第五部分： 多Agent强化学习

章宗长

2023年5月30日

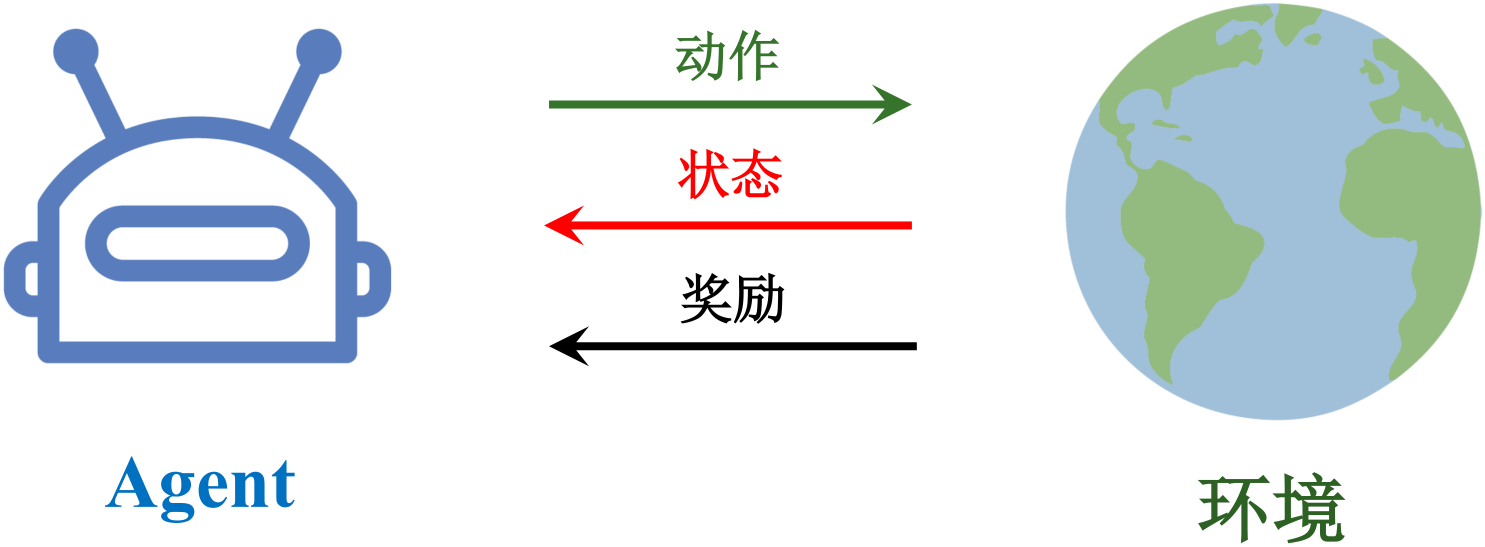
内容安排

博弈与多Agent强化学习

- **概览**
- 随机博弈
- 动态规划
- 均衡学习算法
- 最优反应学习算法

回顾：单Agent强化学习

- 目标：解决单个自治的Agent在环境中的序贯决策问题



动作

状态

奖励

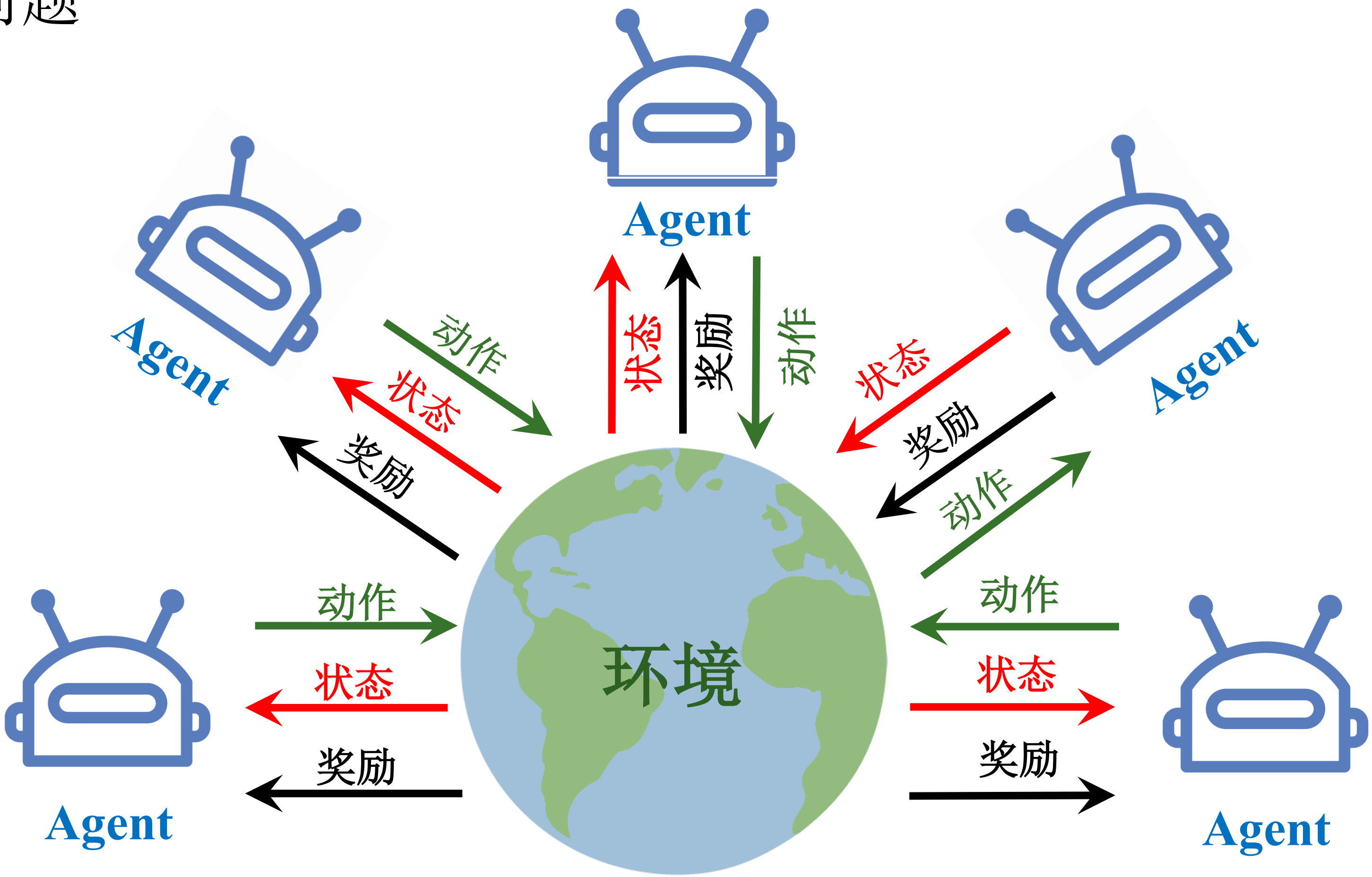
Agent

环境

- 最优策略：能产生最大化期望累积奖励（回报）的动作

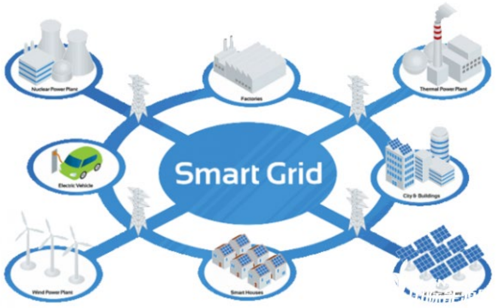
多Agent强化学习

- 目标：解决多个自治的Agent在同一个环境下的序贯决策问题



多Agent强化学习的应用

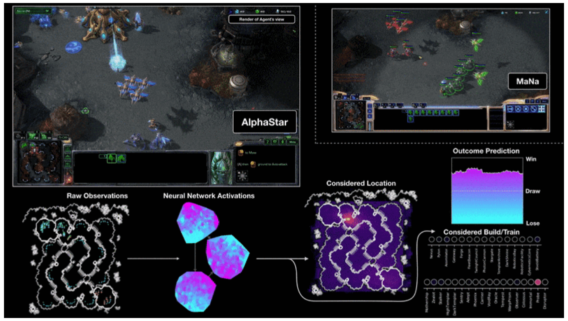
智能交通



智能电网

仓储机器人

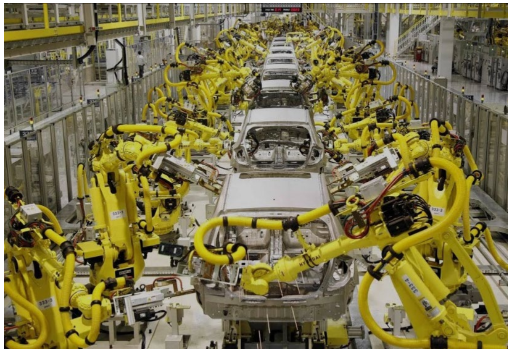
围棋



星际争霸

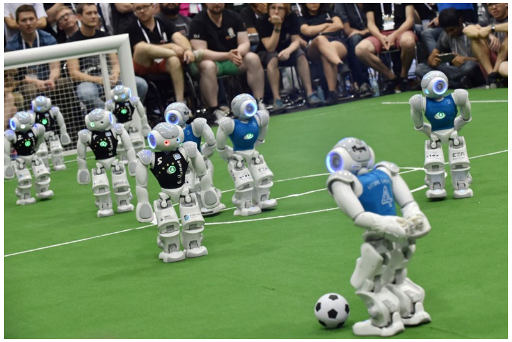
实时广告投标

多Agent强化学习任务的类型



- 完全合作
  - 所有Agent有共同的目标，获得的奖励相同

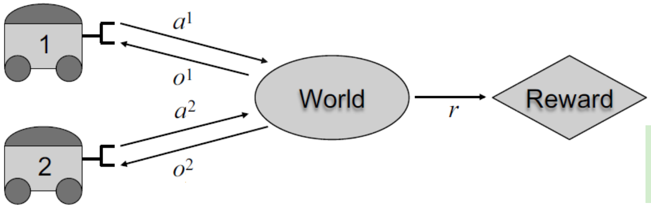
- 完全竞争
  - Agent之间是竞争关系，一方的收益就是另一方的损失



- 混合型
  - Agent分为多个群组，组内的Agent是合作关系，组间的Agent是竞争关系

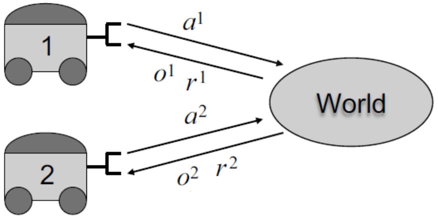
多Agent强化学习的目标

- 学习目标
  - 在完全合作型环境下，Agent学习的目标是清晰的，即最大化团队回报



所有Agent共享一个奖励函数

  - 在非完全合作型环境下，Agent学习的目标不唯一
    - 不存在绝对的最优策略，通常是寻找纳什均衡解
    - 纳什均衡解本身具有一定的局限性，很多情况下并非最优



不同Agent可以有不同的奖励函数

多Agent强化学习的挑战

- 非稳态性
  - 由于环境中的Agent都在进行学习，其策略在不断发生变化
  - 从单个Agent的角度看，其学习环境处在不断地变化之中
- 部分可观察性
  - 单个Agent无法从局部观察中获得学习系统的全部信息
  - Agent往往无法只通过当前的局部观察做出正确决策
- 可扩展性
  - 随着Agent数量的不断增加，问题的搜索空间呈现指数级增长
- ……

序贯决策

- 包括：
  - 马尔可夫决策过程（MDP）
    - 包括一个决策者
    - 多个状态
  - 重复式博弈（Repeated Game）
    - 包括多个决策者
    - 一个状态（如：一个正则形式的博弈）
  - 随机博弈（Stochastic/Markov Game）
    - 包括多个决策者
    - 多个状态（如：多个正则形式的博弈）

MDP
- 一个Agent
- 多个状态

重复式博弈
- 多个Agent
- 一个状态

随机博弈
- 多个Agent
- 多个状态

回顾：马尔可夫决策过程（MDP）

- 一个MDP问题可以形式化地建模为
  - 有限的状态集合$\mathcal{S}$
  - 有限的动作集合$\mathcal{A}$
  - 状态转移函数$T(s'|s,a)$
    - Agent在状态s执行动作a转移到新的状态s'的概率
  - 奖励函数$R(s,a)$
    - Agent在状态s执行动作a所能得到的即时奖励
- MDP模型的特点
  - 考虑了状态转移的不确定性
  - Agent可以直接获得环境的状态信息

回顾：MDP的策略和值函数

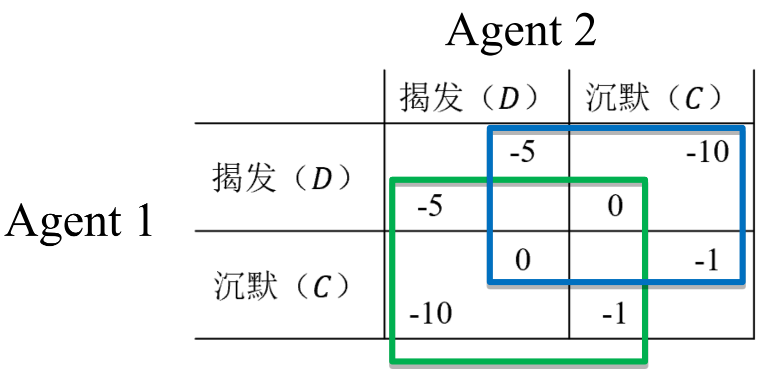
- 策略π：状态s到动作a的映射
  - 确定性策略$\pi(s)$：$\mathcal{S} \longrightarrow \mathcal{A}$
