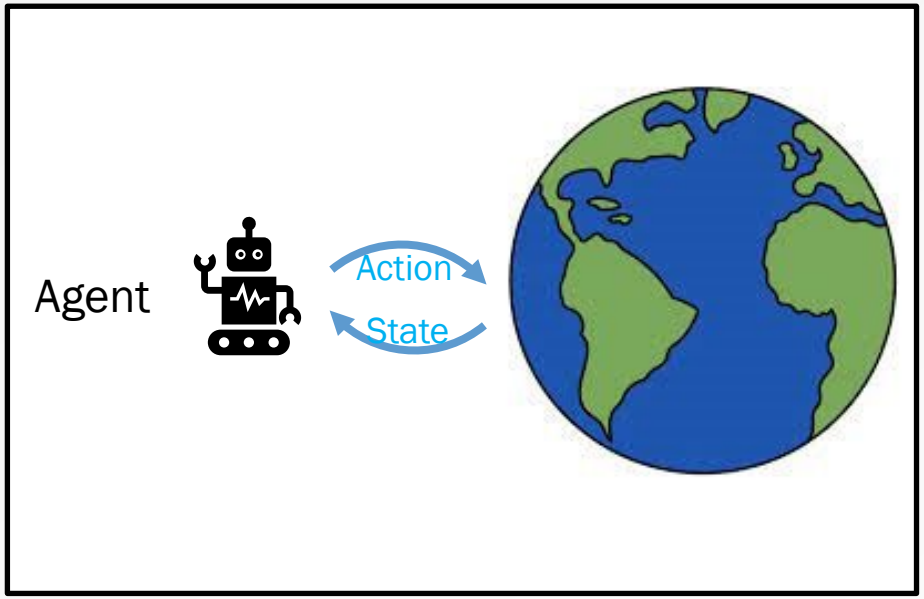
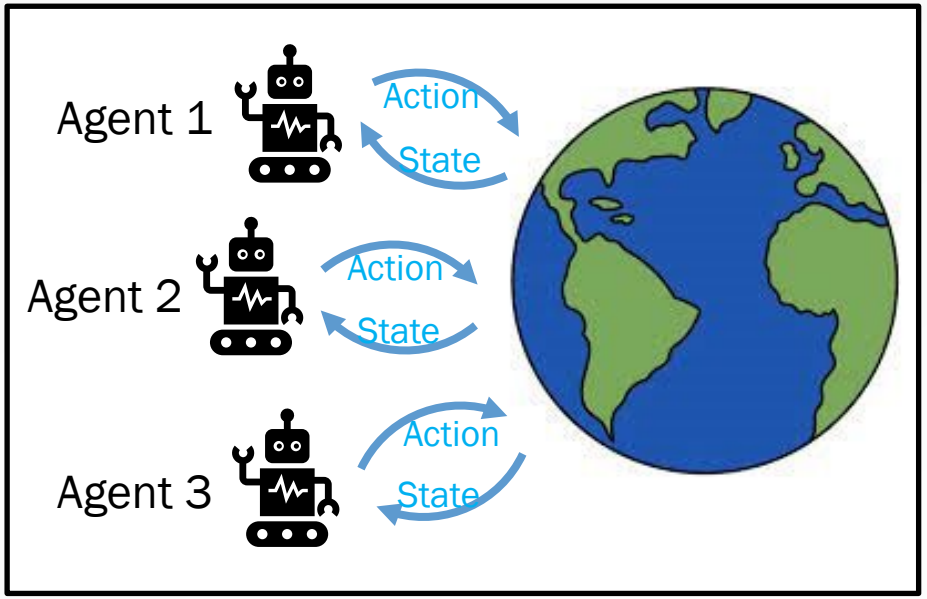
多智能体强化学习

朱圆恒
中国科学院自动化研究所

单智能体 vs 多智能体



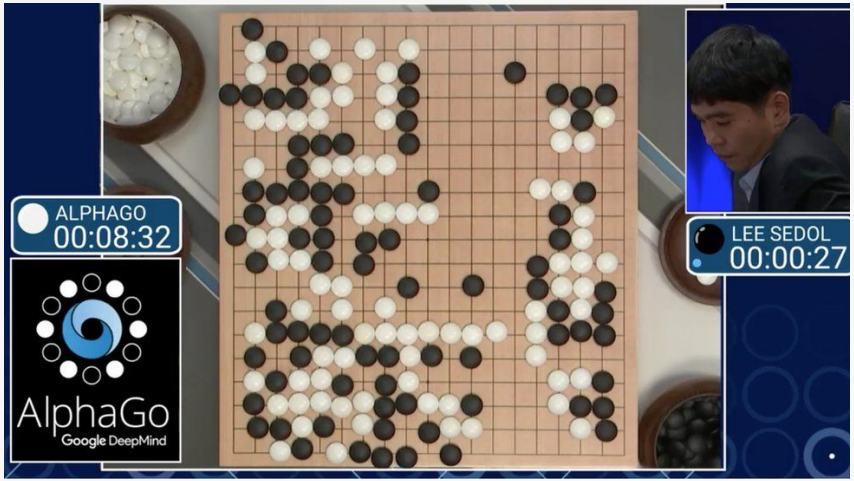
单智能体



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[1,3]*, Igor Babuschkin[1,3], Wojciech M. Czarnecki[1,3], Michaël Mathieu[1,3], Andrew Dudzik[1,3], Junyoung Chung[1,3], David H. Choi[1,3], Richard Powell[1,3], Timo Ewalds[1,3], Petko Georgiev[1,3], Junhyuk Oh[1,3], Dan Horgan[1,3], Manuel Kroiss[1,3], Ivo Danihelka[1,3], Aja Huang[1,3], Laurent Sifre[1,3], Trevor Cai[1,3], John P. Agapiou[1,3], 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

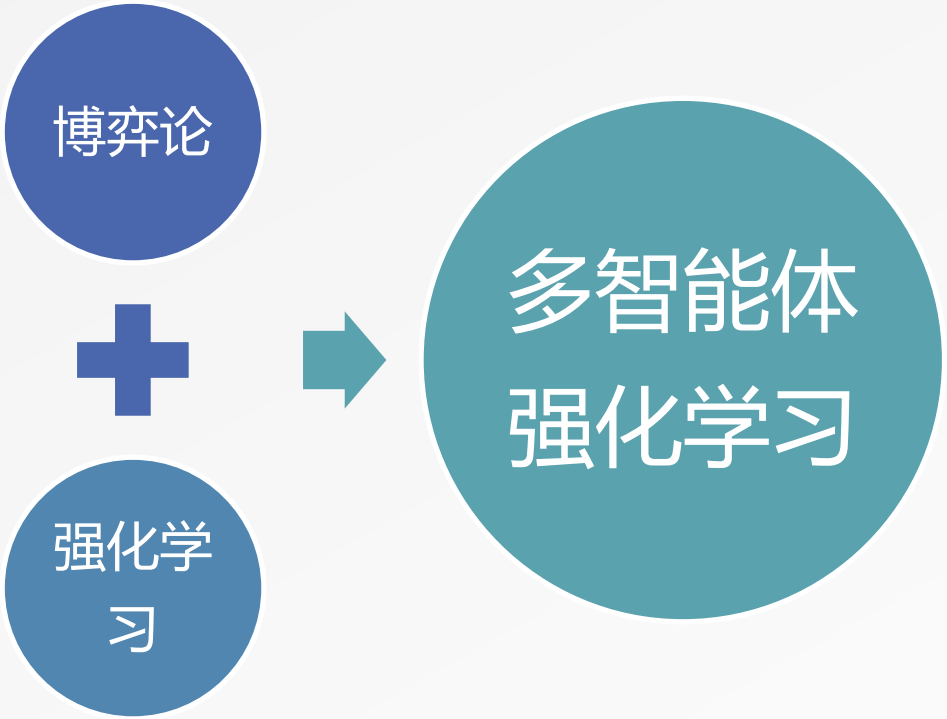
- **多智能体系统**：在同一个环境中存在多个智能体相互交互的系统

交通

股票市场

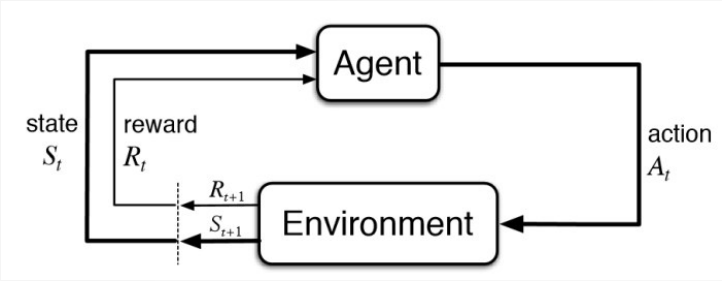
生产线

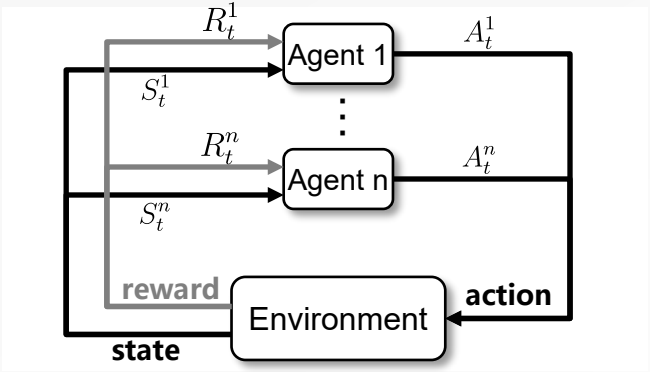
多智能体强化学习(1)



- 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

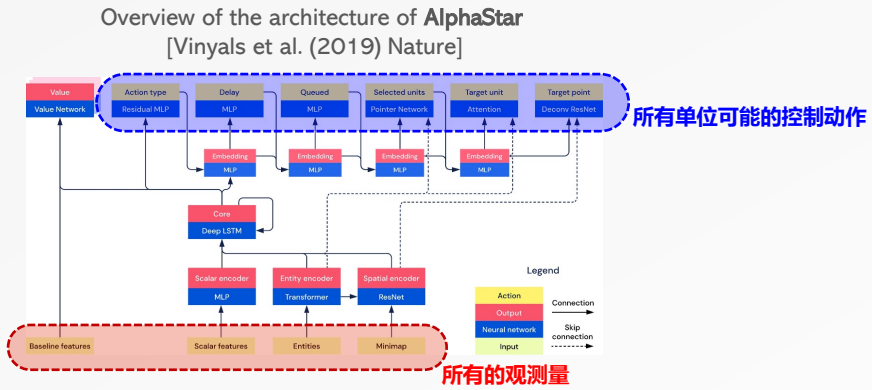
- 状态 $S_t, S_{t+1}, \ldots$
- 动作 $\{A_t^i\}, \{A_{t+1}^i\}, \ldots$
- 奖励 $\{R_t^i\}, \{R_{t+1}^i\}, \ldots$
- 转移 $S_{t+1} \sim P\left(S_t, \{A_t^i\}\right)$

