81.2*(31.8)
corridor	**100.0**(1.2)	**98.4**(0.8)	**98.4**(3.1)	84.4(2.5)	*65.6*(32.1)	*93.8*(3.5)	*93.8*(2.8)

GRF

Scen.	MAPPO	QMix	CDS	TiKick
3v.1	**88.03**(1.06)	8.12(2.83)	76.60(3.27)	76.88(3.15)
CA(easy)	**87.76**(1.34)	15.98(2.85)	63.28(4.89)	/
CA(hard)	**77.38**(4.81)	3.22(1.60)	58.35(5.56)	73.09(2.08)
Corner	**65.53**(2.19)	16.10(3.00)	3.80(0.54)	33.00(3.01)
PS	**94.92**(0.68)	8.05(3.66)	**94.15**(2.54)	/
RPS	**76.83**(1.81)	8.08(4.71)	62.38(4.56)	79.12(2.06)

Hanabi

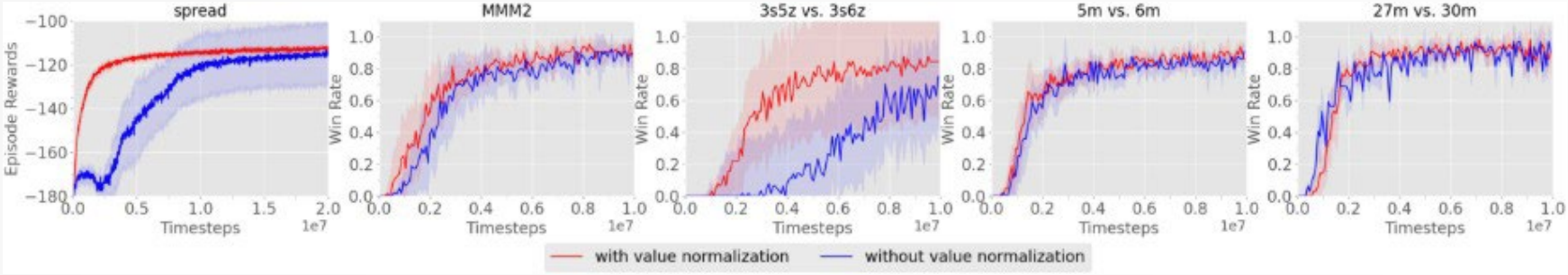
# Players	Metric	MAPPO	IPPO	SAD	VDN
2	Avg.	23.89(0.02)	**24.00**(0.02)	23.87(0.03)	23.83(0.03)
	Best	**24.23**(0.01)	24.19(0.02)	24.01(0.01)	23.96(0.01)
3	Avg.	**23.77**(0.20)	23.25(0.33)	23.69(0.05)	23.71(0.06)
	Best	**24.01**(0.01)	23.87(0.03)	23.93(0.01)	23.99(0.01)
4	Avg.	**23.57**(0.13)	22.52(0.37)	23.27(0.26)	23.03(0.15)
	Best	23.71(0.01)	23.06(0.03)	**23.81**(0.01)	23.79(0.00)
5	Avg.	**23.04**(0.10)	20.75(0.56)	22.06(0.23)	21.28(0.12)
	Best	**23.16**(0.01)	22.54(0.02)	23.01(0.01)	21.80(0.01)

- **agent-specific (AS):** uses the concatenation of the environment provided global state and agent i's observation, oi, as the global input to MAPPO's critic during gradient updates for agent i.
- **feature-pruned global state (FP):** removes the overlapping features in the AS global state, since the global state and local agent observations have overlapping features

Factors Influential to PPO’s Performance

Value Normalization

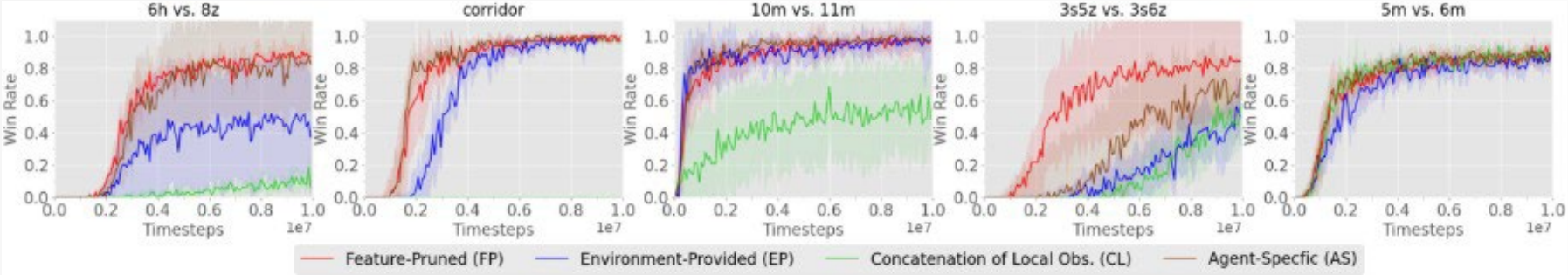
- the scale of the reward functions can vary vastly across environments, and having large reward scales can destabilize value learning.
- We thus use value normalization to normalize the regression targets into a range between 0 and 1 during value learning, and find that this often helps and never hurts MAPPO’s performance.