  - 随机性策略$\pi(a \mid s)$：$\mathcal{S} \times \mathcal{A} \longrightarrow [0,1]$

- 值函数$V^{\pi}(s)$：由状态s开始，执行策略π所能获得的期望折扣回报

$$\mathbb{E}_{\pi}\left[\sum_{t=0}^{T-1} \gamma^{t} R(s_t, \pi(s_t)) \mid s_0 = s\right]$$

回顾：正则形式的博弈

囚徒困境

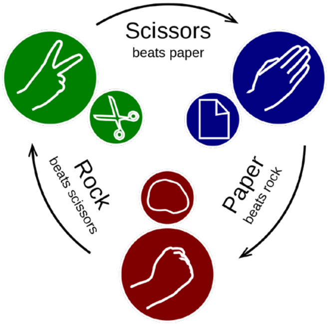


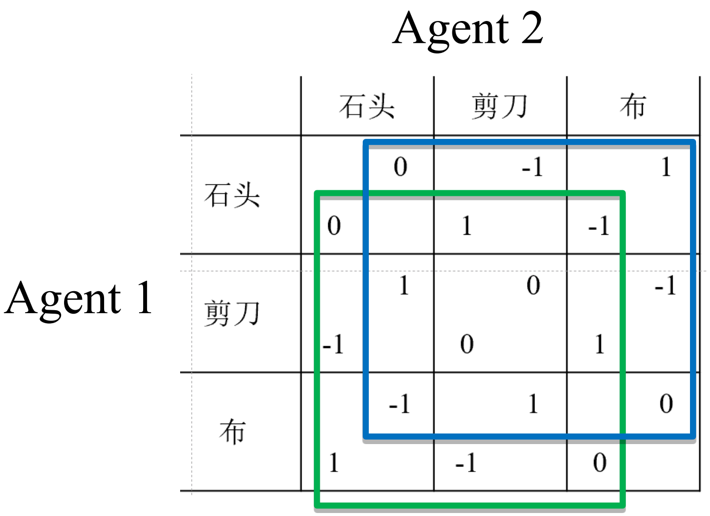
Agent 2

		揭发（*D*）	沉默（*C*）
Agent 1	揭发（*D*）	-5 -5	-10 0
	沉默（*C*）	0 -10	-1 -1

$$R_1=\begin{bmatrix}-5 & 0\\ -10 & -1\end{bmatrix},\ R_2=(R_1)^{\mathrm{T}}$$

石头-剪刀-布





Agent 2

		石头	剪刀	布
Agent 1	石头	0 0	-1 1	1 -1
	剪刀	1 -1	0 0	-1 1
	布	-1 1	1 -1	0 0

$$R_1=\begin{bmatrix}0 & 1 & -1\\ -1 & 0 & 1\\ 1 & -1 & 0\end{bmatrix},\ R_2=-R_1$$

正则形式的博弈：定义

下标i表示Agent的序号

- 一个正则形式的博弈可以定义为 $(m, \mathcal{A}_{1\ldots m}, R_{1\ldots m})$：
  - m：Agent个数
  - $\mathcal{A}_i$：Agent i的动作空间（联合动作空间$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$）
  - R_i：第i个Agent的奖励函数$\mathcal{A} \to \mathbb{R}$

- Agent i的优化目标

$$\pi_i \in \Delta(\mathcal{A}_i), \qquad \text{最大化}\mathbb{E}_{\mathbf{a}\sim\pi}[R_i(\mathbf{a})]$$

这是一个联合策略！

正则形式的博弈：最优反应和纳什均衡

记为$BR(\pi_{-i})$

- π_i是Agent i对策略组合π_{-i}的最优反应，当且仅当：

$$R_i(\pi_i, \pi_{-i}) = \max_{\pi_i'} \mathbb{E}_{a\sim(\pi_i', \pi_{-i})}[R_i(\mathbf{a})]$$

- 联合策略$\pi = (\pi_i, \pi_{-i})$是纳什均衡策略，当且仅当：

$$\forall i \in \{1, 2, \dots, m\},\ \pi_i \in BR(\pi_{-i})$$

零和博弈

- 零和博弈只有唯一的纳什均衡
  - 每个Agent可能具有多种纳什均衡策略
  - 但在这些纳什均衡策略下，每个Agent的期望奖励相同

博弈的价值，记为V

- 如何求解2×2零和博弈的价值和纳什均衡策略？

$$R_1 = \begin{bmatrix} a & b \\ d & c \end{bmatrix}, R_2 = -R_1$$

  - 步骤1：先寻找纯策略的纳什均衡
  - 步骤2：若没有，再寻找混合策略的纳什均衡

求解2×2零和博弈

$$R_1 = \begin{bmatrix} a & b \\ d & c \end{bmatrix}$$

- 步骤1：寻找纯策略的纳什均衡

不存在纯策略的纳什均衡的条件

$a > b, b < c, c > d, d < a$
或者
$a < b, b > c, c < d, d > a$

- 例子

$$R_1 = \begin{bmatrix} -2 & 3 \\ 3 & -4 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} 0 & -10 \\ \enclose{circle}{1} & 2 \end{bmatrix}$$

