博弈论与多智能体强化学习

郭宪

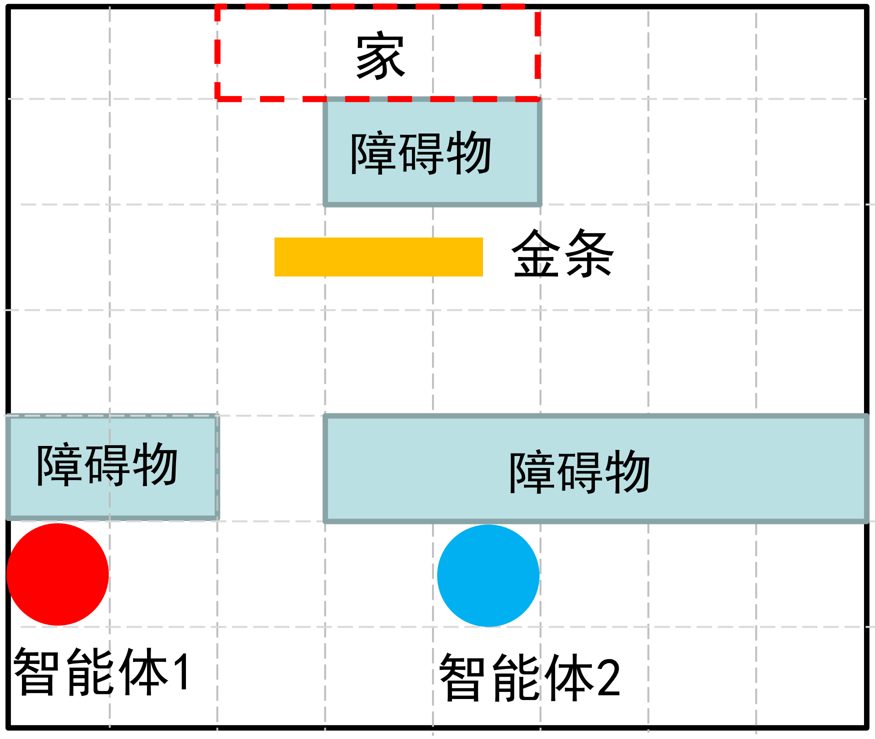
2019. 12. 01

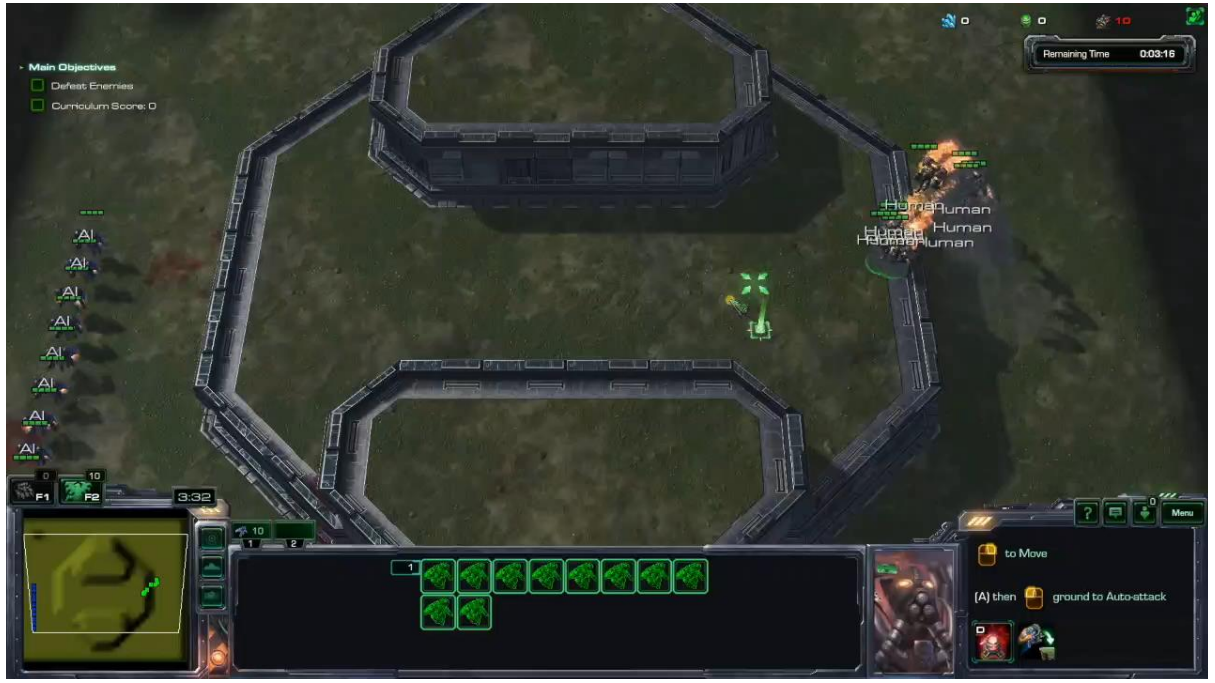
人工智能学院

College of Artificial Intelligence

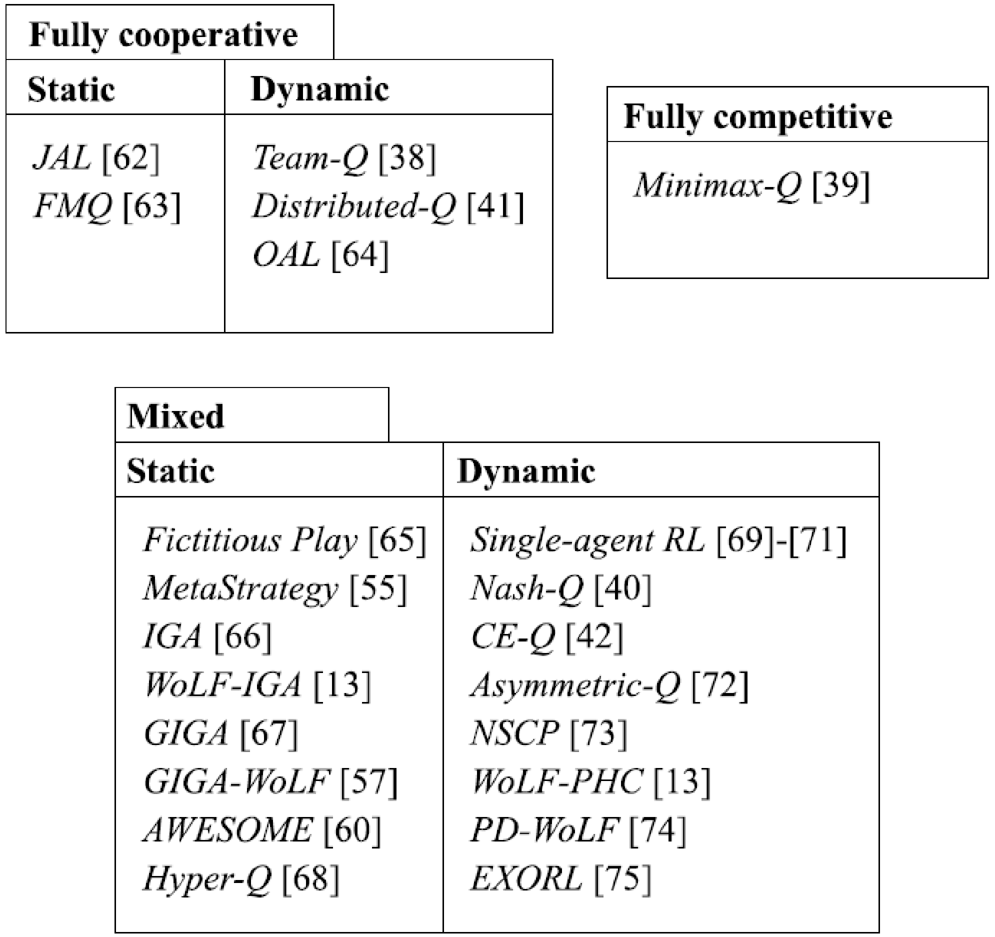
Nankai University

多智能体系统举例


家
障碍物
金条
障碍物
障碍物
智能体1
智能体2


Main Objectives
Defeat Enemies
Curriculum Score: 0
Remaining Time 0:03:16
AI
Human
to Move
(A) then ground to Auto-attack
Menu

多智能体强化学习分类



Fully cooperative	
Static	**Dynamic**
JAL [62]	*Team-Q* [38]