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



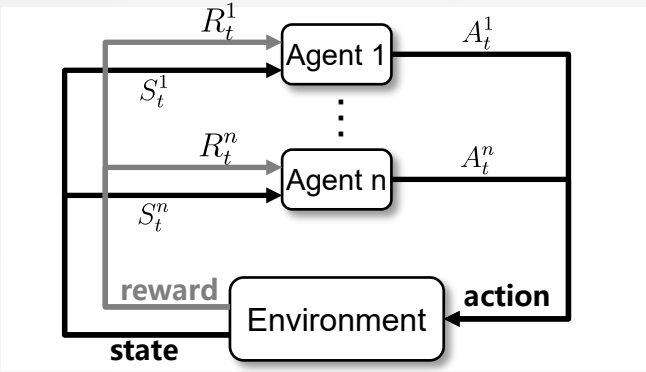
MARL分类(2)

- 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

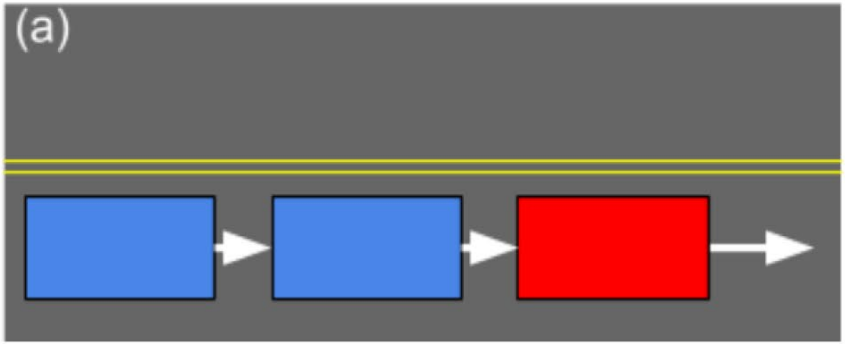
- 状态 $S_t, S_{t+1}, \dots$
- 动作 $\{A_t^i\}, \{A_{t+1}^i\}, \dots$
- 奖励 $\{R_t^i\}, \{R_{t+1}^i\}, \dots$
- 转移 $S_{t+1} \sim P\left(S_t, \{A_t^i\}\right)$

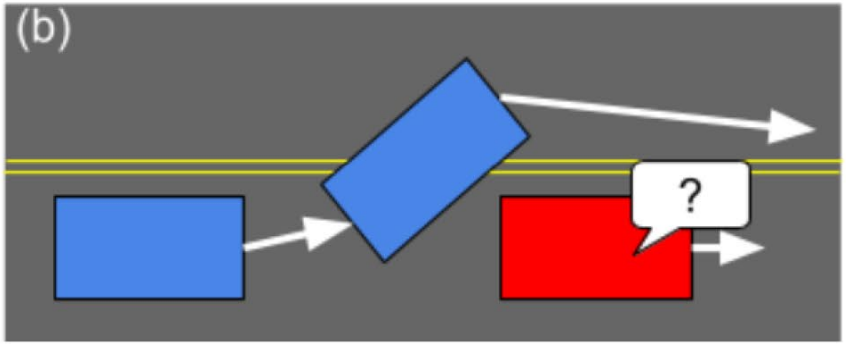
$\max R_t^i + \gamma R_{t+1}^i + \gamma^2 R_{t+2}^i + \cdots$

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

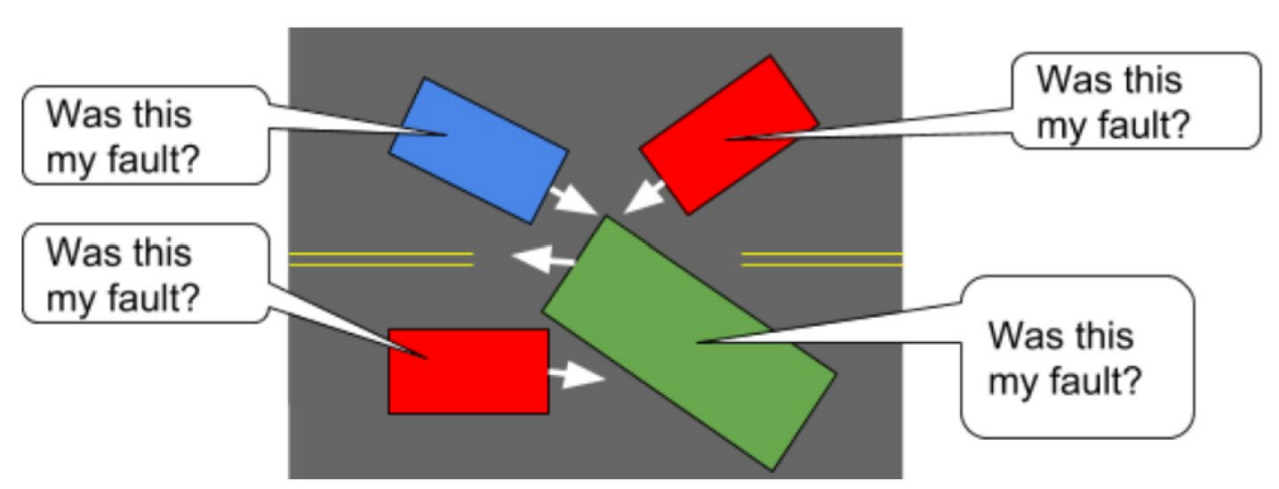
挑战：非静态环境





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。

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

挑战总结

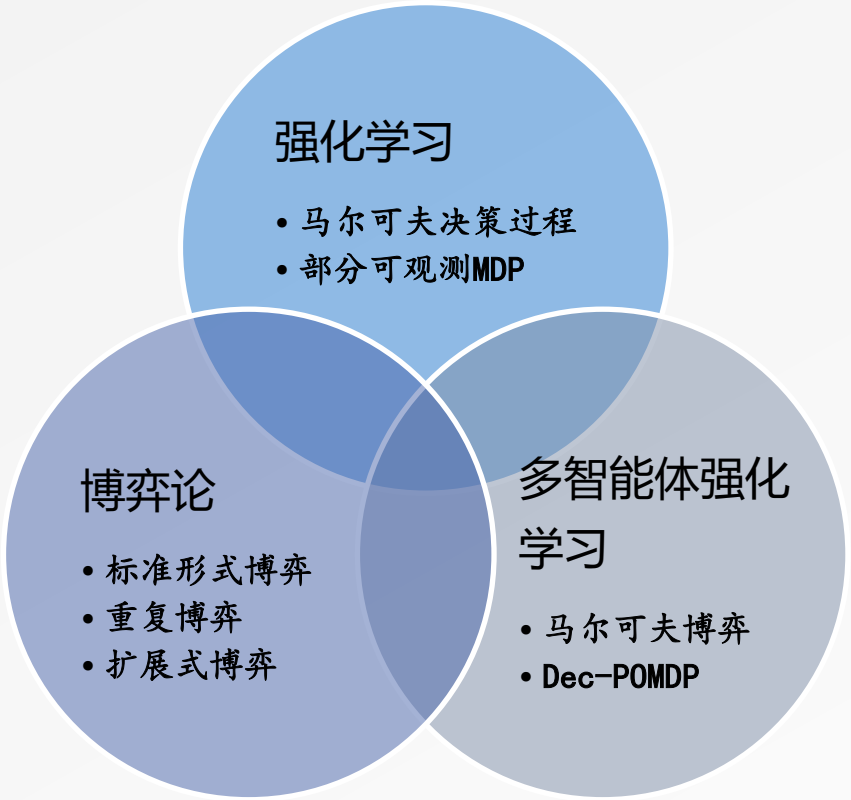
- 单智能体RL，智能体只需要根据自身的动作和对环境的影响，调整自己的策略。在多智能体MARL中，智能体要适应其它智能体的动作和学习；智能体在同一状态执行同一动作会获得不同的奖励
- MARL中智能体有可能无法获得环境全局信息（分布式性），即使是能获取全局信息，也无法预测其它智能体的行为以及环境的变化
- 信誉分配问题：难以确定是哪个智能体要对任务成功或失败负责。在智能体间如何分配奖励，以及局部奖励和全局奖励如何结合，能够加快学习速率或是收敛到全局最优解，是MARL难题

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}

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

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

- 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

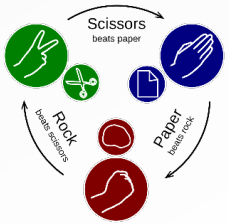
- 确定性纳什均衡策略：纯策略
  - 囚徒困境
- 随机性纳什均衡策略：混合策略
  - 石头-剪子-布 (1/3, 1/3, 1/3)
- 本课我们在随机策略空间寻找纳什均衡

ROW \ COL	Co-operate	Defect
Co-operate	(3, 3)	(0, 5)
Defect	(5, 0)	(1, 1)

Preference to Move Based on Higher Payoff

Nash Equilibrium

	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

- 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 $\pi_i \in \epsilon\text{-}BR(\pi_{-i})$

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 $\pi(a) - \pi(b) \geq V \quad (\text{vs. A})$

列玩家动作B下的期望收益 $-\pi(a) + \pi(b) \geq V \quad (\text{vs. B})$

$$\max V$$

$$\pi(a) + \pi(b) = 1$$

$$0 \leq \pi(a), \pi(b) \leq 1$$

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

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

- 价值函数 (期望回报)