满足$a < b, b > c, c < d, d > a$，
不存在纯策略的纳什均衡

不满足上述任意条件，存在
纯策略的纳什均衡

求解2×2零和博弈（续）

$$R_1 = \begin{bmatrix} a & b \\ d & c \end{bmatrix}$$

- 步骤2：若没有，再寻找混合策略的纳什均衡
  - 假设Agent 1选择动作1（第一行）的概率为p，选择动作2（第二行）的概率为$1-p$，则有

$$ap + d(1-p) = bp + c(1-p)$$

  - 得到

$$p = \frac{c-d}{(a-b)+(c-d)}$$

  - 博弈的价值

$$V = ap + d(1-p) = \frac{ac-bd}{a-b+c-d}$$

例子

$$R_1=\begin{bmatrix} a & b \\ d & c \end{bmatrix}$$

$$p=\frac{c-d}{(a-b)+(c-d)}$$

$$V=ap+d(1-p)$$

- 例1

$$R_1=\begin{bmatrix} -2 & 3 \\ 3 & -4 \end{bmatrix}$$ ⇨ $$p=\frac{7}{12},\qquad q=\frac{7}{12},\qquad V=\frac{1}{12}$$

$$aq+b(1-q)=dq+c(1-q)$$

- 例2

$$R_1=\begin{bmatrix} 0 & -10 \\ \textcircled{1} & 2 \end{bmatrix}$$ ⇨ $$p=\frac{1}{11},\qquad q=\frac{12}{11}>1!$$

- 为什么？
  - 因为有纯策略的纳什均衡策略，对应的$p=0$、$q=1$

回顾：重复式博弈

- 重复进行的正则形式的博弈
  - 进行多次对局，每局的收益矩阵相同
  - 每个Agent都可以看到其他Agent前一轮的动作

- 如：重复进行的囚徒困境
  - 在重复固定轮数的囚徒困境中，不合作是理性行为
  - 在重复无限轮数的囚徒困境中，合作是理性行为

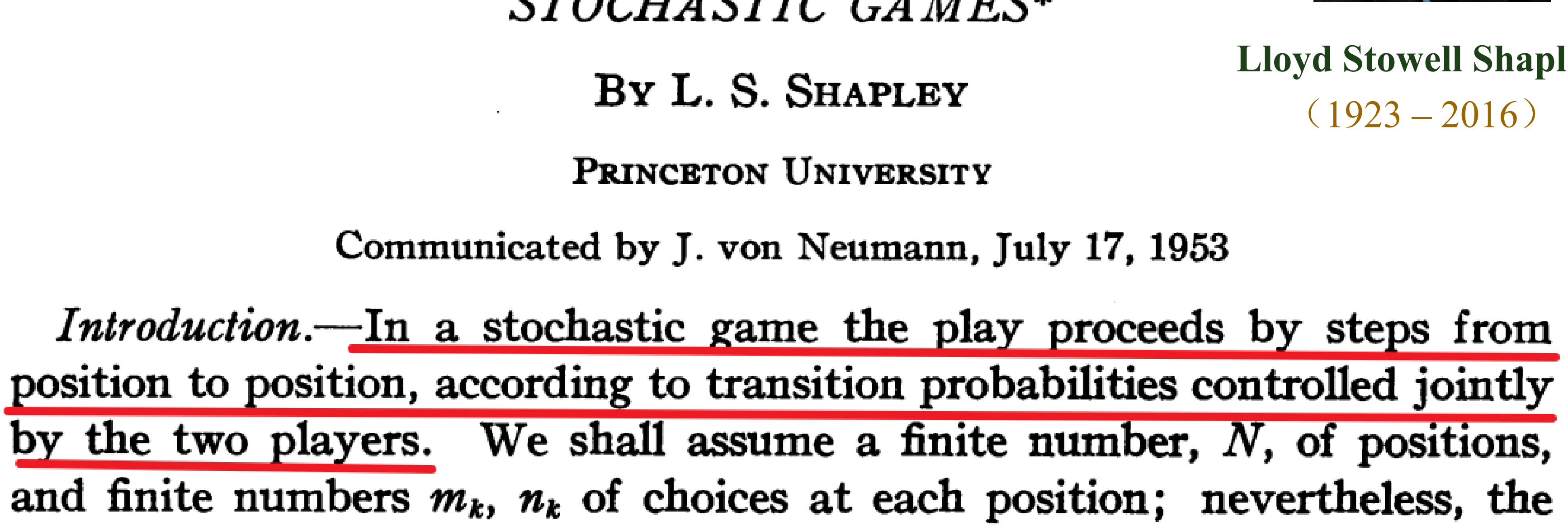
博弈与多Agent强化学习

- 概览
- **随机博弈**
- 动态规划
- 均衡学习算法
- 最优反应学习算法

随机博弈

Lloyd Stowell Shapley

（1923 – 2016）



VOL. 39, 1953 *MATHEMATICS: L. S. SHAPLEY*

*STOCHASTIC GAMES**

BY L. S. SHAPLEY

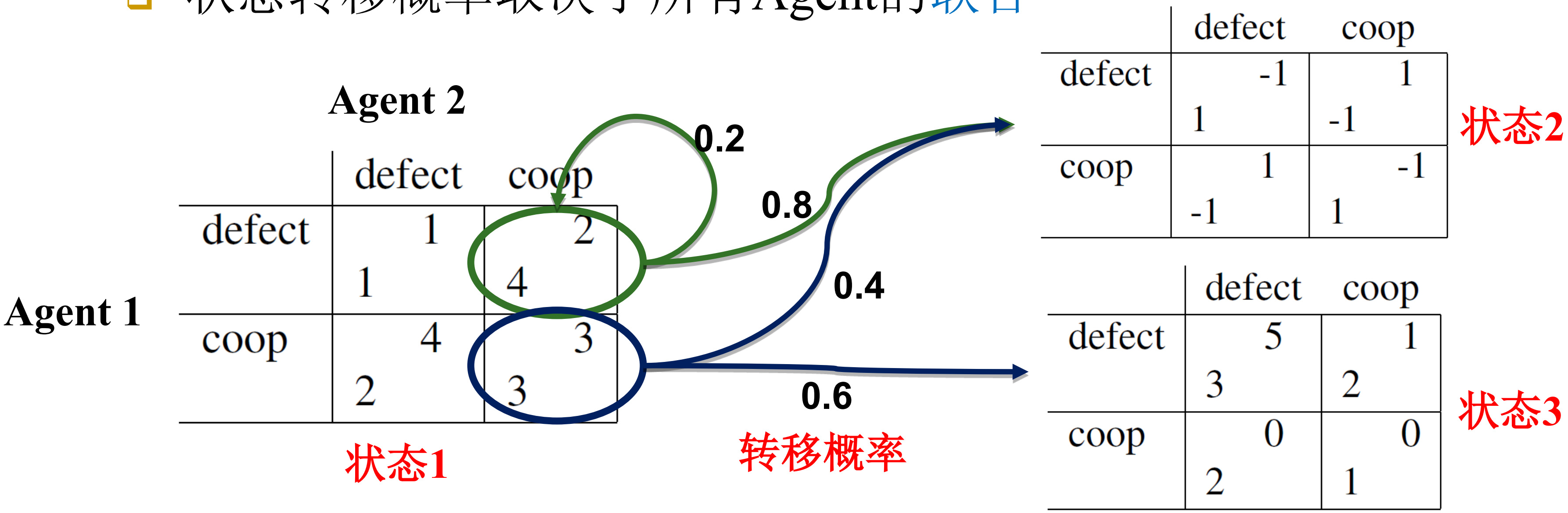
PRINCETON UNIVERSITY

Communicated by J. von Neumann, July 17, 1953

Introduction.—In a stochastic game the play proceeds by steps from position to position, according to transition probabilities controlled jointly by the two players. We shall assume a finite number, N, of positions, and finite numbers m_k, n_k of choices at each position; nevertheless, the

随机博弈

- 一个随机博弈包含多个状态和多个Agent
  - 每个状态对应于一个正则形式的博弈
  - 每一轮结束，博弈转移到另一个状态
  - 状态转移概率取决于所有Agent的联合动作



Agent 2

Agent 1

	defect	coop
defect	1 1	2 4
coop	4 2	3 3

状态1

	defect	coop
defect	-1 1	1 -1
coop	1 -1	-1 1

状态2

	defect	coop
defect	5 3	1 2
coop	0 2	0 1

状态3

- 单状态随机博弈 = （无限轮数的）重复式博弈
- 单Agent随机博弈 = 马尔可夫决策过程

随机博弈：定义

下标i表示Agent的序号

- 一个随机博弈可以定义为元组$(m, S, \mathcal{A}_{1\ldots m}, T, R_{1\ldots m})$：
  - m：Agent个数
  - $\mathcal{S}$：状态集合
  - $\mathcal{A}_i$：Agent i的动作空间（联合动作空间$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$）
  - T：状态转移函数$S \times \mathcal{A} \times S \rightarrow [0,1]$
  - R_i：Agent i的奖励函数$S \times \mathcal{A} \rightarrow \mathbb{R}$
- 和马尔可夫决策过程的区别
  - 有多个Agent
  - 下个状态依赖于联合动作
  - 每个Agent有自己的奖励函数