FMQ [63]	*Distributed-Q* [41]
	OAL [64]

Fully competitive
Minimax-Q [39]

Mixed	
Static	**Dynamic**
Fictitious Play [65]	*Single-agent RL* [69]-[71]
MetaStrategy [55]	*Nash-Q* [40]
IGA [66]	*CE-Q* [42]
WoLF-IGA [13]	*Asymmetric-Q* [72]
GIGA [67]	*NSCP* [73]
GIGA-WoLF [57]	*WoLF-PHC* [13]
AWESOME [60]	*PD-WoLF* [74]
Hyper-Q [68]	*EXORL* [75]

按照任务的类型可以分为三类：

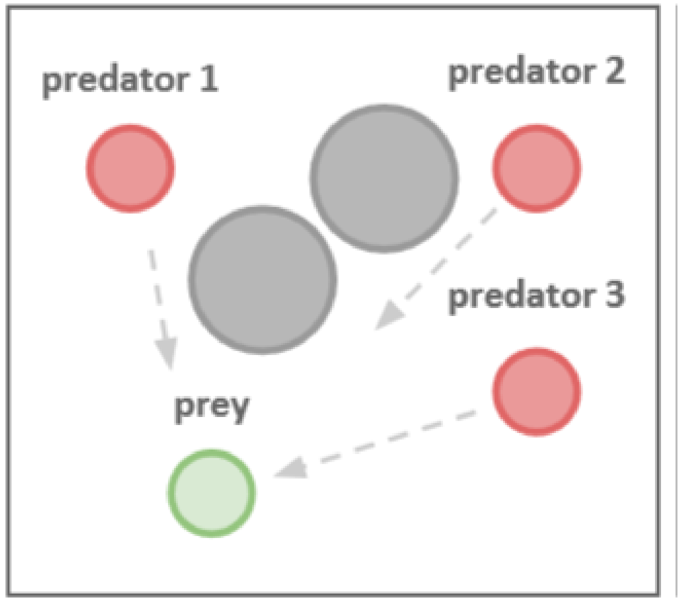
1. 完全协作多智能体系统
2. 完全竞争多智能体系统
3. 混合策略多智能体系统