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

马尔可夫博弈
Markov Game

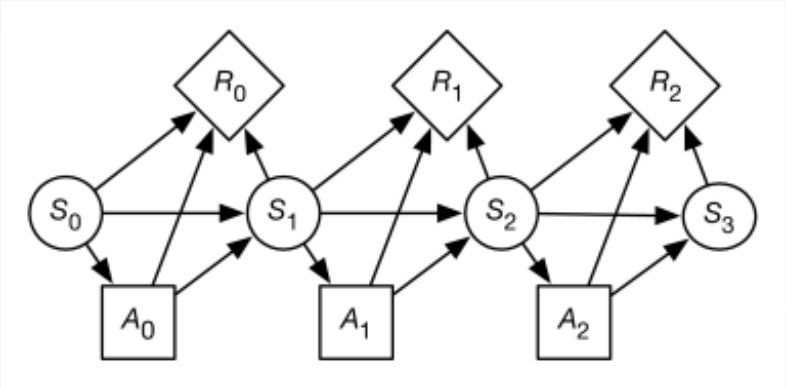
- 纳什均衡：每个玩家的策略都是其它玩家策略的最佳响应

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

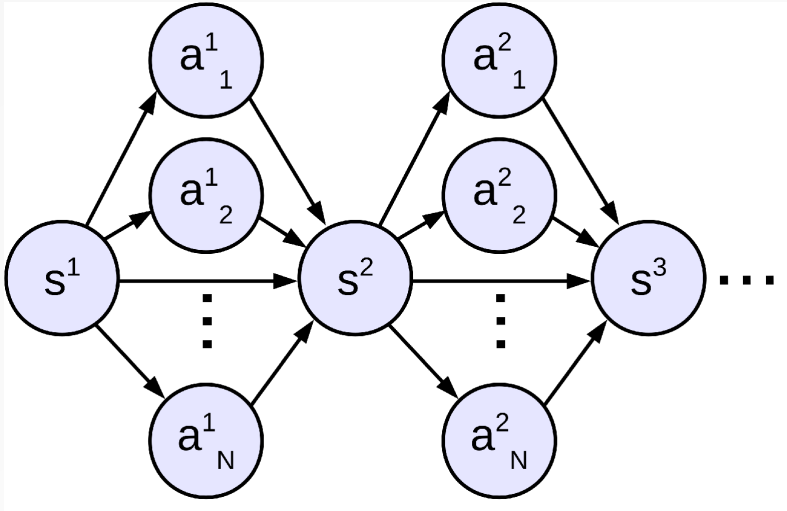
  - 任何一个玩家都不会主动改变自己的策略
  - 最佳响应

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

MDP vs MG

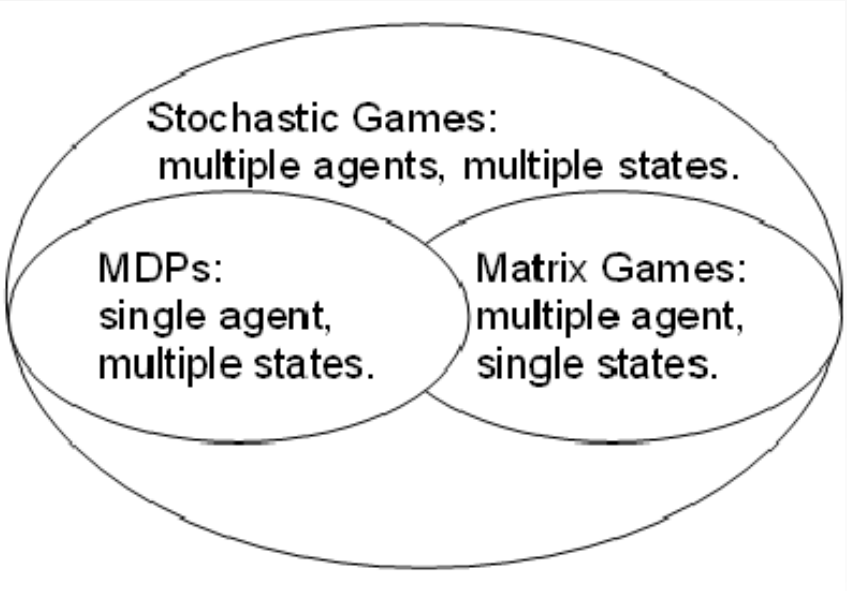


马尔可夫决策过程



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）

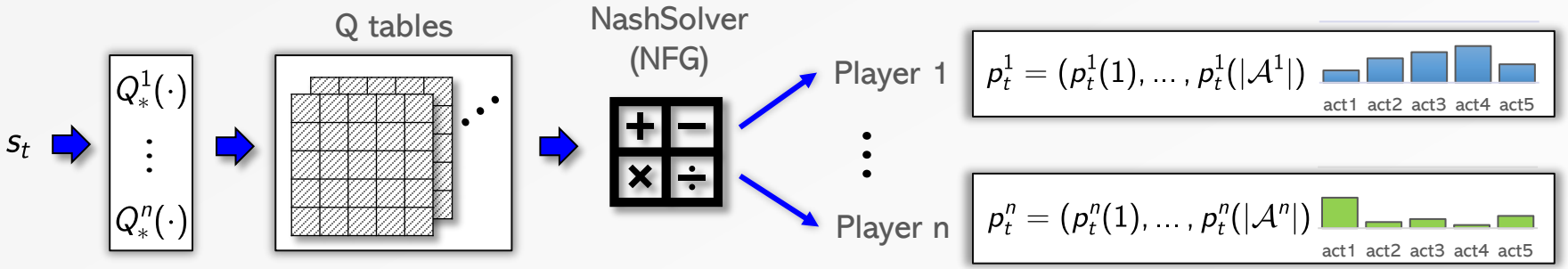
$$Q_*^i(s, a_1, \ldots, a_n) = \mathbb{E}\left[\mathcal{R}^i(s, a_1, \ldots, a_n) + \gamma \sum_{s'} \mathcal{P}(s'|s, a_1, \ldots, a_n) V_*^i(s'|\pi)\right]$$

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

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

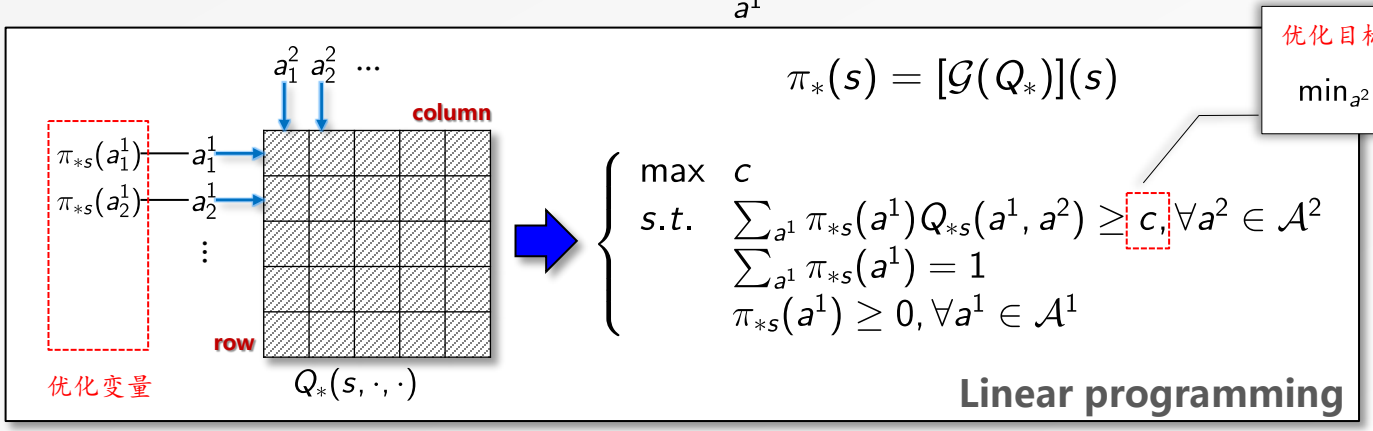
According to Sharpley theory [Shapley (1953) PNAS], there always exists a **unique Nash value** to two-player zero-sum game

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

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 $\{Q_1, Q_2, \ldots\}$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）

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

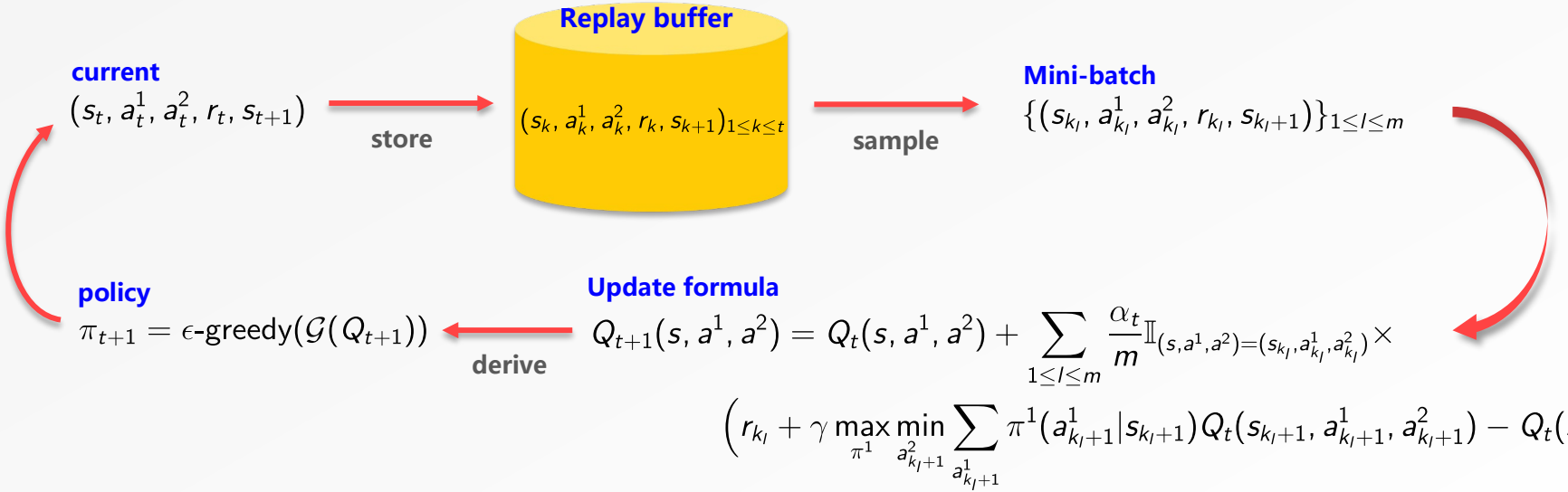
Minimax-Q [Littman (1994) MLP]

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

- 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2pOsMG，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

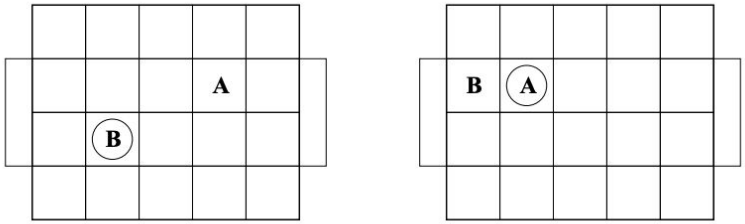
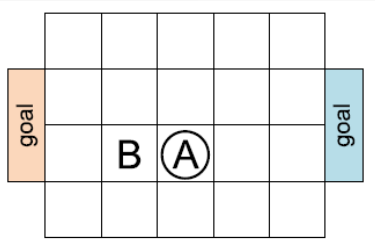
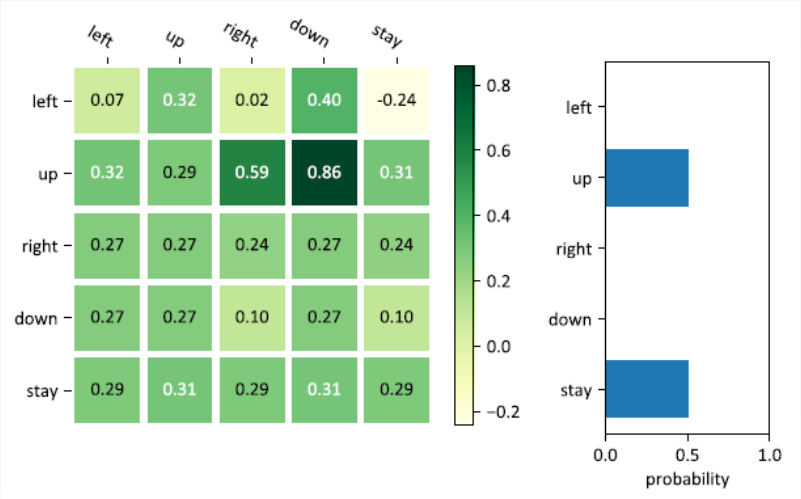


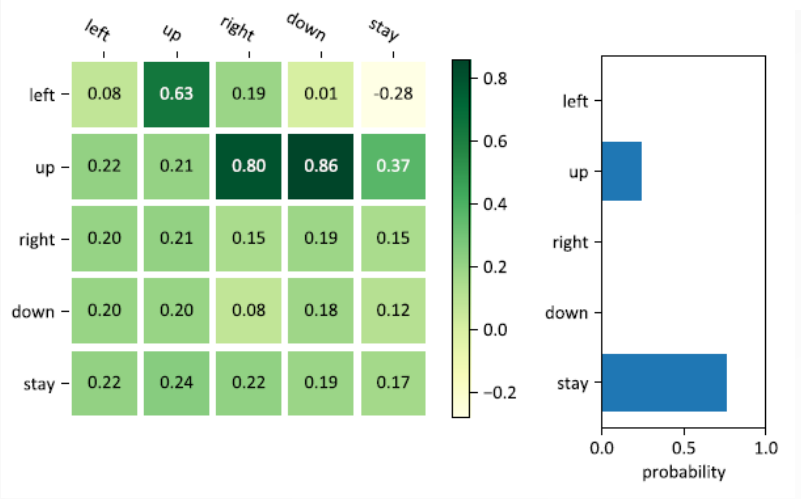
Figure 2: An initial board (left) and a situation requiring a probabilistic choice for A (right).