随机博弈的分类

- 零和随机博弈（竞争）
  - 所有状态都是正则形式的零和博弈

- 团队随机博弈（合作）
  - 所有状态定义的收益矩阵，对于所有Agent来说都相同

- 一般和随机博弈（General-sum stochastic games）
  - 不属于以上两种类型的其他随机博弈

例5-1：胆量（Dare）博弈

- 两个Agent：Agent 1为擂主， Agent 2为挑战者
  - 如果同时采取动作pass，则游戏结束，收益为0
  - 如果Agent 1采取动作pass, Agent 2采取动作dare，Agent 1获得收益1， Agent 2获得收益−1
  - 如果Agent 1采取动作dare，Agent 2采取动作pass，Agent 1获得收益3， Agent 2获得收益−3
  - 如果同时采取动作dare，则角色互换，收益为0，进入下一轮游戏
- 可以建模为有两个状态的零和随机博弈

$$G = \begin{array}{c c} & \begin{array}{cc} \text{pass} & \text{dare} \end{array} \\ \begin{array}{c} \text{pass} \\ \text{dare} \end{array} & \begin{pmatrix} 0 & 1 \\ 3 & -G^{\mathrm{T}} \end{pmatrix} \end{array}$$

博弈G^{T}为博弈G的转置，它们有相同的价值

例5-1：胆量博弈（续）

- 有两个状态的零和随机博弈

$$G = \begin{array}{c|cc} & \text{pass} & \text{dare} \\ \hline \text{pass} & 0 & 1 \\ \text{dare} & 3 & -G^{\mathrm{T}} \end{array} \Leftrightarrow G^{(1)} = \begin{pmatrix} 0 & 1 \\ 3 & G^{(2)} \end{pmatrix} \quad G^{(2)} = \begin{pmatrix} 0 & -3 \\ -1 & G^{(1)} \end{pmatrix}$$

- 令博弈G的价值为V，有

$$V = ap + d(1-p) = \frac{ac-bd}{a-b+c-d}$$

$$V = Val\begin{pmatrix} 0 & 1 \\ 3 & -V \end{pmatrix} = \frac{3}{4+V}$$

- 从而有$V^2 + 4V - 3 = 0 \Rightarrow V = \sqrt{7} - 2$

因为Agent 1执行动作pass，不论Agent 2执行哪个动作，收益均大于等于0，所以$V \geq 0$

- 从而有$p = \frac{5-\sqrt{7}}{3}$，$q = 3 - \sqrt{7} \Rightarrow$

Agent 1的最优策略$(\frac{5-\sqrt{7}}{3}, \frac{\sqrt{7}-2}{3})$
Agent 2的最优策略$(3-\sqrt{7}, \sqrt{7}-2)$

例5-2

- 求解以下随机博弈问题的混合策略纳什均衡

$$G^{(1)} = \begin{pmatrix} \frac{1}{2}(0) + \frac{1}{2}G^{(2)} & 1 \\ 2 & 0 \end{pmatrix} \qquad G^{(2)} = \begin{pmatrix} \frac{1}{3}(-2) + \frac{2}{3}G^{(1)} & 0 \\ 0 & -1 \end{pmatrix}$$

$$V = ap + d(1-p) = \frac{ac - bd}{a - b + c - d}$$

- 令博弈$G^{(i)}$的价值为V_i，有

$$V_1 = \mathrm{Val}\begin{pmatrix} \frac{1}{2}V_2 & 1 \\ 2 & 0 \end{pmatrix} = \frac{4}{6 - V_2} \qquad V_2 = \mathrm{Val}\begin{pmatrix} \frac{2}{3}V_1 - \frac{2}{3} & 0 \\ 0 & -1 \end{pmatrix} = -\frac{2(1 - V_1)}{5 - 2V_1}$$

- 从而有$7V_1^2 - 20V_1 + 10 = 0$ $\Rightarrow$ $V_1 = \frac{10 - \sqrt{30}}{7}$
- 从而有$V_2 = -\frac{2\sqrt{30} - 10}{5}$

博弈与多Agent强化学习

- 概览
- 随机博弈
- **动态规划**
- 均衡学习算法
- 最优反应学习算法

随机博弈的状态值函数

- 随机博弈中Agent *i*的状态值函数：

$$\begin{aligned}
V_i^\pi(s) =&\mathbb{E}_\pi\left\{\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+1) \mid s_t=s\right\}\\
=&\mathbb{E}_\pi\left\{r_i(t+1)+\gamma\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+2) \mid s_t=s\right\}\\
=&\sum_a \pi(a \mid s)\sum_{S'} p(s' \mid s,a)\left[r_i(s',a)+\gamma\mathbb{E}_\pi\left\{\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+2) \mid s_{t+1}=s'\right\}\right]\\
=&\sum_a \underbrace{\pi(a \mid s)}_{\text{在状态}s\text{时选取联合动作}a\text{的概率}}\sum_{s'} p(s' \mid s,a)\left[r_i(s',a)+\gamma V_i^\pi(s')\right]
\end{aligned}$$

和MDP类似

在状态*s*时选取联合动作*a*的概率

$$V_i^\pi(s) \le \frac{M}{1-\gamma}$$

其中 $M \equiv \max_{i,s,a}|r_i(s,a)|$

- 每个Agent都有自己的状态值函数
  - 原因：每个Agent有自己的奖励函数
  - 依赖于所有Agent的联合策略，而不仅仅是Agent自身的决策

每个状态的博弈

- 定义每个状态s的博弈为

$$G^{(s)} = \left(\underbrace{r_i(s, a_i, a_{-i})}_{\text{在状态}s\text{执行联合动作}(a_i, a_{-i})\text{后得到的收益/奖励}} + \gamma \sum_{s'} p(s' \mid s, a_i, a_{-i}) G^{(s')} \right)$$

- 和正则形式的博弈不同的是，一轮博弈后不一定会结束
  - 博弈结束的概率为$1 - \gamma$
  - 如果未结束，则进入下一个状态再次进行博弈

- 例

$$G^{(1)} = \begin{pmatrix} \frac{1}{2}(0) + \frac{1}{2}G^{(2)} & 1 \\ 2 & 0 \end{pmatrix} \quad G^{(2)} = \begin{pmatrix} \frac{1}{3}(-2) + \frac{2}{3}G^{(1)} & 0 \\ 0 & -1 \end{pmatrix}$$

值迭代

- **定理** (Shapley (1953))：每次博弈$G^{(s)}$都有一个状态值$V(s)$。这些状态值是以下方程组的唯一解

$$V(s) = \underline{\mathrm{Val}}\left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p\big(s' \mid s, a_i, a_{-i}\big) V(s')\right) \quad \text{for } s \in \mathcal{S}$$

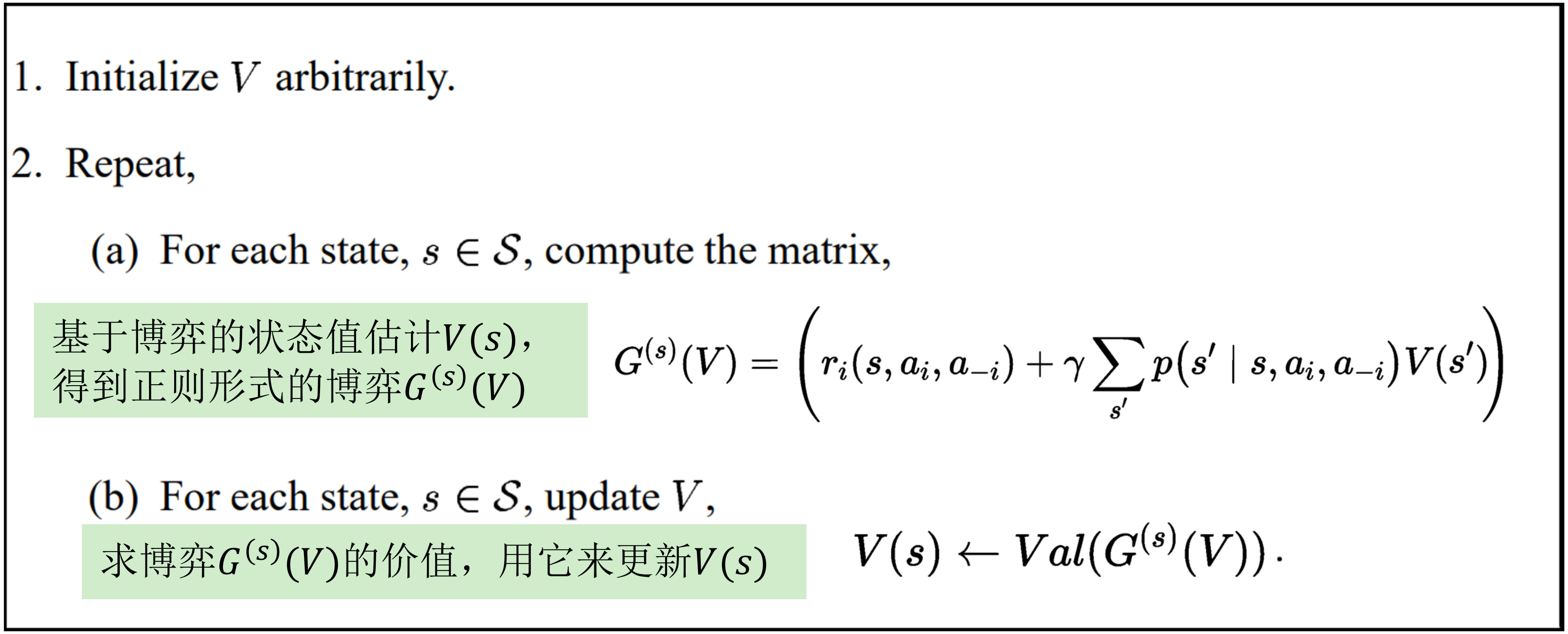
Val：求解博弈在纳什均衡时的价值

- 给定V，$G^{(s)}(V)$是一个正则形式的博弈

$$G^{(s)}(V) = \left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p\big(s' \mid s, a_i, a_{-i}\big) V(s')\right)$$