L. Busoniu, R. Babuska, B.D. Schutter
A Comprehensive Survey of Multi-Agent Reinforcement Learning，2008

强化学习算法用于多智能体系统的挑战

1. 马尔科夫决策过程假设环境是静态的。从单智体的视角看，其他智能体都是动态的环境。

2. 智能体之间是竞争关系时，最优策略不存在，需要求均衡解



联合博弈论和强化学习

博弈论：无状态转移的策略博弈（多智能体之间的交互，如协作和竞争）

马尔科夫博弈：涉及状态转移的策略博弈

博弈论基本分类

1. 根据参与人行动的先后顺序：

静态博弈：参与人同时选择行动，如猜丁壳等。常用标准型表述（normal form）：矩阵博弈

动态博弈：参与人的动作有先后顺序，且后者能观察到前者所选择的行动，如围棋。常用扩展型表述(extensive form): 博弈树

2. 根据参与人对特征，战略空间是否已知：

完美信息博弈：参与人对相关信息已知。如棋类游戏。

不完美信息博弈：参与人对相关信息并不完全知道，如牌类游戏。

完美信息静态博弈，完美信息动态博弈，不完美信息静态博弈，不完美信息动态博弈

完美信息静态博弈的基本概念

1. 参与者：参与博弈的智能体 i

2. 动作空间：参与博弈的智能体 i的动作记为a_i，$A_i = \{a_i\}$ 表示可供 i 选择的所用行动的集合。

   联合动作为：n个参与者的行动的有序集，$a = (a_1, \cdots, a_n)$

3. 策略：参与者在当前情况下动作空间上的分布。我们用 π_i 表示第 i 个玩家的策略。

   纯策略（pure policy）：玩家 i 选择某一行为的概率为 1，而其他动作选择概率为 0

   混合策略（mixed policy）：玩家 i 选择某一行为的概率小于1

   联合策略为：n个参与者的策略的有序集，$\pi = (\pi_1, \cdots, \pi_n)$

完美信息静态博弈的基本概念

4. 回报函数：参与者在联合行为 $a=(a_1,\cdots,a_n)$ 下所获得的回报，记为 $Q_i(a_1,\cdots,a_n)$

5. 值函数：参与者在联合策略下的值函数

$$V_i(\pi_1,\cdots,\pi_n)=\sum_{a_1,\cdots,a_n\in A_1\times\cdots\times A_n} Q_i(a_1,\cdots,a_n)\pi_1(a_1)\cdots\pi_n(a_n)$$

完美信息静态博弈举例

例子：囚徒困境

$$\begin{array}{c|cc} & \text{坦白} & \text{抵赖} \\ \hline \text{坦白} & -3,-3 & 0,-5 \\ \text{抵赖} & -5,0 & -1,-1 \end{array}$$

	a1	a2
a1	(1,-1)	(-1,1)
a2	(-1,1)	(1,-1)

零和博弈

	a1	a2
a1	**(2,1)**	(0,0)
a2	(0,0)	**(1,2)**

性别之战

参与者：两个囚徒

动作空间：{坦白，抵赖}

回报函数：如上回报矩阵

博弈论的解？

策略：两个囚徒如何是选择坦白还是抵赖

博弈的最优解

在博弈论中，玩家的奖励函数依赖于其他玩家的策略。

因此，玩家不能只简单地最大化个人的收益，因为玩家往往不能同时实现个人收益最大。

	a1	a2
a1	**(2,1)**	(0,0)
a2	(0,0)	**(1,2)**

性别之战

Best response: 当其他玩家策略固定时，最好的策略:

$$R_k(\sigma_{-k} \cup \sigma_k^*) \geq R_k(\sigma_{-k} \cup \sigma_k^{'}) \ \forall \sigma_k^{'} \in \mu(A_k)$$

纳什均衡策略：当所有玩家都是best response时，称为纳什均衡。

$$\begin{array}{c|cc} & \text{坦白} & \text{抵赖} \\ \hline \text{坦白} & -3,-3 & 0,-5 \\ \text{抵赖} & -5,0 & -1,-1 \end{array}$$

完美信息静态博弈：纳什均衡

纳什均衡策略（Nash equilibrium）：

1. 个人是理性的
2. 没有玩家能通过单独偏离他的策略可以改善自己的回报

$$\begin{array}{c|cc} & \text{坦白} & \text{抵赖} \\ \text{坦白} & -3,-3 & 0,-5 \\ \text{抵赖} & -5,0 & -1,-1 \end{array}$$

用 π_i^* 表示纳什均衡策略，则纳什均衡策略应该满足：

$$V_i(\pi_1^*,\cdots,\pi_i^*,\cdots,\pi_n^*) \geqslant V_i(\pi_1^*,\cdots,\pi_i,\cdots,\pi_n^*)$$

囚徒困境的纳什均衡策略是什么？

“坦白，坦白” 对于囚徒来说是纳什均衡策略

双人零和博弈：纳什均衡

矩阵博弈：双人联合行为对所对应的元素构成矩阵，因此称为矩阵博弈

双人策略为：(π_1, π_2)