足球博弈
Soccer Game



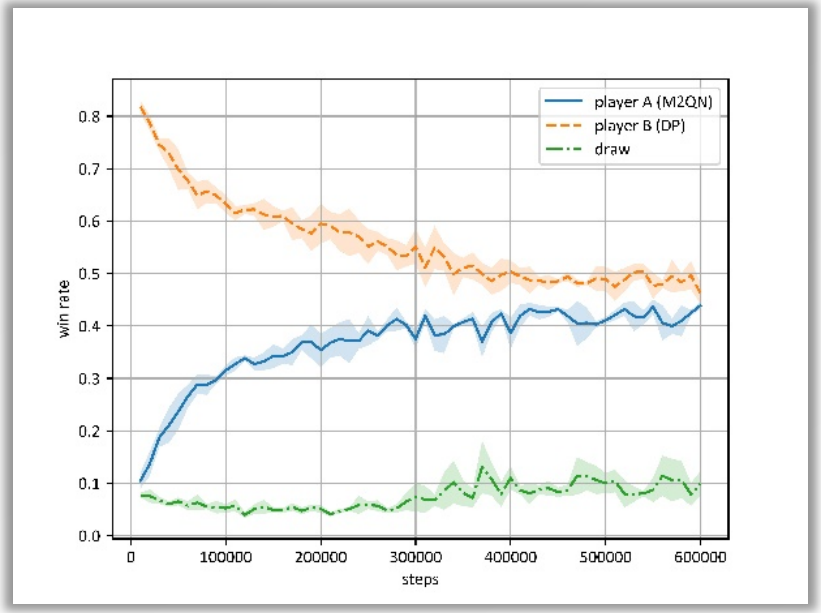


价值迭代收敛的纳什Q函数在状态点上的Q值以及玩家A的动作概率分布



Minimax-Q学习的Q函数在状态点上的Q值以及玩家A的动作概率分布

足球博弈
Soccer Game



- Minimax-Q在线学习过程中，学到的玩家A策略与纳什均衡玩家B策略(价值迭代得到)的对抗测试曲线
- 随着Minimax-Q算法运行，玩家A策略的对抗性能不断提升，逐渐达到纳什均衡策略水平

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

NashConv

- 两人零和的NashConv

$$\text{NashConv}(\pi, \mu) = \max u^1(\cdot, \mu) - \min u^1(\pi, \cdot)$$

  - 在玩家1的收益函数上，玩家1取最大化与玩家2取最小化之间的差值
- 两人零和+对称博弈：玩家1和玩家2的策略是对称互换的

$$\text{Exploitability}(\pi) = \max u^1(\cdot, \pi) - \min u^1(\pi, \cdot) = 2 \max u^1(\cdot, \pi)$$

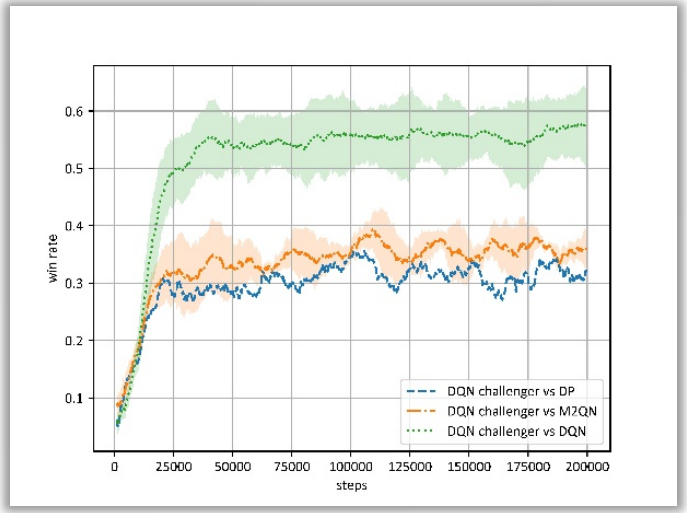
  - 玩家策略 π 会被对手利用的程度
  - 纳什均衡策略利用度是0 （石头-剪刀-布的纳什均衡策略（1/3，1/3，1/3））
  - 利用度越高，越会被对手针对

足球博弈
Soccer Game

- 比较价值迭代/Minimax-Q/self-play Q-learning 结果的Exploitability
  - 将上述学到的玩家策略π固定，使用Q学习求解对手的最佳响应策略 (max MDP)

$$\text{Exploitability}(\pi) = \max u^1(\cdot, \pi)$$

- 最佳响应对手的Q学习过程
  - 价值迭代/Minimax-Q学到的策略不会让最佳响应对手获得较高的胜率 （利用度底）
  - Self-play Q-learning 学到的策略会让最佳响应对手得到较高的胜率 （利用度高）



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)]

  - β 衰减因子
  - $(\pi_*^1,\ldots,\pi_*^n)$ 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)

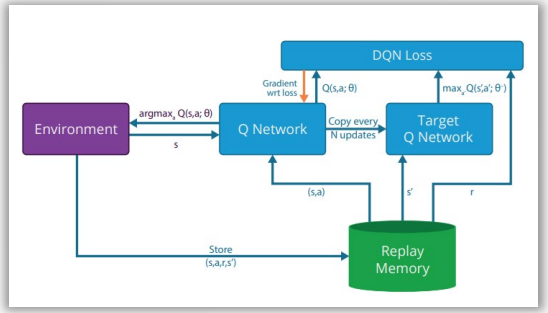
Let $t := t + 1$.

- $NashQ_t^i(s') = \pi^1(s') \cdots \pi^n(s') \cdot Q_t^i(s')$ ：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

Deep Q network [Mnih et al. (2015) Nature]



Double DQN [van Hasselt et al. (2016) AAAI]

$$Y_t^{\text{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)$$

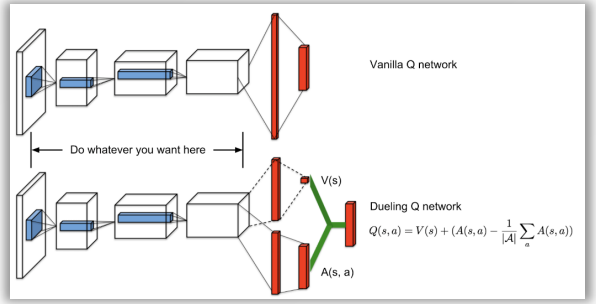
Target network

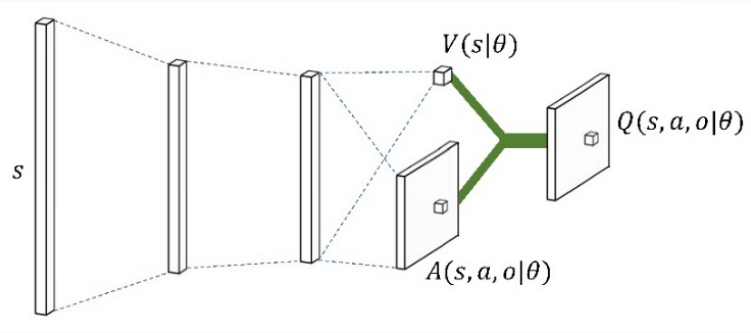
Avoid overestimation

Current network

$$Y_t^{\text{DoubleDQN}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-)$$

Dueling network [Wang et al. (2016) ICML]





两人零和Q函数网络

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 $Q(s, a^1, a^2, \ldots, a^n)$，复杂度随动作数量和玩家数量的增加指数型增长（维数灾）

  - 是否可以直接对玩家策略优化/学习，获得收敛的纳什均衡策略？

基于策略优化的强化学习

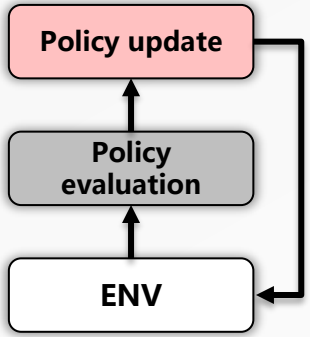
策略强化学习

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

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

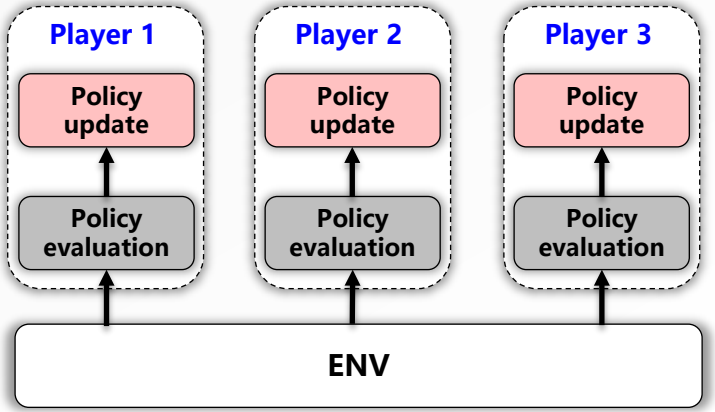


- 多智能体/多玩家：independent learning + policy RL

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

⋮

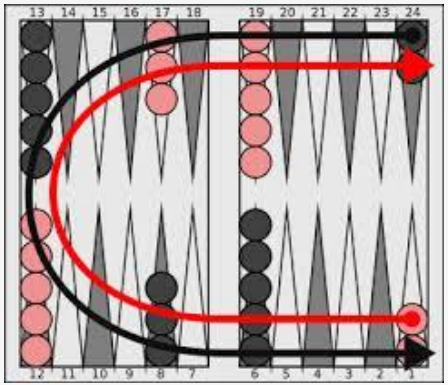


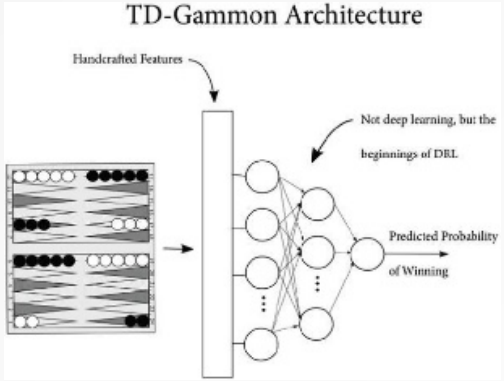
- 独立学习下由于其它玩家的策略变化，导致环境是动态变化，policy update不稳定，所有玩家容易出现策略震荡

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]



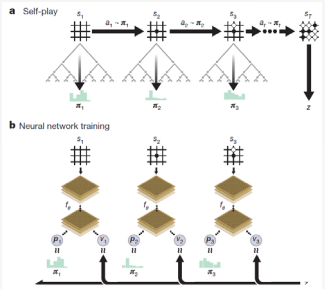


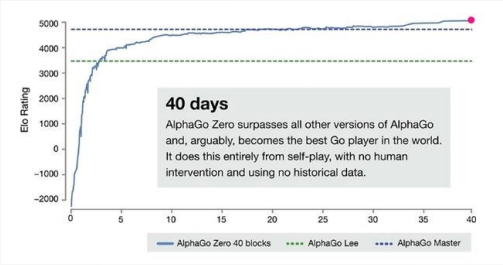
TD-Gammon: neural network that can learn to play backgammon solely by playing against itself and learning from the results.