- 每个Agent在状态s对应的博弈$G^{(s)}$有一个固定的最优混合策略

值迭代（续）



1. Initialize V arbitrarily.
2. Repeat,
   (a) For each state, $s \in \mathcal{S}$, compute the matrix,

基于博弈的状态值估计$V(s)$，得到正则形式的博弈$G^{(s)}(V)$

$$G^{(s)}(V) = \left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s' \mid s, a_i, a_{-i}) V(s') \right)$$

   (b) For each state, $s \in \mathcal{S}$, update V,

求博弈$G^{(s)}(V)$的价值，用它来更新$V(s)$

$$V(s) \leftarrow Val(G^{(s)}(V)).$$

- 与MDP中的值迭代比较
  - 把MDP中的max算子改成了Val算子

例5-3

- 考虑只有一个状态的零和随机博弈问题

$$G = \begin{pmatrix} 1 + (3/5)G & 3 + (1/5)G \\ 1 + (4/5)G & 2 + (2/5)G \end{pmatrix}$$

问：两个Agent的最优策略分别是什么？

分析：

两个Agent都选择动作1，那么Agent 1得到奖励1，且有3/5的概率再次进行博弈

从Agent 2的角度出发，选取动作2的即时奖励比动作1高，但选取动作2博弈结束的概率更高，所以会导致未来奖励变小

例5-3（续）

- 假设最优策略不是纯策略，可以得到解后验证
- 当收敛时应有

$$V=ap+d(1-p)=\frac{ac-bd}{a-b+c-d}$$

$$V=\mathrm{Val}\begin{pmatrix}1+(3/5)V & 3+(1/5)V\\ 1+(4/5)V & 2+(2/5)V\end{pmatrix}$$

$$=\frac{(1+(3/5)V)(2+(2/5)V)-(1+(4/5)V)(3+(1/5)V)}{1+(3/5)V-3-(1/5)V+2+(2/5)V-1-(4/5)V}$$

$$=1+V-(2/25)V^2$$

- 由于V为正，解得$V=\frac{5\sqrt{2}}{2}$
- 将V代回得到博弈矩阵：

$$\boxed{\begin{pmatrix}1+(3/2)\sqrt{2} & 3+(1/2)\sqrt{2}\\ 1+2\sqrt{2} & 2+\sqrt{2}\end{pmatrix}}$$

- 解得两个Agent的混合策略
  - Agent 1：$(\sqrt{2}-1,2-\sqrt{2})$，Agent 1：$(1-\sqrt{2}/2,\sqrt{2}/2)$

与假设一致！

例5-4

- 考虑有两个状态的2×2零和随机博弈问题

$$G^{(1)} = \begin{pmatrix} 4 + .3G^{(1)} & 0 + .4G^{(2)} \\ 1 + .4G^{(2)} & 3 + .5G^{(1)} \end{pmatrix} \quad G^{(2)} = \begin{pmatrix} 0 + .5G^{(1)} & -5 \\ -4 & 1 + .5G^{(2)} \end{pmatrix}$$

（1）该博弈在状态1和状态2的价值分别是多少？

（2）两个Agent在状态1和状态2的最优策略分别是什么？

2×2零和博弈的价值：

$$V = \frac{ac - bd}{a - b + c - d}$$

值迭代的更新规则：

$$V(s) = \mathrm{Val}\left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s' \mid s, a_i, a_{-i}) V(s') \right) \quad \text{for } s \in S$$

例5-4（续）

- 初始化$V_0 = (0,0)$，则有

$$V_1(1) = \mathrm{Val}\begin{pmatrix} 4 & 0 \\ 1 & 3 \end{pmatrix} = 2 \quad V_1(2) = \mathrm{Val}\begin{pmatrix} 0 & -5 \\ -4 & 1 \end{pmatrix} = -2$$

- 即$V_1 = (2, -2)$，继续迭代

$$V_2(1) = \mathrm{Val}\begin{pmatrix} 4.6 & -.8 \\ .2 & 4 \end{pmatrix} = 2.0174 \quad V_2(2) = \mathrm{Val}\begin{pmatrix} 1 & -5 \\ -4 & 0 \end{pmatrix} = -2$$

- 继续迭代，有

$$\begin{array}{ll} V_3(1) = 2.0210 & V_3(2) = -1.9983 \\ V_4(1) = 2.0220 & V_4(2) = -1.9977 \\ V_5(1) = 2.0224 & V_5(2) = -1.9974 \\ V_6(1) = 2.0225 & V_6(2) = -1.9974 \end{array}$$

因为最小的终止概率为0.5，所以收敛率至少为0.5^n，V_6的最大误差至多为0.0002

例5-4（续）

- 使用V_6来求解最优策略

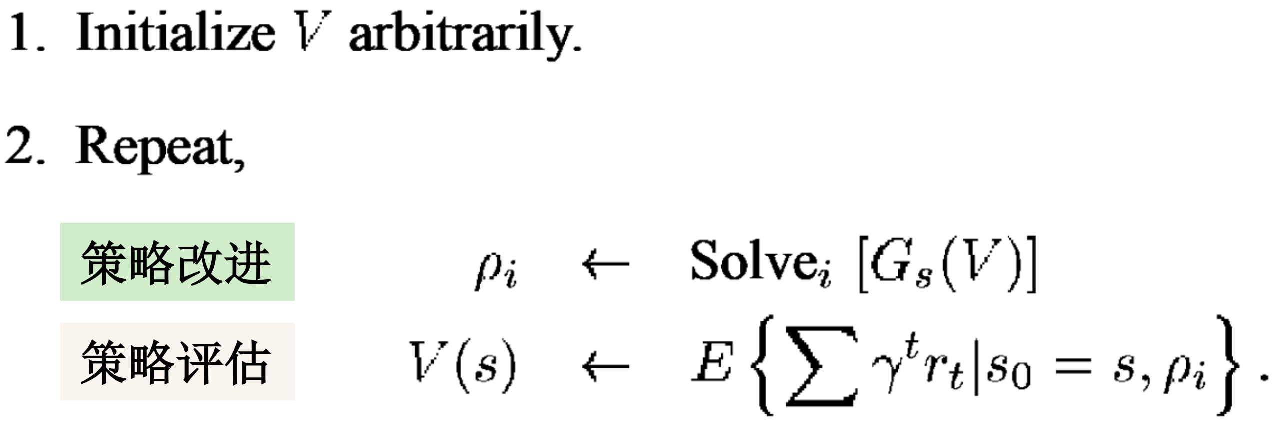
$$V_6(1) = 2.0225 \qquad V_6(2) = -1.9974$$

$$\boxed{G^{(s)}(V) = \left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s' \mid s, a_i, a_{-i}) V(s') \right)}$$

$$G^{(1)}(V_6) = \begin{bmatrix} 4.60675 & -0.79896 \\ 0.20104 & 4.01125 \end{bmatrix} \qquad G^{(2)}(V_6) = \begin{bmatrix} 1.01125 & -5 \\ -4 & 0.0013 \end{bmatrix}$$

- 博弈$G^{(1)}$的最优策略为
  - Agent 1：$(0.4134, 0.5866)$
  - Agent 2：$(0.5219, 0.4718)$
- 博弈$G^{(2)}$的最优策略为
  - Agent 1：$(0.3996, 0.6004)$
  - Agent 2：$(0.4995, 0.5005)$

策略迭代



1. Initialize V arbitrarily.
2. Repeat,

策略改进 $$\rho_i \leftarrow \text{Solve}_i\left[G_s(V)\right]$$

策略评估 $$V(s) \leftarrow E\left\{\sum \gamma^t r_t | s_0 = s, \rho_i\right\}.$$

- 策略评估：根据当前策略的实际回报来更新值函数
- 策略改进：每个Agent根据当前值函数来选择均衡策略

博弈与多Agent强化学习

- 概览
- 随机博弈
- 动态规划
- **均衡学习算法**
- 最优反应学习算法

均衡学习算法

- 目标
  - 找到随机博弈的纳什均衡策略
  - 求解一般和随机博弈的纳什均衡策略是困难的
    - 现有学习算法往往聚焦于一类子问题
    - 如：零和博弈或者双人一般和随机博弈
- 优点
  - 能保证每个Agent性能的下界
  - 这个下界不依赖于其他Agent的策略
  - 每个Agent至少能得到纳什均衡策略对应的回报
- 缺点
  - Agent无法根据对手策略的变化调整其策略