对于玩家 i 来说，值函数为： $V_i = \pi_1 R_i \pi_2^T$

双人矩阵博弈的纳什均衡是指策略对(π_1^*, π_2^*)，满足：

$$V_i(\pi_i^*, \pi_{-i}^*) \geqslant V_i(\pi_i, \pi_{-i}^*)$$

求解双人零和矩阵博弈中的纳什均衡等价于求解下列方程的最小解：

$$\max_{\pi_i} \min_{\pi_{-i}} \sum_{a_i \in A_i} R_i^T \pi_i(a_i)$$

双人零和博弈：纳什均衡的求解

求解双人零和矩阵博弈中的纳什均衡等价于求解下列方程的最小解：

$$\max_{\pi_i}\min_{\pi_{-i}}\sum_{a_i\in A_i}R_i^T\pi_i(a_i)$$

转化为线性规划的方法，利用单纯型法求解。

给定一个 $2\times 2_1$ 的矩阵， $$R_1=\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22}\end{bmatrix}, R_2=-R_1$$

线性优化问题：

$$\begin{aligned} &\max\ V_1 \\ s.t.\quad & r_{11}p_1+r_{21}p_2\geqslant V_1 \\ & r_{12}p_1+r_{22}p_2\geqslant V_1 \\ & p_1+p_2=1 \\ & p_j\geqslant 0,\ j=1,2 \end{aligned}$$

现实的多智能体涉及到转态的转移！如何去描述？马尔科夫博弈

随机博弈（马尔科夫博弈）

Stochastic game可以用一个元组 $(n,\ S,\ A_1,\cdots,A_n,T,\gamma,R_1,\cdots,R_n)$ 来描述。其中 n 为玩家的个数。

联合动作空间：

$$\vec{A}=(A_1,\cdots,A_n)$$

联合状态转移概率：

$$T:S\times A_1\times A_2\times\cdots\times A_n\times S\rightarrow[0,1]$$

联合立即回报：

$$R_i:S\times A_1\times\cdots\times A_n\times S\rightarrow[0,1]$$

局部策略：

$$h_i:S\times A_i\rightarrow[0,1]$$

联合策略：

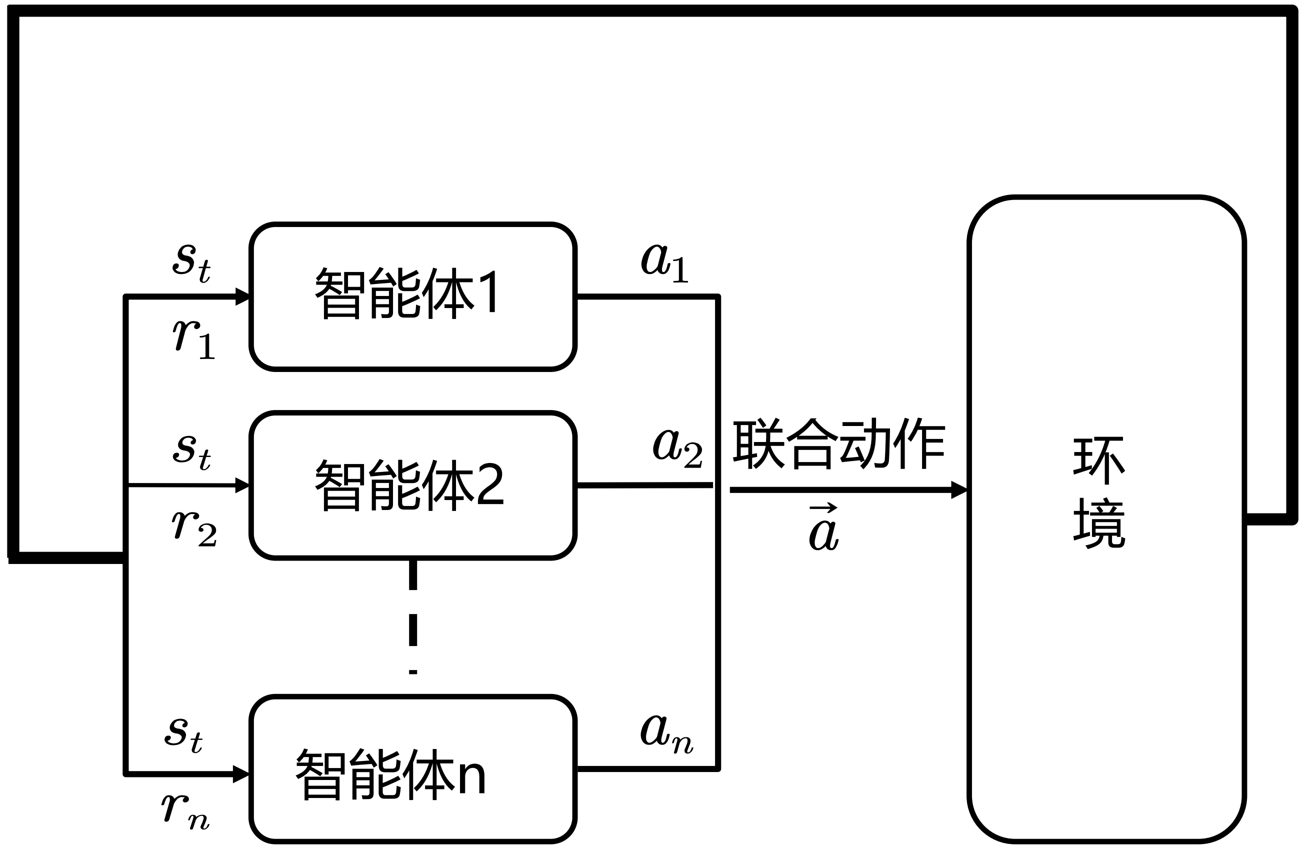
$$\vec{h}=(h_1,\cdots,h_n)$$

每个智能体的行为值函数依赖于联合动作：

$$Q_i^h(s,a_1,\cdots,a_i,\cdots,a_n)=R(s,a_1,\cdots,a_i,\cdots,a_n)$$
$$+\gamma\sum_{s'}T(s,a_1,\cdots,a_i,\cdots,a_n)V(s')$$