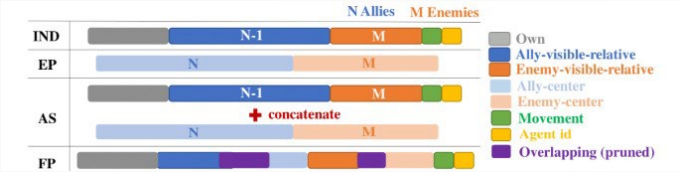


Factors Influential to PPO’s Performance

Value Function Inputs

- Since the value-function is solely used during training updates and is not needed to compute actions, it can utilize global information to make more accurate predictions.
- This practice is common in other multi-agent policy gradient methods and is referred to as centralized training with decentralized execution. We evaluate MAPPO with several global state inputs, as well as local observation inputs.



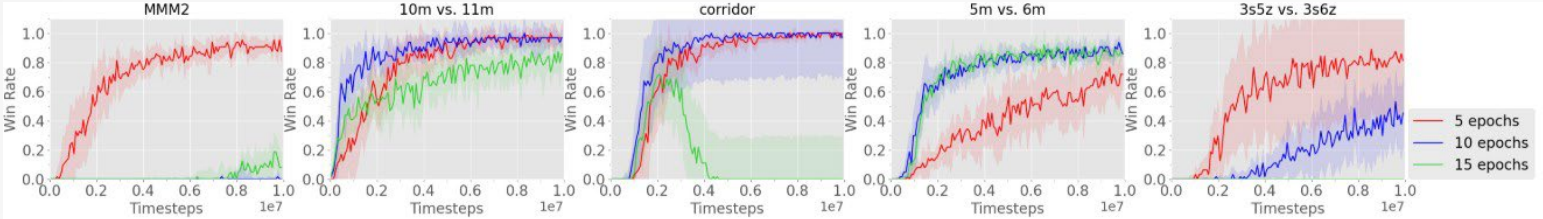


CL(concatenation of local observation)
EP(Environment-Provided global state)
AS(Agent-Specific Global State)
FP(Featured-Pruned Agent-Specific Global State)

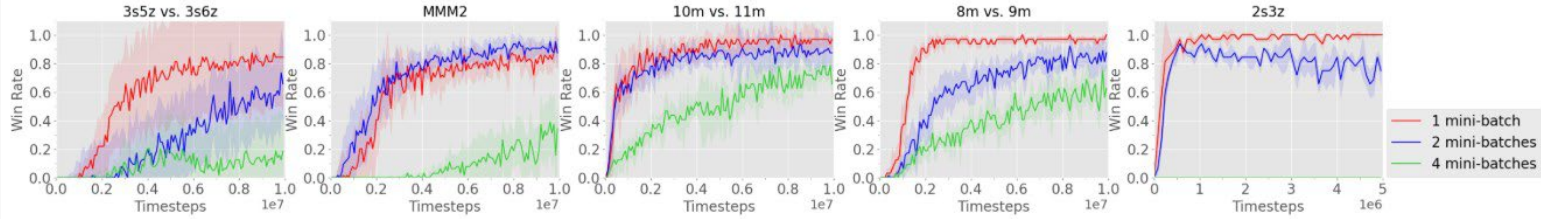
Factors Influential to PPO’s Performance

Training Data Usage

- It is typical for PPO to perform many epochs (10~) of updates on a batch of training data using mini-batch gradient descent.
- We find that high data reuse is detrimental in multi-agent settings;
  - *we recommend using 15 training epochs for easy tasks, and 10 or 5 epochs for more difficult tasks.*



(a) effect of different training epochs.



(b) effect of different mini-batch numbers.