随机博弈的动作值函数

- 随机博弈中Agent *i*的动作值函数：

$$
\begin{aligned}
&Q_i^\pi(s,a) \\
&=\mathbb{E}_\pi\left\{\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+1) \mid s_t=s, a_t=a\right\} \\
&=\mathbb{E}_\pi\left\{r_i(t+1)+\gamma\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+2) \mid s_t=s, a_t=a\right\} \\
&=\sum_{s'} p(s' \mid s,a)\left[r_i(s',a)+\gamma\mathbb{E}_\pi\left\{\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+2) \mid s_{t+1}=s', a_t=a\right\}\right] \\
&=\sum_{s'} p(s' \mid s,a)\left[r_i(s',a)+\gamma V_i^\pi(s')\right]
\end{aligned}
$$

和MDP类似

在状态s时选取联合动作a后转投到状态s'的概率

- 每个Agent都有自己的动作值函数
    - 依赖于所有Agent的联合策略，而不仅仅是Agent自身的决策

均衡学习算法

- 一个均衡学习算法的解可以表示成满足下列方程组的一个不动点$\pi^* = (\pi_i^*, \pi_{-i}^*)$，满足

和MDP中的贝尔曼最优等式类似

$$\forall_{i=1\ldots n} \qquad Q_i^*(s,a) = r_i(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i^*(s')$$

$$\underline{V_i^*(s)} = \sum_s \pi^*(a \mid s) Q_i^*(s,a)$$

联合策略是纳什均衡策略π^*时，Agent i在状态s获得的期望回报

极小极大Q学习算法

Minimax Q

- 解决的是双人零和随机博弈问题
- 给定状态s，Agent i的状态值函数为

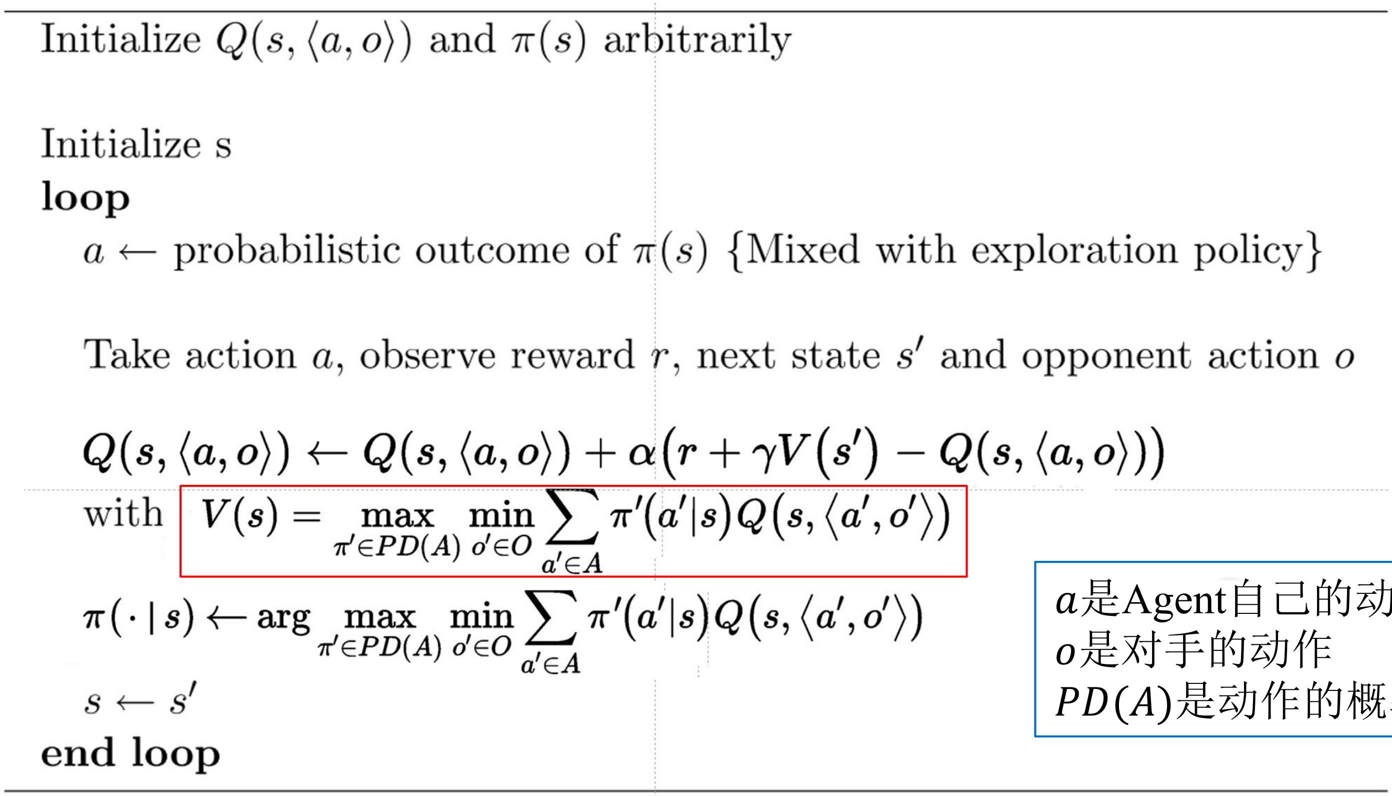
$$V_i^*(s) = \max_{\pi_i(\cdot \mid s)} \min_{a_{-i} \in A_{-i}} \sum_{a_i \in A_i} \pi_i(a_i \mid s) Q_i^*(s, a_i, a_{-i})$$

对比上一页的$V_i^*(s) = \sum_s \pi^*(a \mid s) Q_i^*(s, a)$

上式需要使用线性规划求解

- 如果无限频繁访问所有可能的状态和Agent所有可能的动作，则该算法可保证收敛于纳什均衡

极小极大Q学习算法（续）



```
Initialize $Q(s,\langle a,o\rangle)$ and $\pi(s)$ arbitrarily

Initialize s
loop
    $a \leftarrow$ probabilistic outcome of $\pi(s)$ {Mixed with exploration policy}
    Take action $a$, observe reward $r$, next state $s'$ and opponent action $o$
    $Q(s,\langle a,o\rangle) \leftarrow Q(s,\langle a,o\rangle) + \alpha\left(r + \gamma V(s') - Q(s,\langle a,o\rangle)\right)$
    with $\boxed{V(s) = \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi'(a'|s) Q(s,\langle a',o'\rangle)}$
    $\pi(\cdot|s) \leftarrow \arg\max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi'(a'|s) Q(s,\langle a',o'\rangle)$
    $s \leftarrow s'$
end loop
```

a是Agent自己的动作
o是对手的动作
$PD(A)$是动作的概率分布

- 缺点
  - 在每次迭代中必须采用线性规划来求解状态值和策略
  - 是一个与对手无关的算法

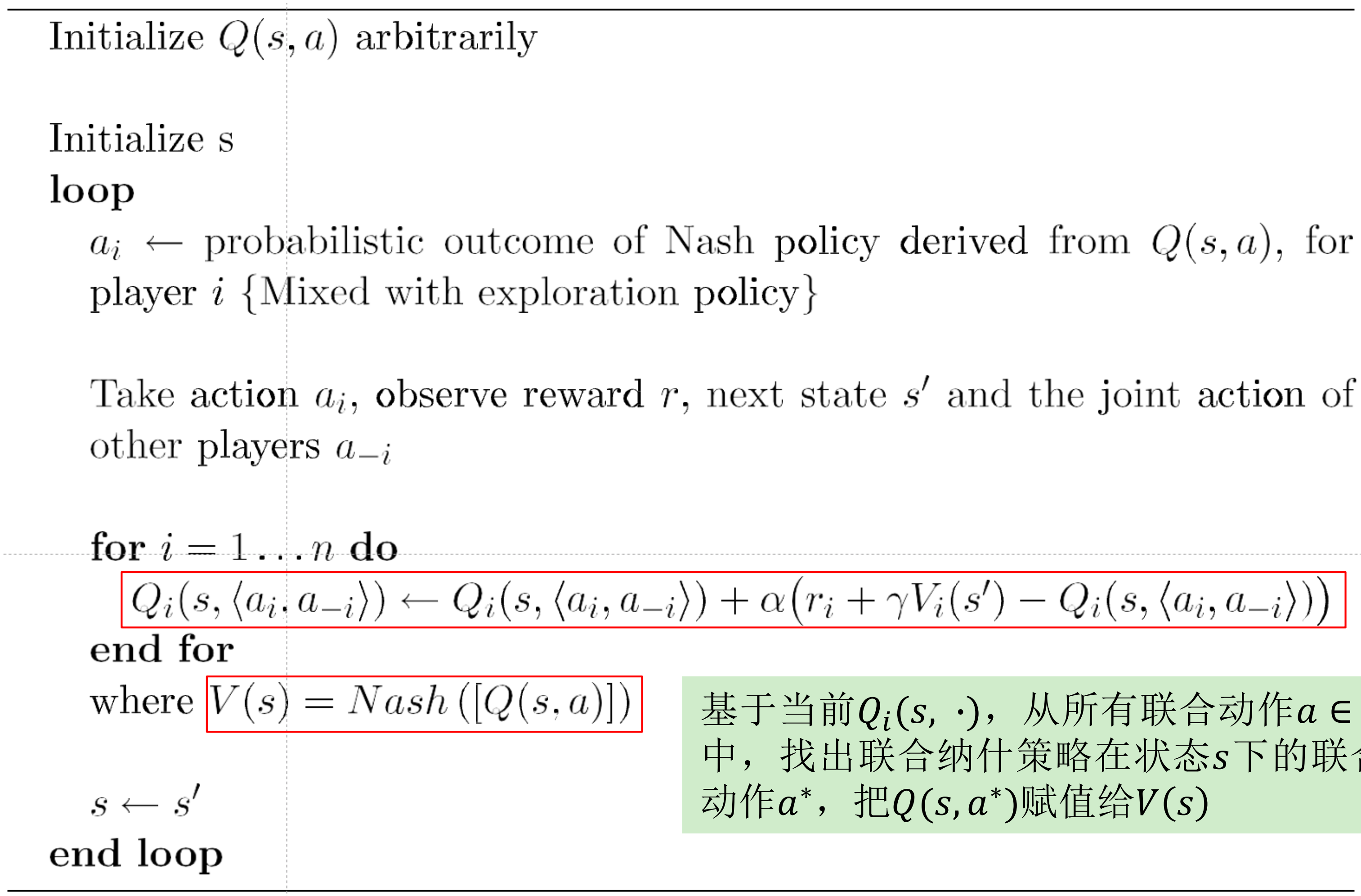
Littman, Michael L. “Markov games as a framework for multi-agent reinforcement learning”. ICML 1994.