联合策略下的行为值函数

多智能体强化学习



多智能体强化学习：回报的定义

对于智能体 i 其值函数为：

$$V_k^{\pi}(s) = E^{\pi}\left\{\sum_{t=0}^{\infty} \gamma^t r_k(t+1) \mid s(0) = s\right\}$$

其中 $\pi=(\pi_1,\cdots,\pi_n)$

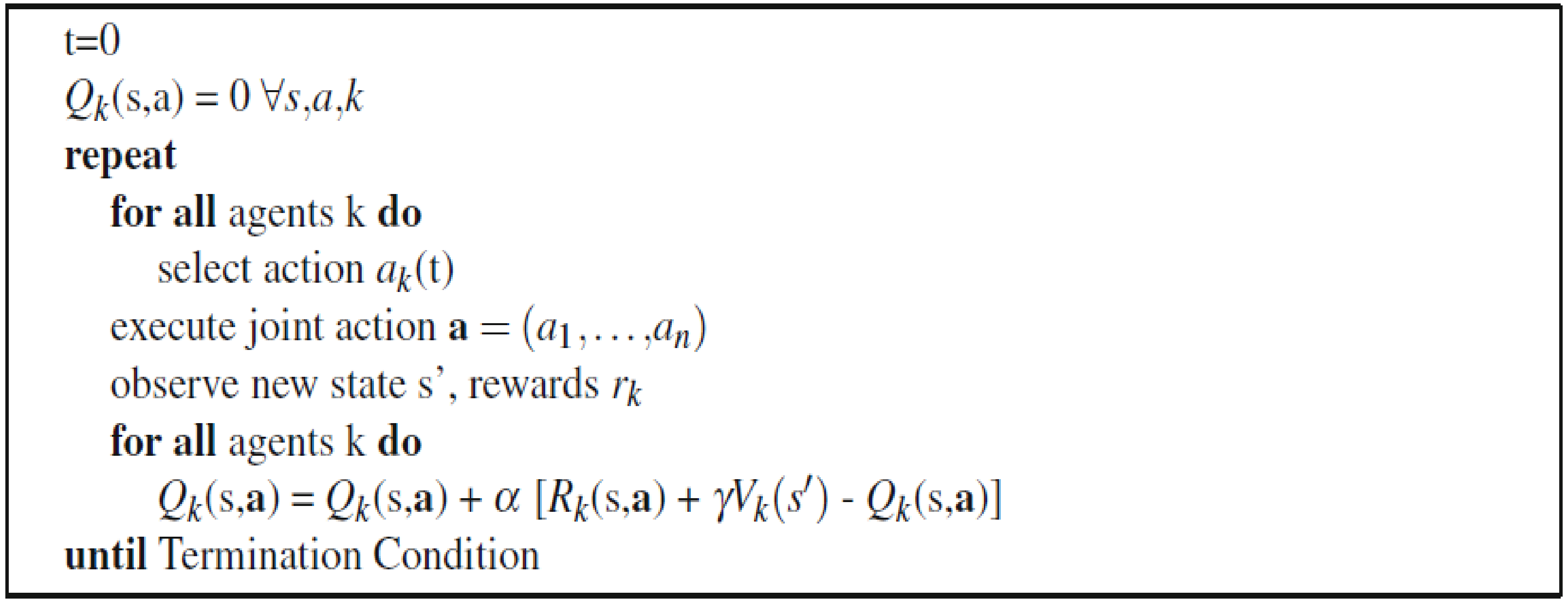
多智能体的值函数的大小不仅依赖于自己的策略 π_i ，还依赖于其他智能体的策略

一般情况下，同时最大化每个智能体的目标函数是不可能的，因为智能体可能有相矛盾的目标。

最好反映和纳什均衡的概念被扩展到马尔科夫博弈中。

多智能体强化学习的Qlearning算法

基于值函数的多智能体强化学习算法的一般框架：



```
t=0
$Q_k(s,a) = 0 \ \forall s,a,k$
repeat
    for all agents k do
        select action $a_k(t)$
    execute joint action $\mathbf{a} = (a_1, \ldots, a_n)$
    observe new state s', rewards $r_k$
    for all agents k do
        $Q_k(s,\mathbf{a}) = Q_k(s,\mathbf{a}) + \alpha\,[R_k(s,\mathbf{a}) + \gamma V_k(s') - Q_k(s,\mathbf{a})]$
until Termination Condition
```