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

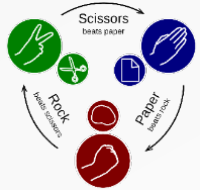
AlphaGo Zero [David Silver, et al, Nature, 2017]





"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 。。。

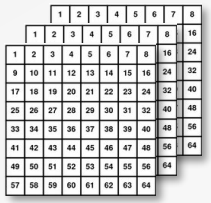
AlphaGo Zero解决办法：保存历史最强的策略作为对手，只有当前策略提升到以55%胜率打败最强对手时，才替换成最强策略

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



Payoff matrix

round	1	2	3	...	k
Player 1	a_1^1	a_2^1	a_3^1	...	a_k^1
Player 2	a_1^2	a_2^2	a_3^2	...	a_k^2

Empirical frequency of actions

$$q_k^1(a) = \frac{1}{k}\sum_k \mathcal{I}(a = a_k^1)$$
$$q_k^2(a) = \frac{1}{k}\sum_k \mathcal{I}(a = a_k^2)$$

Best-response strategy

$$p_{k+1}^1 = BR(q_k^2)$$
$$p_{k+1}^2 = BR(q_k^1)$$

(k+1)-th round action

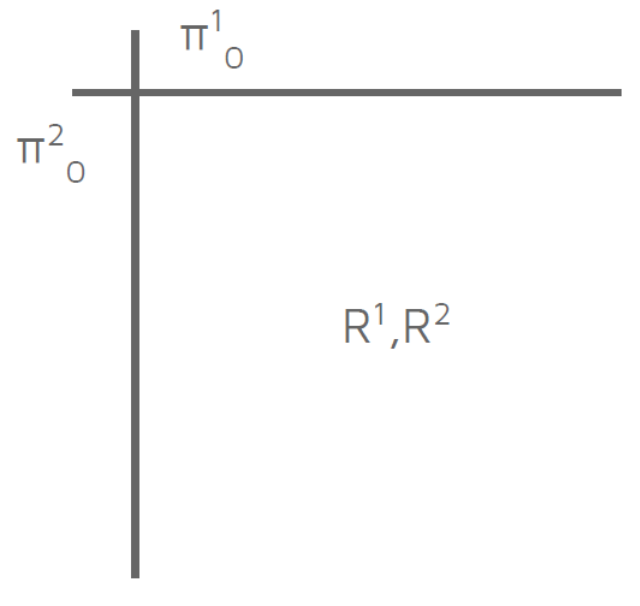
$$a_{k+1}^1 \sim p_{k+1}^1$$
$$a_{k+1}^2 \sim p_{k+1}^2$$

- 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

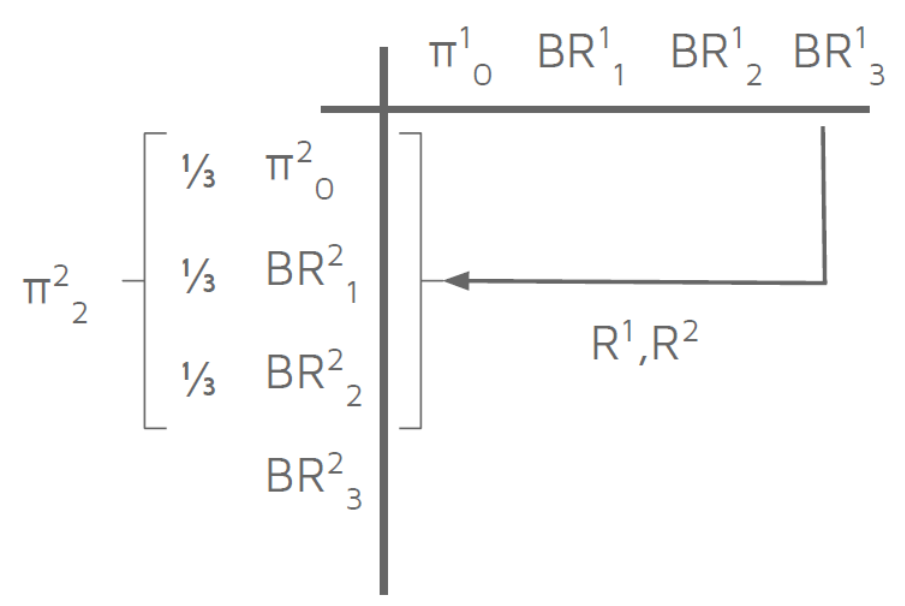
- 标准形式+重复博弈 （RPS博弈）



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

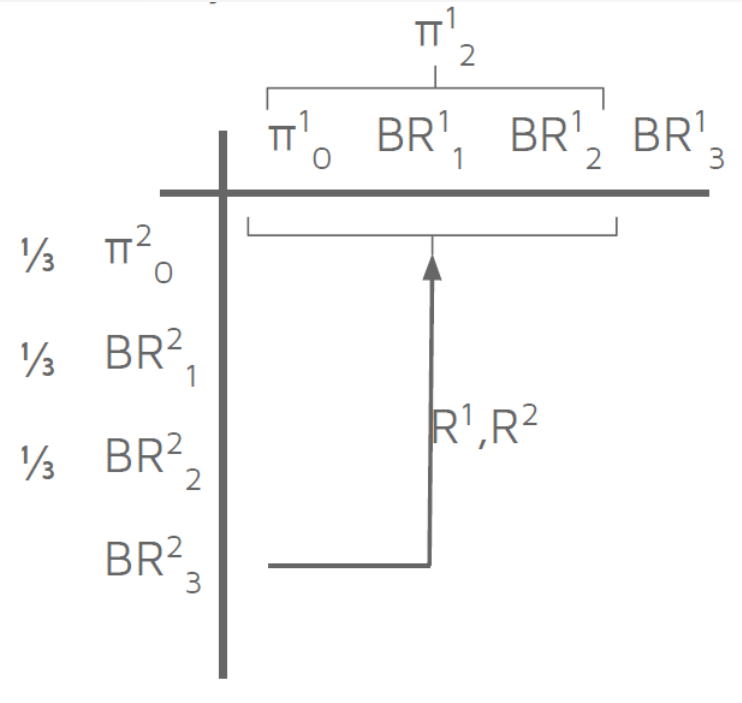
- 标准形式+重复博弈 （RPS博弈）



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）



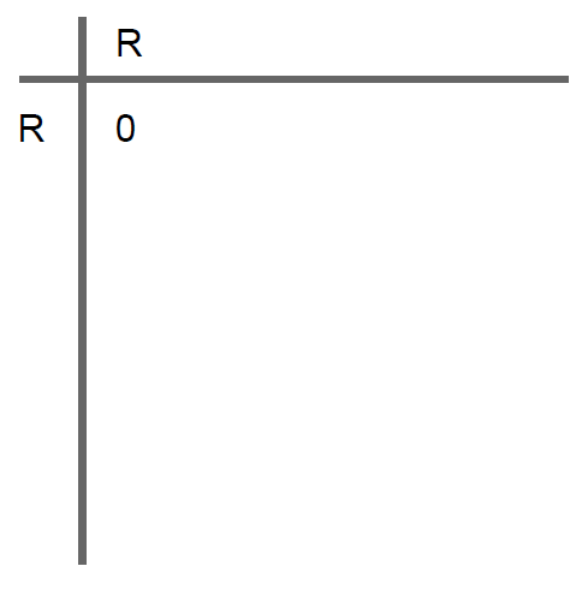
– 从玩家任意策略出发 $(\boldsymbol{\pi_0^1}, \boldsymbol{\pi_0^2})$

– 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）



- 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）

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

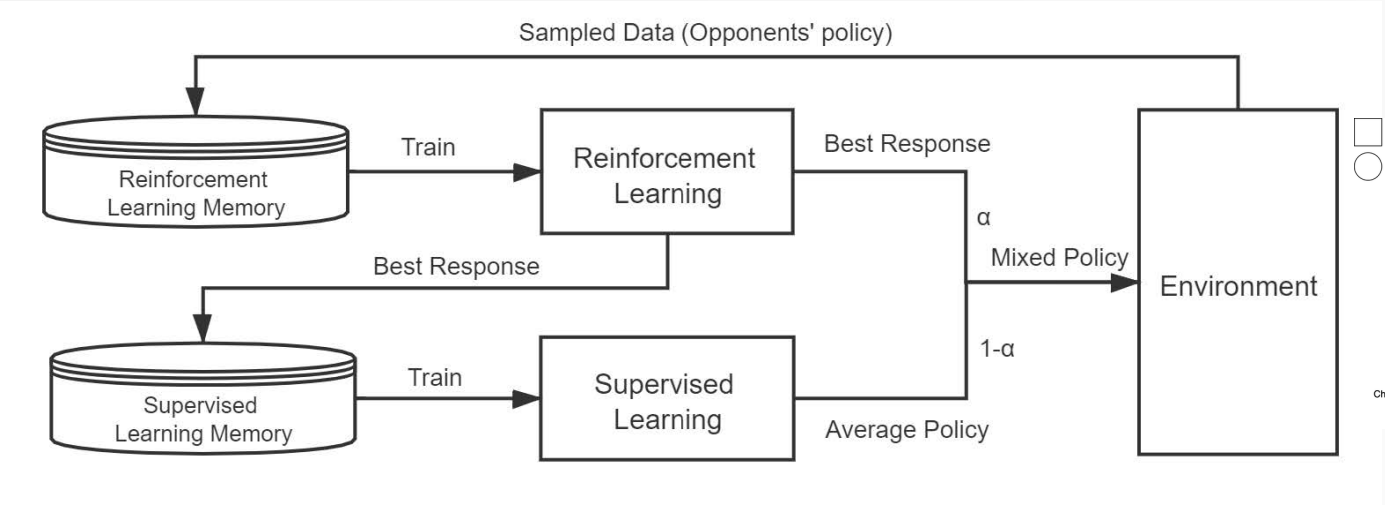
    - 根据玩家历史行为(s_t, a_t)监督训练一个平均策略网络（average policy net）
  - 下一轮博弈的策略采用针对其它玩家经验行为的最佳响应

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

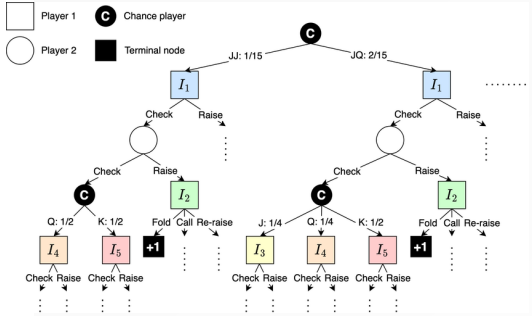
    - 强化学习最佳响应 $\max \sum \gamma^t r_t^i$， against $\tilde{\pi}_k^{-i}$