纳什Q学习算法

Nash Q

- 将极小极大Q学习算法从零和随机博弈扩展到一般和博弈
- 定义最优Q值为纳什均衡策略下所得到的值，并称之为纳什Q值
  - 需要使用二次规划来求解一般和的纳什均衡
  - 需要学习其他Agent的Q值
    - 求解纳什均衡时，每个Agent需要知道其他Agent的Q值
- 需计算每个状态下的纳什Q值，以更新动作值函数并得到均衡策略

纳什Q学习算法（续）



```
Initialize $Q(s, a)$ arbitrarily

Initialize s
loop
  $a_i \leftarrow$ probabilistic outcome of Nash policy derived from $Q(s, a)$, for player $i$ {Mixed with exploration policy}

  Take action $a_i$, observe reward $r$, next state $s'$ and the joint action of other players $a_{-i}$

  for $i = 1 \ldots n$ do
    $\boxed{Q_i(s, \langle a_i, a_{-i} \rangle) \leftarrow Q_i(s, \langle a_i, a_{-i} \rangle) + \alpha\big(r_i + \gamma V_i(s') - Q_i(s, \langle a_i, a_{-i} \rangle)\big)}$
  end for
  where $\boxed{V(s) = Nash\left([Q(s, a)]\right)}$

  $s \leftarrow s'$
end loop
```

基于当前$Q_i(s,\ \cdot)$，从所有联合动作$a \in A$中，找出联合纳什策略在状态s下的联合动作a^*，把$Q(s, a^*)$赋值给$V(s)$

Hu, Junling and Wellman, Michael P. “Nash Q-learning for general-sum stochastic games.” Journal of Machine Learning Research, vol 4, pp. 1039-1069, 2003.

朋友或敌人Q学习算法

Friend-or-Foe Q

- 也是Minimax-Q的一种拓展，用来解决一般和随机博弈问题

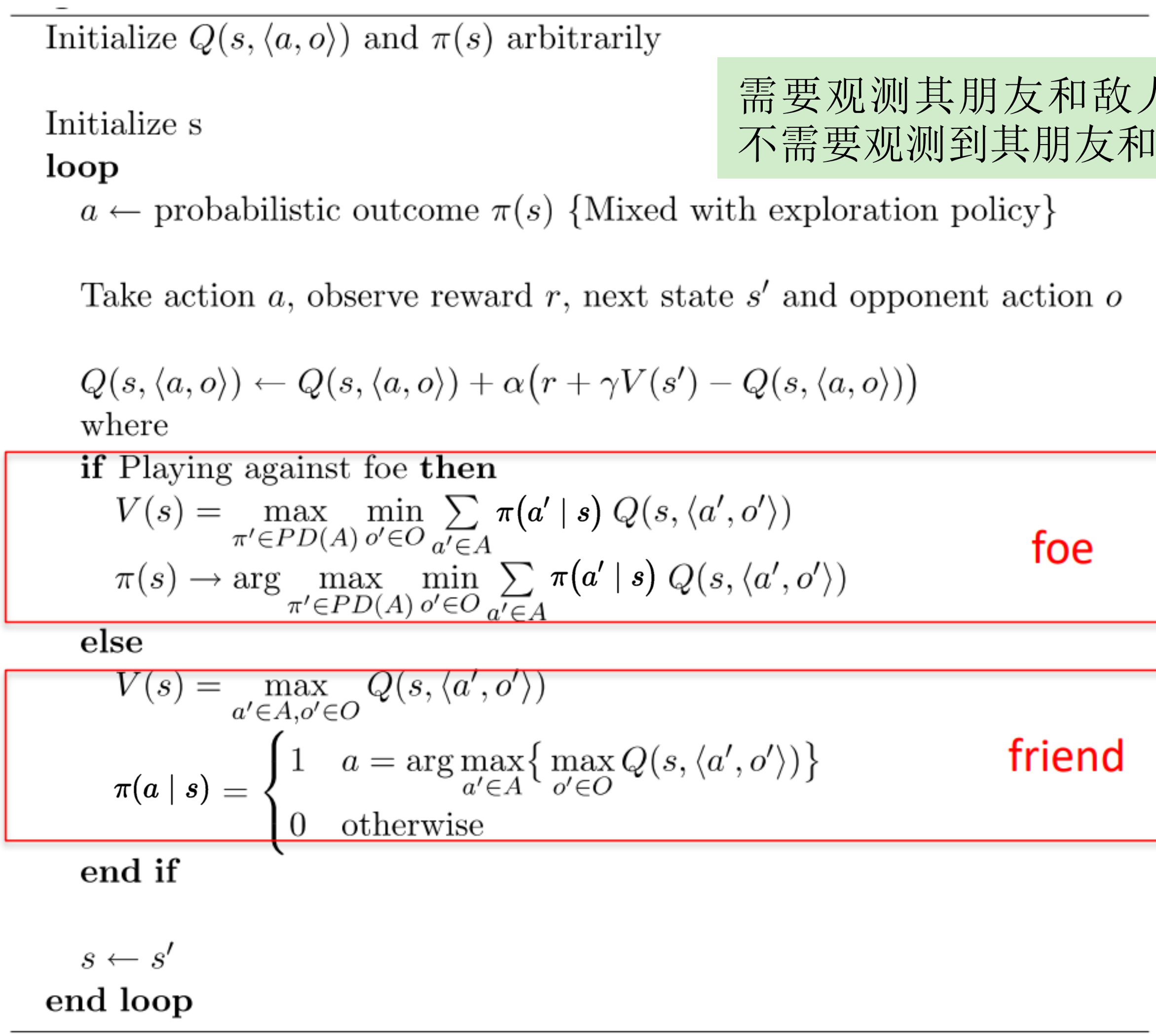
- 把其他Agent分为两类
  - Agent i的朋友：共同合作以使得Agent i的回报最大化
  - Agent i的敌人：共同合作以使得Agent i的回报最小化

- 把一个n人一般和随机博弈看作具有扩展动作集的双人零和博弈

- 如果所有状态和动作可无限访问，该算法可保证收敛到纳什均衡

Friend-or-Foe Q

需要观测其朋友和敌人的动作，但不需要观测到其朋友和敌人的奖励



```
Initialize $Q(s, \langle a, o \rangle)$ and $\pi(s)$ arbitrarily

Initialize s
loop
  $a \leftarrow$ probabilistic outcome $\pi(s)$ {Mixed with exploration policy}

  Take action $a$, observe reward $r$, next state $s'$ and opponent action $o$

  $Q(s, \langle a, o \rangle) \leftarrow Q(s, \langle a, o \rangle) + \alpha\big(r + \gamma V(s') - Q(s, \langle a, o \rangle)\big)$
  where
  if Playing against foe then
    $V(s) = \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(a' \mid s)\, Q(s, \langle a', o' \rangle)$
    $\pi(s) \rightarrow \arg \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(a' \mid s)\, Q(s, \langle a', o' \rangle)$
  else
    $V(s) = \max_{a' \in A, o' \in O} Q(s, \langle a', o' \rangle)$

    $\pi(a \mid s) = \begin{cases} 1 & a = \arg\max_{a' \in A}\{\max_{o' \in O} Q(s, \langle a', o' \rangle)\} \\ 0 & \text{otherwise} \end{cases}$
  end if

  $s \leftarrow s'$
end loop
```

foe

friend

Littman, Michael L. “Friend-or-foe Q-learning in general-sum games.” ICML 2001.