Algorithm 23. Multi-Agent Q-Learning

多智能体强化学习：值迭代

与单智能体强化学习的区别：

1. 估计的值函数Q(s, a)不同

对于每个智能体，估计联合动作值函数：$Q(s,\vec{a})$

对于智能体i，估计依赖于动作 a_{-k} 的行为值函数：

因此，智能体i在更新行为值函数的时候，必须观测其他智能体的动作

2. 对值函数估计的时候，需要用到下一个状态的最大行为值函数

多智能体强化学习的Qlearning算法

如何得到下一个状态处的值函数：

$$Q_k(s,\vec{a}) = Q_k(s,\vec{a}) + \alpha\left[R_k(s,\vec{a}) + \gamma V_k(s') - Q_k(s,\vec{a})\right]$$

1. 对手建模的方法

其他智能体可能的联合动作频率：

$$F(s,a_{-k}) = \frac{c(s,a_{-k})}{\sum_{a'_{-k}\in A} n(s,a'_{-k})}$$

计算值函数：

$$V_k(s) = \max_{a_k} Q(s,a_k) = \sum_{a_{-k}\in A_{-k}} F(s,a_{-k})Q(s,a_k,a_{-k})$$

多智能体强化学习的Qlearning算法

如何得到下一个状态处的值函数：

$$Q_k(s,\vec{a}) = Q_k(s,\vec{a}) + \alpha\left[R_k(s,\vec{a}) + \gamma V_k(s') - Q_k(s,\vec{a})\right]$$

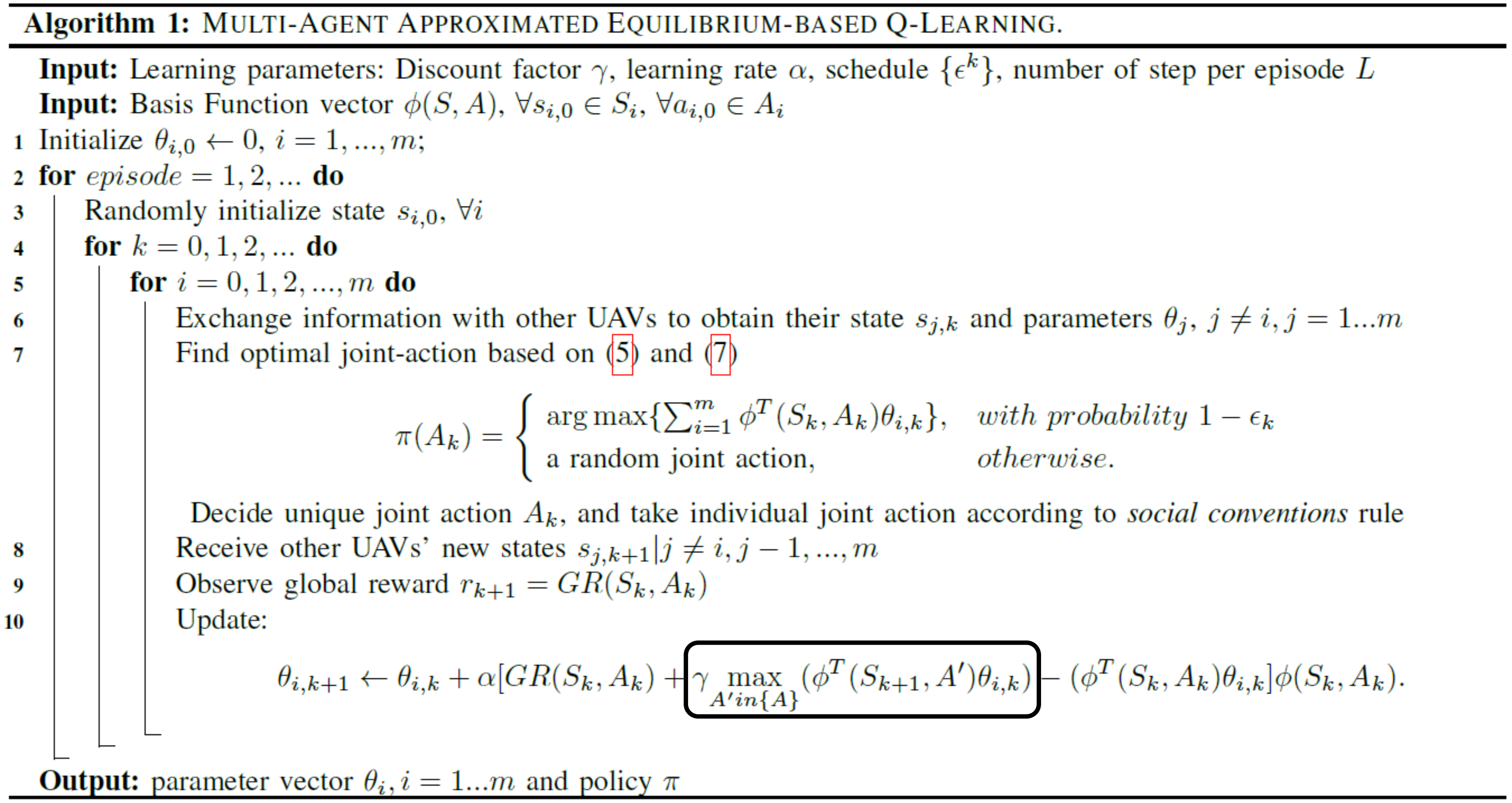
2. 假设对手按照某种规则选择动作，如零和博弈中的最小最大原则

$$V_k(s) = \min_{a'} \max_{\sigma\in\mu(A)} \sum_{a\in A} \sigma(a) Q(s,a,a')$$

3. 假设智能体在下一个状态处都执行均衡策略，Nash-Q/相关均衡/Stackelberg均衡.