神经网络虚拟自我博弈

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



Leduc Hold'em

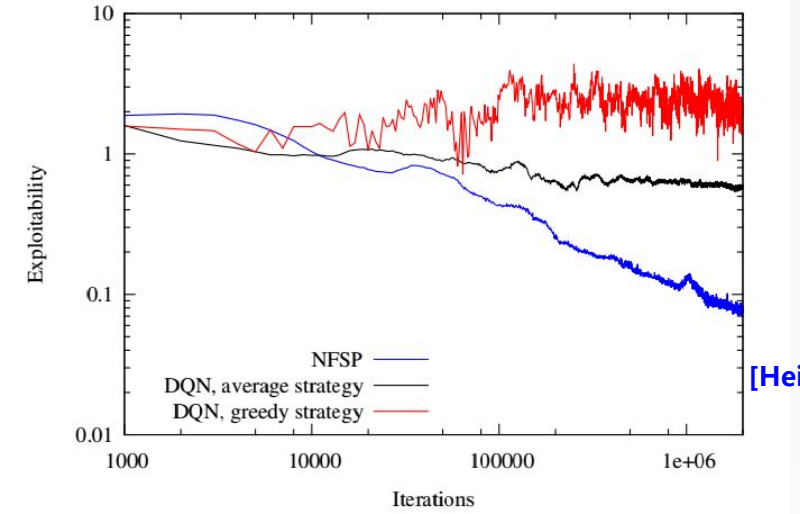


- 每个玩家以一定概率执行best response或average policy
- 所有数据都进入RL Memory用于DQN训练BR
- BR策略执行的动作进入SL Memory训练average policy

神经网络虚拟自我博弈

Neural Fictitious Self Play

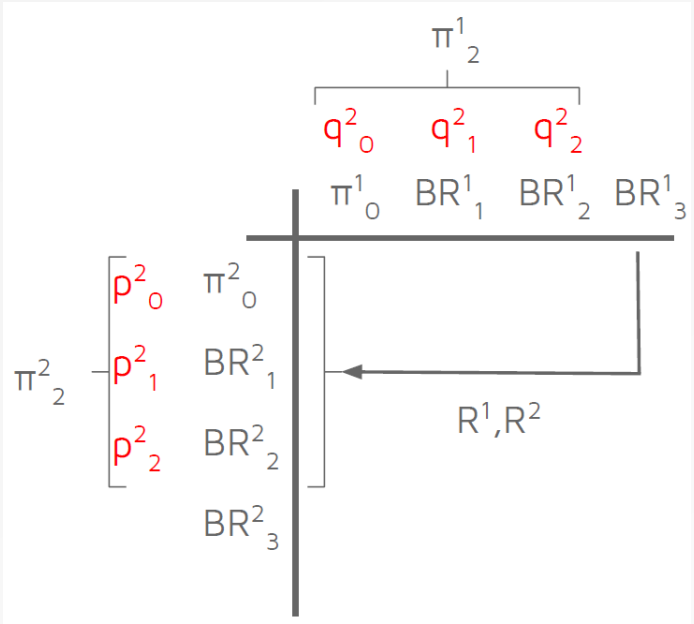
- Leduc Hold’ em poker experiments:



[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

	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)

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)

Double Oracle

- 收敛次数更快

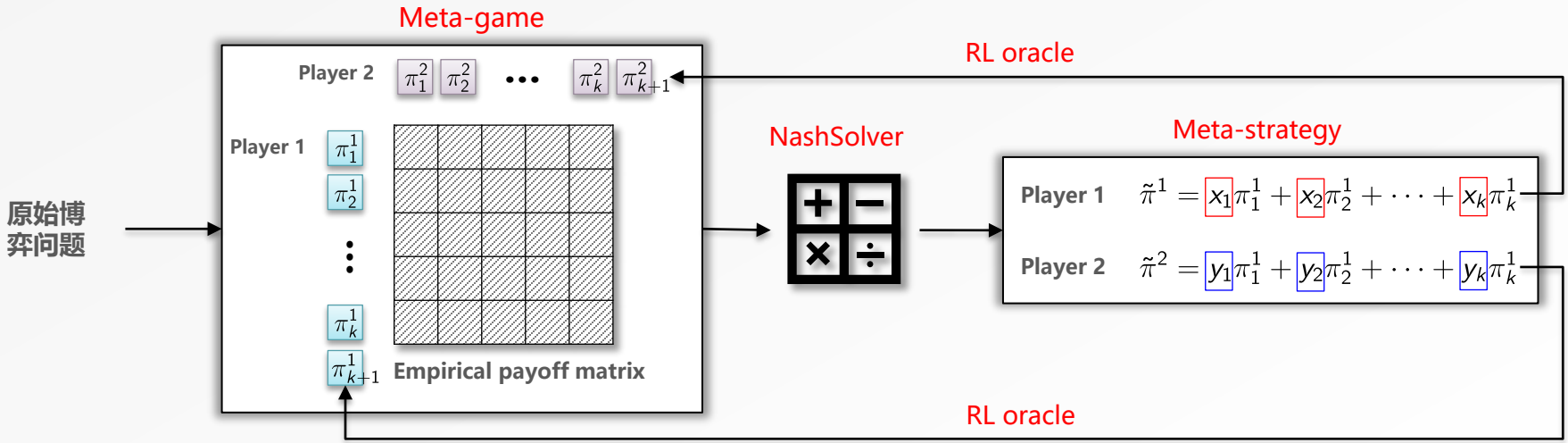
	R	P	S
R	0	1	-1
P	-1	0	1
S	1	-1	0

- 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



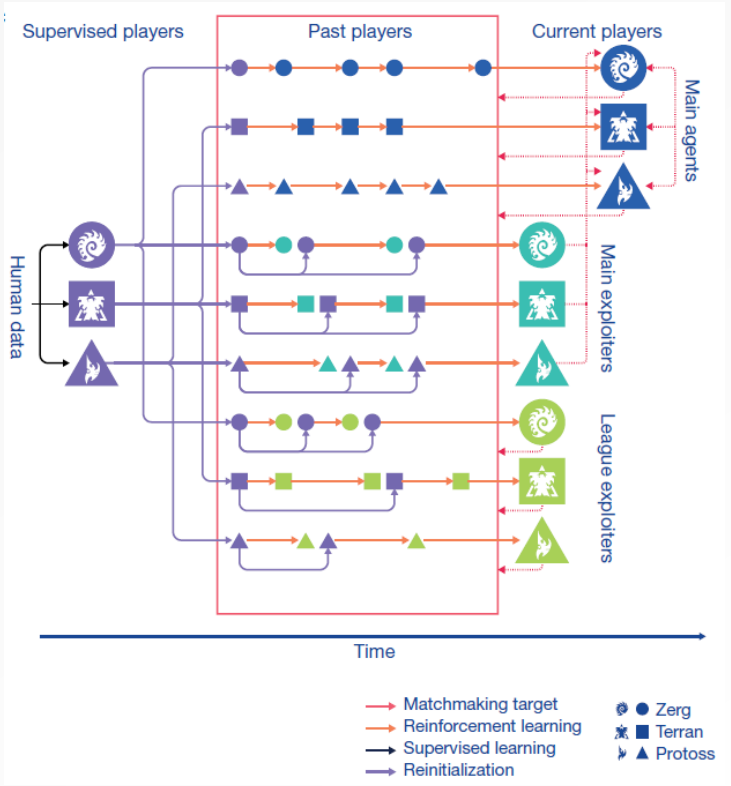
PSRO

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

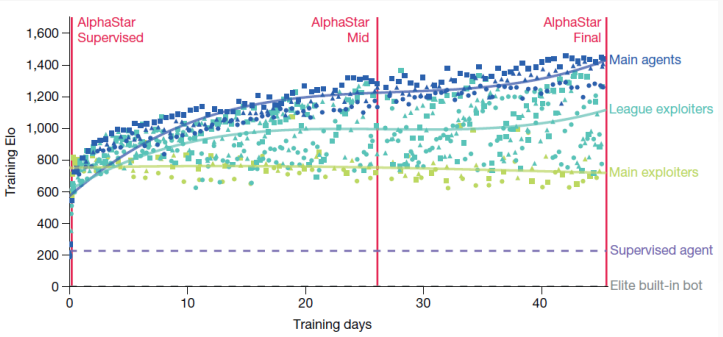
- PSRO的思路在于构建策略池 policy pool，从中选取适合于作对手的策略
- 将新学习的策略加入到策略池，不断补充和完善策略集合
- AlphaStar policy pool (league) 包含三种 past agents:
  - **Main agents** 主策略
  - **Main exploiters** 发现当前Main agents的缺陷
  - **League exploiters** 发现 league 中的缺陷



[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)

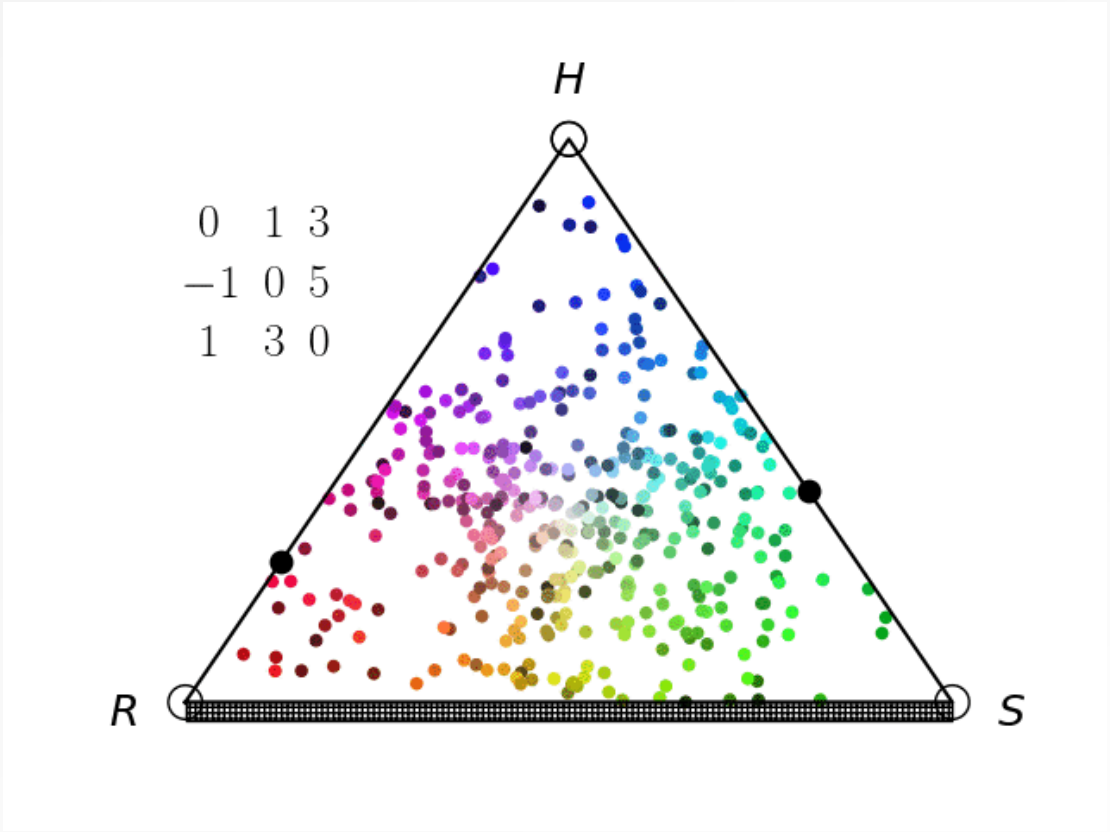
$$\frac{f(\mathbb{P}[A \text{ beats } B])}{\sum_{C \in \mathcal{C}} f(\mathbb{P}[A \text{ beats } C])}$$