Factors Influential to PPO’s Performance

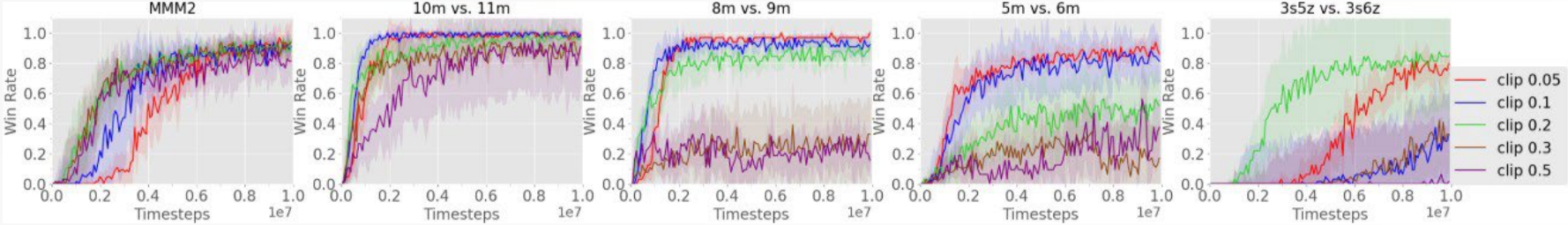
Training Data Usage

- It is typical for PPO to perform many epochs (10~) of updates on a batch of training data using mini-batch gradient descent.
- We find that high data reuse is detrimental in multi-agent settings;
  - *we recommend using 15 training epochs for easy tasks, and 10 or 5 epochs for more difficult tasks.*
- We hypothesize that the number of training epochs can control the challenge of non-stationarity in MARL.
  - *Non-stationarity arises from the fact that all agents’ policies are changing simultaneously throughout training.*
  - *Using more training epochs will cause larger changes to the agents’ policies, which exacerbates the non-stationarity challenge.*
- We additionally avoid splitting a batch of data into mini-batches, as this results in the best performance.

Factors Influential to PPO’s Performance

Policy/Value Clipping

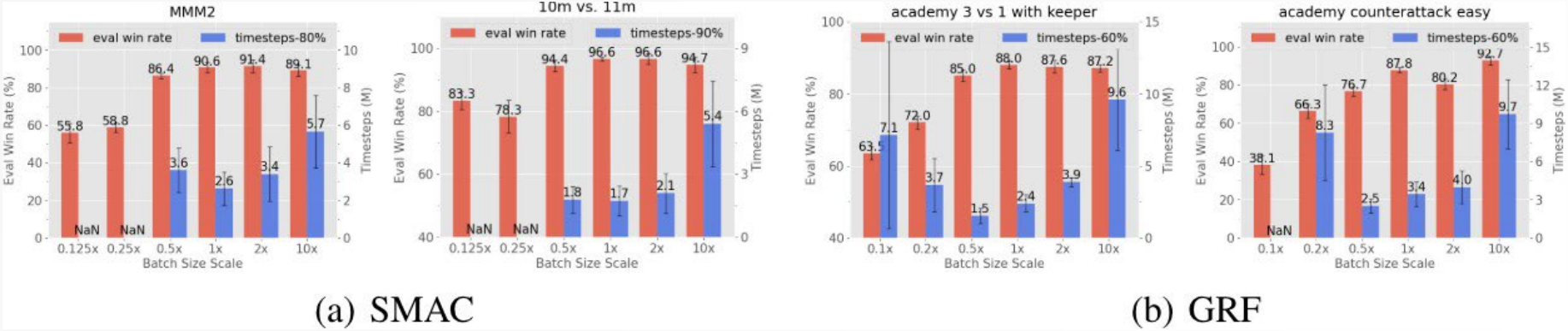
- A core feature of PPO is the use of clipping in the policy and value function losses;
  - *this is used to constrain the policy and value functions from drastically changing between iterations in order to stabilize the training process*
  - *The strength of the clipping is controlled by the ϵ hyperparameter: large ϵ allows for larger changes to the policy and value function.*
- Similar to mini-batching, clipping may control the non-stationarity problem, as smaller ϵ values encourage agents’ policies to change less per gradient update.
- We generally observe that smaller ϵ values correspond to more stable training, whereas larger ϵ values result in more volatility in MAPPO’s performance.



Factors Influential to PPO’s Performance

Batch Size

- Effect of batch size on MAPPO’s performance in SMAC and GRF.
  - *Red bars show the final win-rates. The blue bars show the number of environment steps required to achieve a strong win-rate (80% or 90% in SMAC and 60% in GRF) as a measure of sample efficiency.*
  - *“NaN” means such a win-rate was never reached. The x-axis specifies the batch-size as a multiple of the batchsize used in our main results.*
- A sufficiently large batch-size is required to achieve the best final performance/sample efficiency; further increasing the batch size may hurt sample efficiency.



(a) SMAC (b) GRF

总结

- 多智能体强化学习
  - 博弈论+强化学习
  - 均衡问题/非稳态问题/部分可观测/信誉分配
- 博弈理论
  - 标准形式博弈/马尔可夫博弈/部分可观测马尔可夫博弈
  - 纳什均衡/近似纳什均衡/相关均衡
- 博弈强化学习
  - 价值优化
  - 策略演化
- 协作多智能体强化学习
  - 策略梯度/值分解/智能体通信