$$\mathrm{V}_k(s) = Nash_k(s, Q_1, \cdots, Q_n)$$

1. 完全协作强化学习算法



```
Algorithm 1: MULTI-AGENT APPROXIMATED EQUILIBRIUM-BASED Q-LEARNING.
   Input: Learning parameters: Discount factor $\gamma$, learning rate $\alpha$, schedule $\{\epsilon^k\}$, number of step per episode $L$
   Input: Basis Function vector $\phi(S,A)$, $\forall s_{i,0} \in S_i$, $\forall a_{i,0} \in A_i$
1  Initialize $\theta_{i,0} \leftarrow 0,\ i = 1, ..., m$;
2  for $episode = 1, 2, ...$ do
3      Randomly initialize state $s_{i,0}$, $\forall i$
4      for $k = 0, 1, 2, ...$ do
5          for $i = 0, 1, 2, ..., m$ do
6              Exchange information with other UAVs to obtain their state $s_{j,k}$ and parameters $\theta_j$, $j \neq i, j = 1...m$
7              Find optimal joint-action based on ($\boxed{5}$) and ($\boxed{7}$)
                   $\pi(A_k) = \begin{cases} \arg\max\{\sum_{i=1}^{m} \phi^T(S_k, A_k)\theta_{i,k}\}, & with\ probability\ 1 - \epsilon_k \\ \text{a random joint action}, & otherwise. \end{cases}$
               Decide unique joint action $A_k$, and take individual joint action according to *social conventions* rule
8              Receive other UAVs' new states $s_{j,k+1} | j \neq i, j - 1, ..., m$
9              Observe global reward $r_{k+1} = GR(S_k, A_k)$
10             Update:
                   $\theta_{i,k+1} \leftarrow \theta_{i,k} + \alpha[GR(S_k, A_k) + \boxed{\gamma \max_{A' in\{A\}} (\phi^T(S_{k+1}, A')\theta_{i,k})} - (\phi^T(S_k, A_k)\theta_{i,k}]\phi(S_k, A_k).$
   Output: parameter vector $\theta_i, i = 1...m$ and policy $\pi$
```

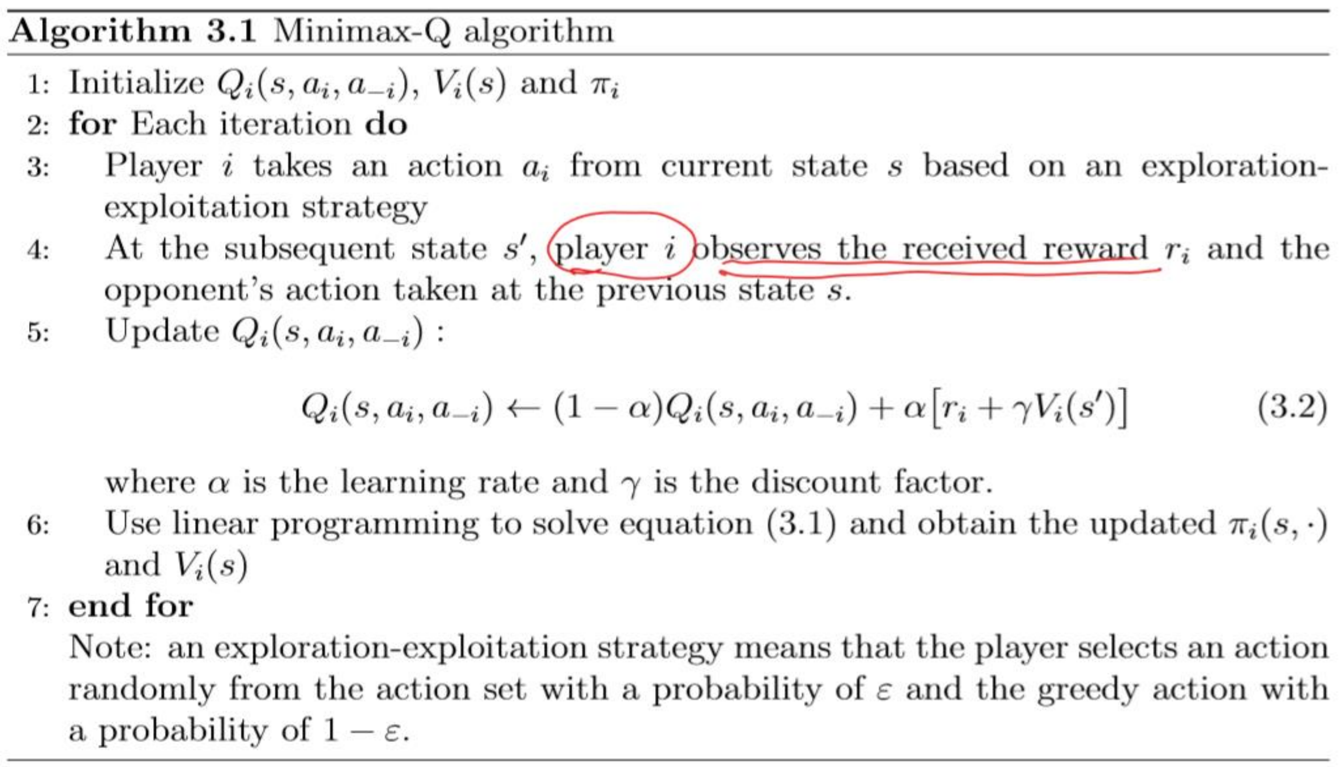
Cooperative and Distributed Reinforcement Learning of Drones for Field Coverage