  - $f_{\text{hard}}(x) = (1-x)^p$ 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

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

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

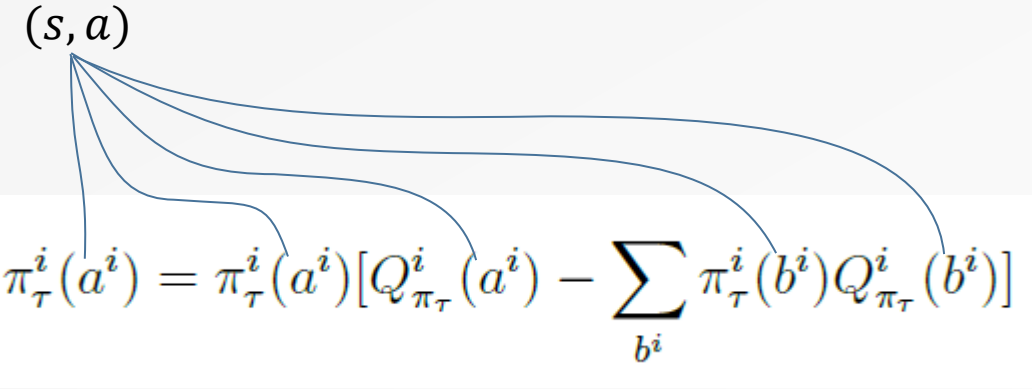
Current frequency of strategy
（多少a的个体可以进入下一代）

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

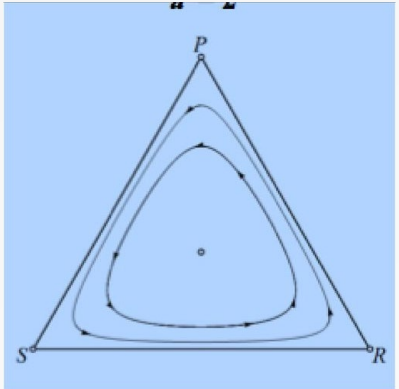
- 马尔可夫/序贯博弈问题



  - 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

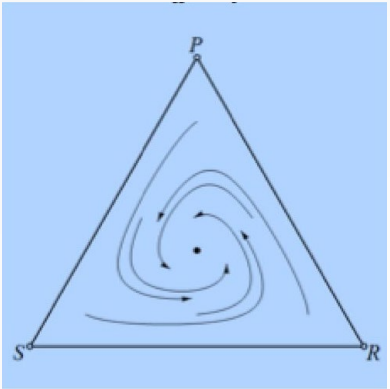
Replicator Dynamics

	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

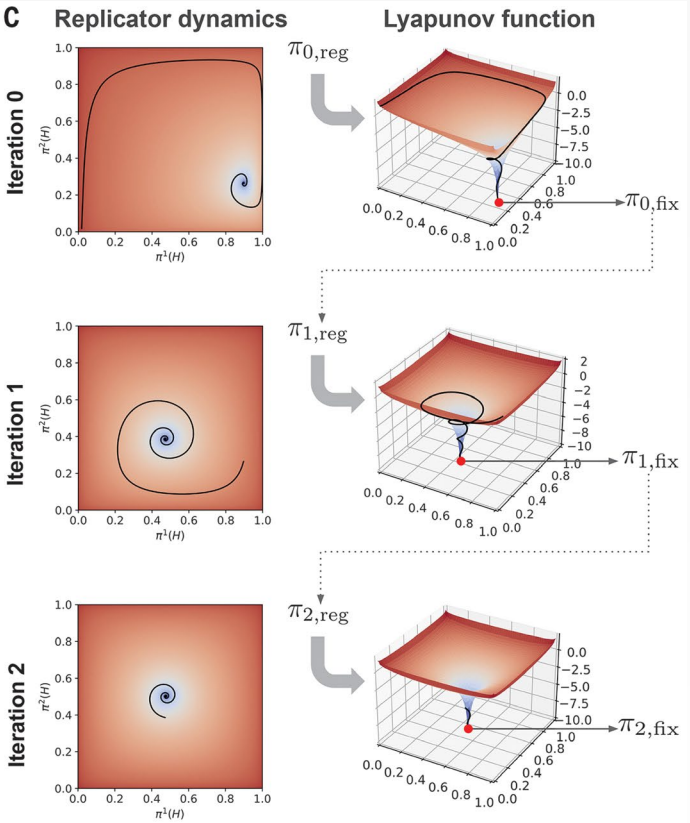
R-NaD Iteration

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:
$$\pi_{m+1,\text{reg}} = \pi_{m,\text{fix}}$$

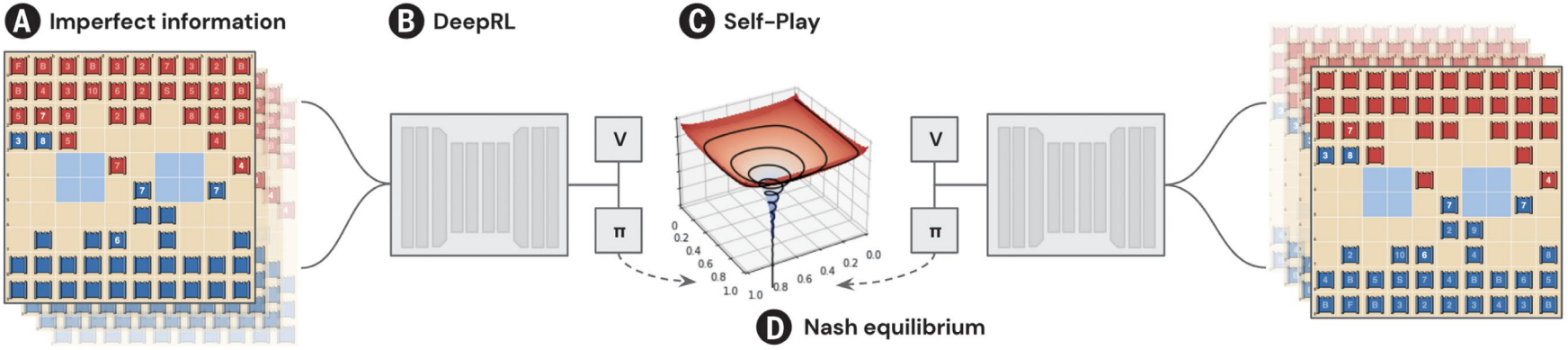
Repeat stages until convergence

C



[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.

Independent Q Learning

- 单智能体中的Q学习：根据观测量 (s_t, a_t, r_t, s_{t+1})，学习Q函数，然后选择ϵ-greedy动作

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_t \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q_t(s_t, a_t) \right]$$

$$a_{t+1} = \epsilon\text{-greedy}(Q_{t+1}(s_{t+1}, \cdot))$$

[Chris Watkins, Q-learning, 1989]

- 多智能体Independent Q Learning：玩家根据个体的观测量 $(a_t^i, r_t^i)|(a_t^{-i})$，学习Q函数，然后选择正则化的greedy动作

$$Q_{n+1}^i(a^i) = Q_n^i(a^i) + \lambda_{n+1} \mathbb{I}_{\{a_n^i = a^i\}} \frac{R_n^i - Q_n^i(a_n^i)}{\beta^i(Q_n^i)(a_n^i)}, \quad \text{for each } a^i \in A^i,$$

修正动作选择概率

$$\beta^i(Q^i) = \underset{\pi^i \in \Delta^i}{\operatorname{argmax}} \left\{ \sum_{a^i \in A^i} \pi^i(a^i) Q^i(a^i) + \tau v^i(\pi^i) \right\}.$$

[Leslie & Collins, Individual Q-learning, 2005]

策略　　系数　　正则项

Independent Q Learning

- 在学习率满足 $\sum_{n\geq 1}\lambda_n=\infty,\qquad \sum_{n\geq 1}(\lambda_n)^2<\infty.$ 下，IQL过程描述成连续时间常微分方程

$$\frac{\mathrm{d}}{\mathrm{d}t}q_t^i(a^i)=r^i(a^i,\beta^{-i}(q_t))-q_t^i(a^i)$$

  - 每一时刻（代），玩家i的动作 a^i 相对于当前策略 β^i 的生存优势
    （其它玩家策略为 β^{-i}）

Independent Q Learning

- 增加的正则项可以使策略光滑连续
  - e.g. negative Gibbs entropy

$$v^i(\pi^i) = -\sum_{a^i \in A^i} \pi^i(a^i) \log \pi^i(a^i)$$

$$\beta^i(Q^i) = \underset{\pi^i \in \Delta^i}{\operatorname{argmax}} \left\{ \sum_{a^i \in A^i} \pi^i(a^i) Q^i(a^i) + \tau v^i(\pi^i) \right\}$$

$$\beta^i(Q^i)(a^i) = \frac{e^{Q^i(a^i)/\tau}}{\sum_{b^i \in A^i} e^{Q^i(b^i)/\tau}}$$

  - Greedy策略（确定/不唯一/非光滑）--> softmax策略（随机/唯一/光滑）

Independent Q Learning

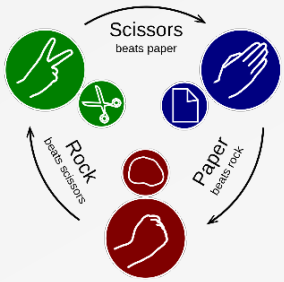
- IQL在两人零和、两人合作博弈上收敛到纳什分布 (nash distribution)
  - 两人零和 $\boldsymbol{R^1(a) = -R^2(a)}$
  - 两人合作 $\boldsymbol{R^1(a) = R^2(a)}$
  - 纳什分布：正则化的收益函数

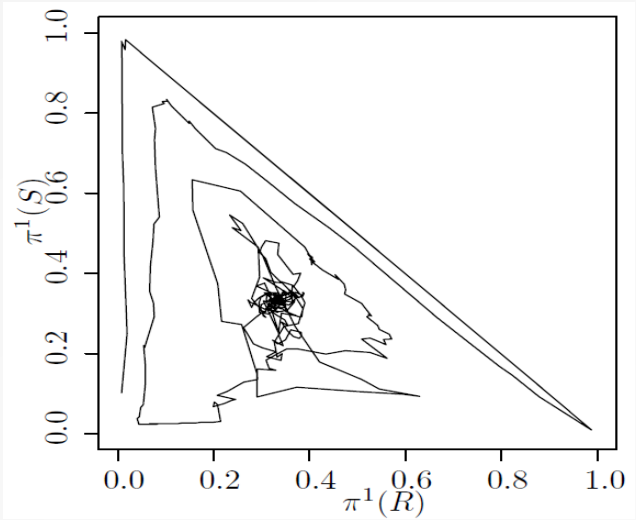
$$u^1(\pi^1, \pi^2) = \mathbb{E}_{\pi^1,\pi^2}\left[R^1(a^1, a^2)\right] + \tau v^1(\pi^1)$$

$$u^2(\pi^1, \pi^2) = \mathbb{E}_{\pi^1,\pi^2}\left[R^2(a^1, a^2)\right] + \tau v^2(\pi^2)$$