博弈与多Agent强化学习

- 概览
- 随机博弈
- 动态规划
- 均衡学习算法
- **最优反应学习算法**

多Agent学习算法的理想特性：合理性

- 合理性（Rationality）：如果其他Agent的策略收敛于固定策略，那么学习算法会将Agent的策略收敛到对其他Agent策略的最优反应策略
- 例：猜硬币游戏
  - 对手的合理性策略：在每局中采用50%的概率选择正面和反面
  - 对手的糟糕策略：总是选择正面
- 均衡学习算法不满足合理性
  - 如：极小极大Q学习算法、纳什Q学习算法、朋友或敌人Q学习算法
  - 在与采用糟糕策略的对手对局时，还是采用正/反面各占50%的策略
- 合理性学习算法
  - 当对手正在采用糟糕策略时，能把策略调整为糟糕策略的最优反应

多Agent学习算法的理想特性：收敛性

- 收敛性（Convergence）：学习算法必然会收敛到一个固定策略
- 定义：Agent i的学习算法能收敛到一个固定策略π，当且仅当对于任意$\epsilon > 0$，存在一个时刻$T > 0$，使得：

$$\forall t > T,\ a_i \in \mathcal{A}_i, s \in \mathcal{S}, \quad P(s,t) > 0 \Rightarrow |P(a_i \mid s,t) - \pi(s,a_i)| < \varepsilon$$

$P(s,t)$是博弈在时刻t时处在状态s的概率

$P(a_i|s,t)$是在时刻t且博弈处在状态s时算法选择动作a_i的概率

- 在一定条件下，极小极大Q学习算法、纳什Q学习算法、朋友或敌人Q学习算法均满足收敛性
- 学习算法的收敛性通常条件于其他Agent的学习算法
  - 如：其他Agent采用合理性算法或者与Agent相同的算法
  - 如果所有Agent都采用合理性算法，则它们的策略会收敛到纳什均衡

特殊情形：其他Agent的策略固定

- 对于一个Agent而言，当所有其他Agent的策略固定时，随机博弈会退化为MDP问题
  - 可以用所有其他Agent的固定策略来重新定义一个等效的MDP问题的状态转移概率和奖励函数

- 假设其他Agent的联合策略固定为π_{-i}
  - Agent i的策略为π_i
  - 那么等效MDP问题的参数为：
    - 状态转移函数T：$T(s'|s,a_i) = \sum_{a_{-i}} \pi_{-i}(a_{-i}|s)p(s'|s,a_i,a_{-i})$
    - 奖励函数R：$R_i(s,a_i) = \sum_{a_{-i}} \pi_{-i}(a_{-i}|s)r_i(s,a_i,a_{-i})$

独立学习算法

Independent Learning

- 每个Agent学习一个局部策略，可以根据局部观察做出决策
- 优点：不获取其他Agent的动作信息，避免了联合学习算法对于通信带宽的高要求，具备一定的可扩展性
- 缺点：把其他Agent视为环境的一部分，Agent之间缺乏协调机制，因此训练过程不稳定也无法保证收敛性
- 代表性的算法：独立Q学习

  Independent Q-Learning, IQL

  - 每个Agent利用单Agent的Q学习方法同时而又独立地学习各自的局部动作值函数$q_i(s, a_i)$
  - $q_i(s, a_i)$是联合动作值函数Q的一种平均映射：

$$q_i(s, a_i) = \sum_{a_{-i}} \underline{\pi_{-i}(a_{-i})} Q(s, a_i, a_{-i})$$

其他Agent选择动作a_{-i}的概率

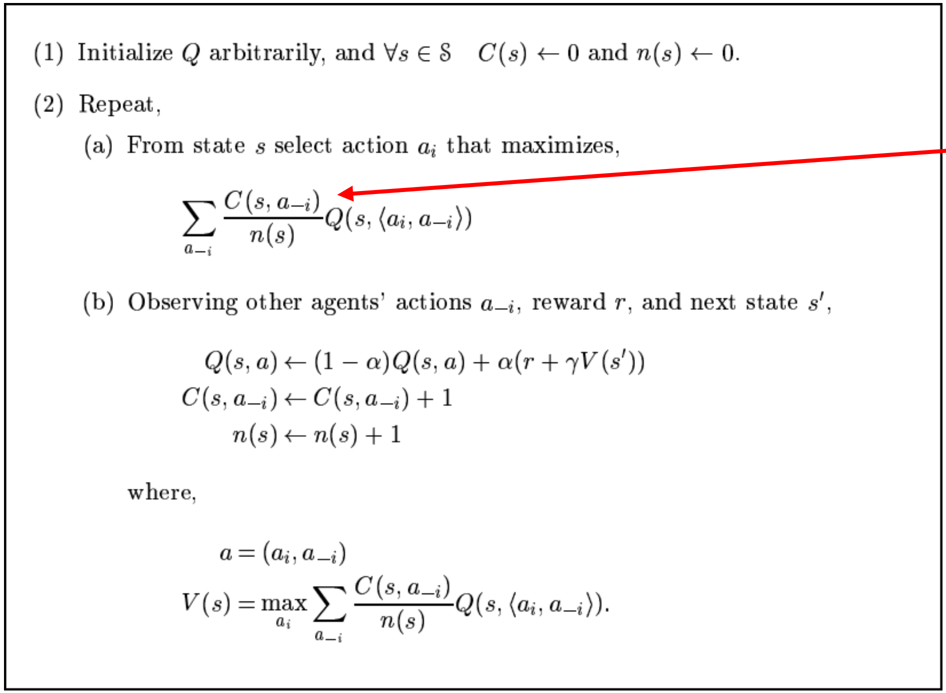
联合动作学习算法

Joint Action Learning, JAL

- 学习联合动作的Q值
  - 将所有Agent视为一个整体，称作元Agent
  - 动作空间为所有Agent的联合动作空间
  - 观察空间为所有Agent的联合观察空间（可组合成全局状态）
- 优点
  - 可以将单Agent强化学习的算法直接用在多Agent系统中
- 缺点
  - 算法在联合动作（-观察）空间中搜寻最优解，可扩展性差
  - 要求Agent间的通信信道有足够的带宽，能够把每个Agent的局部观察传送到一个集中式的控制器中
  - 在Agent间没有协调的情况下，不能保证其他Agent也在同一个学习阶段，甚至不能保证其他Agent也在学习

对手建模

- 假设其他Agent采用的是固定策略，显式地学习其他Agent的模型
- 记录状态访问次数和对手选取不同动作的次数来获得对其他Agent策略的一种推测



```
(1) Initialize $Q$ arbitrarily, and $\forall s \in \mathcal{S}$  $C(s) \leftarrow 0$ and $n(s) \leftarrow 0$.
(2) Repeat,
    (a) From state $s$ select action $a_i$ that maximizes,
            $\sum_{a_{-i}} \frac{C(s,a_{-i})}{n(s)} Q(s,\langle a_i,a_{-i}\rangle)$
    (b) Observing other agents' actions $a_{-i}$, reward $r$, and next state $s'$,
            $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r+\gamma V(s'))$
            $C(s,a_{-i}) \leftarrow C(s,a_{-i}) + 1$
            $n(s) \leftarrow n(s) + 1$
        where,
            $a = (a_i, a_{-i})$
            $V(s) = \max_{a_i} \sum_{a_{-i}} \frac{C(s,a_{-i})}{n(s)} Q(s,\langle a_i,a_{-i}\rangle)$.
```

对手建模是把其他Agent看成了一个整体，它们有执行联合动作的能力，并能获得关于它们的统计信息

- JAL和对手建模在本质上是相同的
  - JAL：集中式求解，假设其他Agent的策略给定
  - 对手建模：分布式求解，不知道对手的当前策略，但可以通过观测到的动作获得它们的一个估计

课后思考5-1（不用交）

- 假设Agent 1是行玩家，Agent 2是列玩家。考虑如下零和博弈：

$$G_1 = \begin{pmatrix} 0 & 3 \\ 2 & -1 \end{pmatrix} \qquad G_2 = \begin{pmatrix} 0 & 1 \\ 4 & 3 \end{pmatrix} \qquad G = \begin{pmatrix} G_1 & 4 \\ 5 & G_2 \end{pmatrix}$$

求博弈G_1、G_2和G的价值和两个Agent在各博弈中的最优策略。

课后思考5-2（不用交）

- 假设Agent 1是行玩家，Agent 2是列玩家。考虑如下零和博弈：

$$G_1 = \begin{pmatrix} G_2 & 0 \\ 0 & G_3 \end{pmatrix}, \quad G_2 = \begin{pmatrix} G_1 & 1 \\ 1 & 0 \end{pmatrix}, \quad G_3 = \begin{pmatrix} G_1 & 2 \\ 2 & 0 \end{pmatrix}$$

求博弈G_1、G_2和G_3的价值和两个Agent在各博弈中的最优策略。

课后思考5-3（不用交）

- 假设Agent 1是行玩家，Agent 2是列玩家。考虑如下零和随机博弈：

$$G = \begin{pmatrix} 4 & 1 + (1/3)G \\ 0 & 1 + (2/3)G \end{pmatrix}$$

求该博弈的价值和两个Agent的最优策略。

课后思考5-4（不用交）

- 假设Agent 1是行玩家，Agent 2是列玩家。考虑如下有两个状态的零和随机博弈：

$$G^{(1)} = \begin{pmatrix} 2 & 2 + (1/2)G^{(2)} \\ 0 & 4 + (1/2)G^{(2)} \end{pmatrix}$$

$$G^{(2)} = \begin{pmatrix} -4 & 0 \\ -2 + (1/2)G^{(1)} & -4 + (1/2)G^{(1)} \end{pmatrix}$$

（1）该博弈在状态1和状态2的价值分别是多少？

（2）两个Agent在状态1和状态2的最优策略分别是什么？

（3）初始化$V_0 = (0, 0)$，利用值迭代求解V_2 。