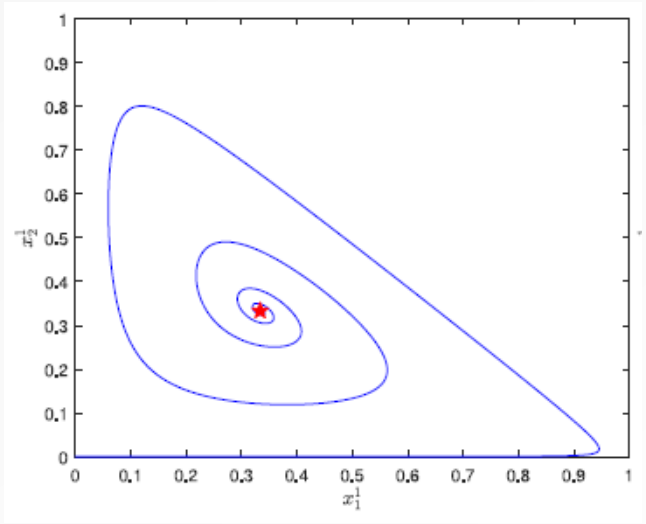
$$u^1(\pi_*^1, \pi_*^2) \geq u^1(\pi^1, \pi_*^2) \qquad u^2(\pi_*^1, \pi_*^2) \geq u^2(\pi_*^1, \pi^2)$$

  - 当正则化系统 $\tau \to 0$, Nash distribution --> Nash equilibrium





[Leslie & Collins, Individual Q-learning, 2005]



[Gao et al., IEEE TAC, 2021]

Continuous-Time Learning Dynamics

- 序贯决策中优势函数描述玩家某一动作相对当前策略下的优势

$$A(s,a) = Q(s,a) - V(s)$$

- 同时要考虑状态分布对收益的加权

$$\begin{aligned}\rho(s) &= P(s_0 = s) + \gamma P_\pi(s_1 = s) + \gamma^2 P_\pi(s_2 = s) + \dots \\ &= P(s_0 = s) + \gamma \sum_{s'} [\mathcal{P}_\pi(s|s') P(s_0 = s') + \gamma \mathcal{P}_\pi(s|s') P_\pi(s_1 = s') + \dots] \\ &= P(s_0 = s) + \gamma \sum_{s'} \mathcal{P}_\pi(s|s') \rho(s')\end{aligned}$$

$$\rho = P_0 + \gamma \mathcal{P}_\pi^T \rho \quad \Rightarrow \quad \rho = (I - \gamma \mathcal{P}_\pi^T)^{-1} P_0$$

  - P_0 初始状态分布，$\mathcal{P}_\pi(s_{t+1}|s_t) = \sum_a \pi(a|s_t) P(s_{t+1}|s_t, a)$ 策略π下的马尔可夫转移概率

Continuous-Time Learning Dynamics

- 因此马尔可夫博弈的策略演化动态（Continuous-Time Learning Dynamics）

Normal form game

$$\dot{Q}_t^i(a) = \alpha\left(U^i(a, \pi_t^{-i}) - Q_t^i(a)\right)$$

Markov game

$$\dot{y}_t^i(s,a) = \eta\left[\rho_{\pi_t}(s) A^i_{\pi_t}(s,a) - y_t^i(s,a)\right]$$

or

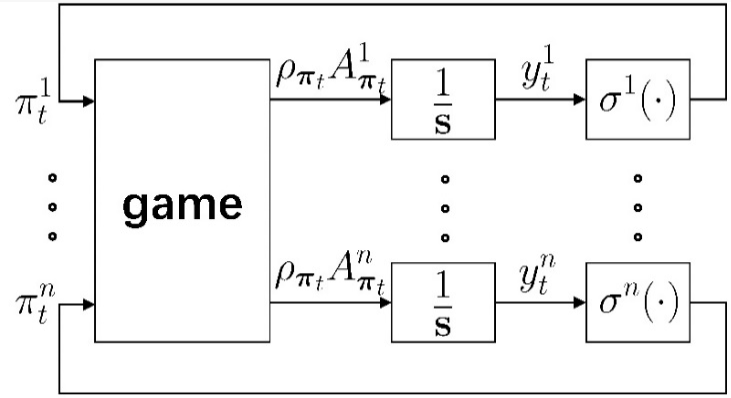
$$y_t^i(s,a^i) = e^{-\eta t} y_0^i(s,a^i) + \eta \int_0^t e^{-\eta(t-\tau)} \rho_{\pi_\tau}(s) A^i_{\pi_\tau}(s,a^i) d\tau$$

- 玩家策略生成

$$[\sigma^i(y^i)](s,a^i) = \frac{\exp\left(\frac{1}{\epsilon} y^i(s,a^i)\right)}{\sum_b \exp\left(\frac{1}{\epsilon} y^i(s,b)\right)}$$

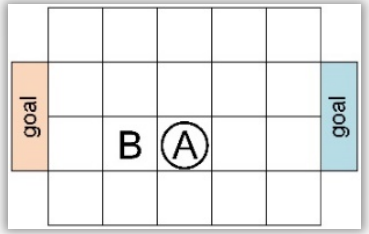
Continuous-Time Learning Dynamics

- 以连续时间运行，每个时刻完成三步计算：
  - 每个玩家根据自己的得分（score）y_t^i，计算玩家当前的策略 π_t^i
  - 所有玩家的策略在博弈环境中博弈，每个玩家观测自己策略的状态分布 ρ_{π_t} 和优势函数 $A_{\pi_t}^i$
  - 玩家根据观测量计算时间导数 $\dot{y}_t^i$，更新自己的得分

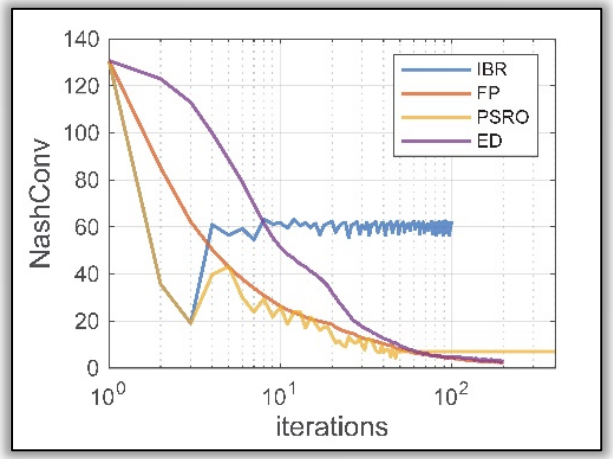


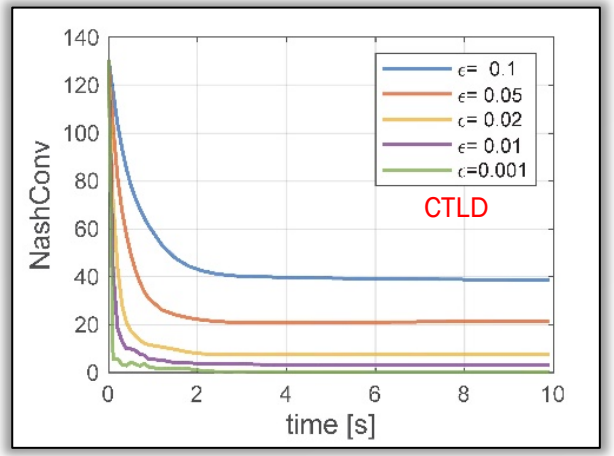
[Zhu et al, IEEE TC, 2022]

足球游戏



- Two-player
- Symmetric game
- +1 for win
- −1 for lose

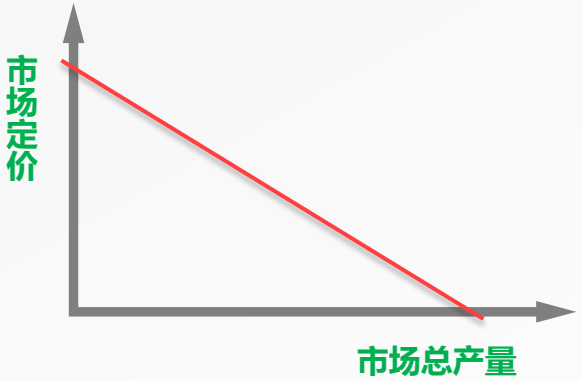




- Iterated Best Response　　不收敛
- Fictitious Play　　收敛
- PSRO　　震荡收敛，有误差
- Exploitability Decent　　收敛
- CTLD　　收敛，正则项系数越小，越接近纳什均衡解

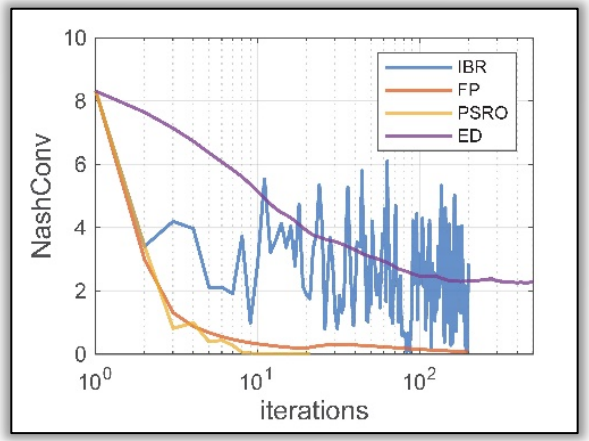
古诺竞争
Cournot Competition

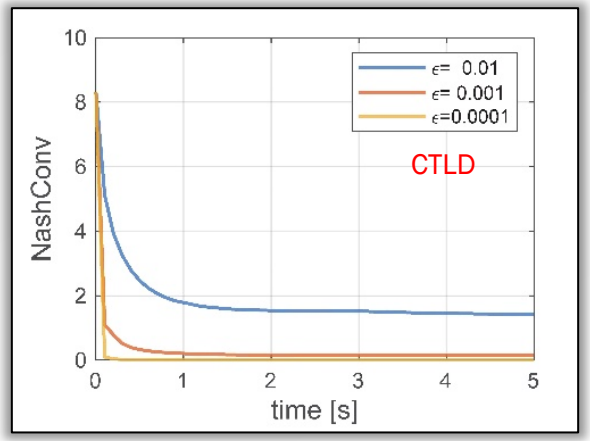
- N个玩家
- 玩家状态：商品产量 x_k^i
  - 市场总商品量 $\sum x_k^i$
  - 商品价格：$P(x_k) = a - \sum b^i x_k^i$
- 玩家动作：下月产量变化 $\Delta x_k^i \ (\pm, 0)$
- 玩家奖励：
  - 商品成本： c^i
  - 奖励信号： $r^i(\boldsymbol{x}_k) = x_k^i P(\boldsymbol{x}_k) - c^i x_k^i$



古诺竞争

Cournot Competition





- Iterated Best Response　　不收敛
- Fictitious Play　　收敛
- PSRO　　收敛
- Exploitability Decent　　很大误差
- CTLD　　收敛，正则项系数越小，越接近纳什均衡解

总结

- 多智能体强化学习
- 纳什均衡
- 价值函数均衡求解
- 策略均衡求解