2. 完全竞争：MinimaxQ learning算法

定义值函数：

$$V_i^*(s) = \max_{\pi_i(s,\cdot)} \min_{a_{-i} \in A_{-i}} \sum_{a_i \in A_i} Q_i^*(s,a_i,a_{-i})\pi_i(s,a_i) \quad (i=1,2)$$

如果 $Q_i^*(s,a_i,a_{-i})$ 是已知的，那么可以利用线性规划的方法来求解。但是对于一个问题，$Q_i^*(s,a_i,a_{-i})$ 需要学习



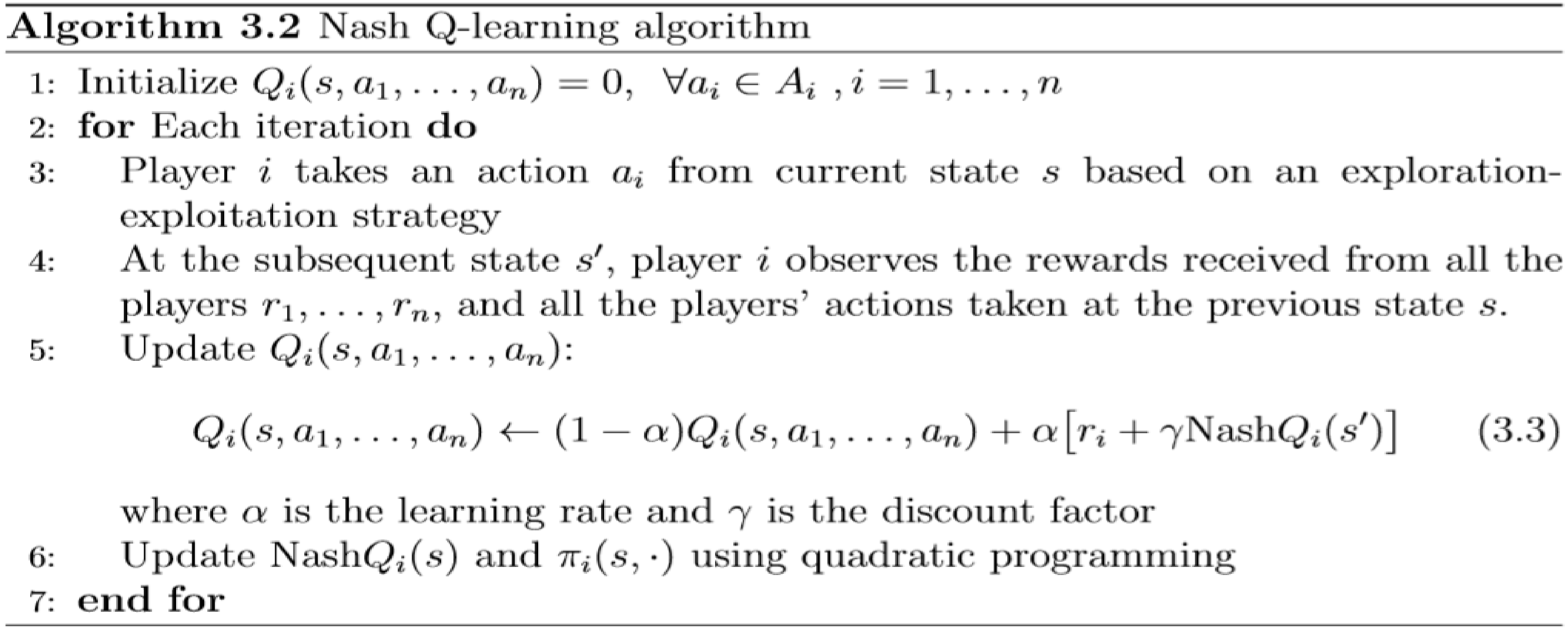
Algorithm 3.1 Minimax-Q algorithm

```
1: Initialize $Q_i(s,a_i,a_{-i})$, $V_i(s)$ and $\pi_i$
2: for Each iteration do
3:     Player $i$ takes an action $a_i$ from current state $s$ based on an exploration-exploitation strategy
4:     At the subsequent state $s'$, player $i$ observes the received reward $r_i$ and the opponent's action taken at the previous state $s$.
5:     Update $Q_i(s,a_i,a_{-i})$ :
           $Q_i(s,a_i,a_{-i}) \leftarrow (1-\alpha)Q_i(s,a_i,a_{-i}) + \alpha\left[r_i + \gamma V_i(s')\right]$    (3.2)
       where $\alpha$ is the learning rate and $\gamma$ is the discount factor.
6:     Use linear programming to solve equation (3.1) and obtain the updated $\pi_i(s,\cdot)$ and $V_i(s)$
7: end for
```

Note: an exploration-exploitation strategy means that the player selects an action randomly from the action set with a probability of ε and the greedy action with a probability of $1-\varepsilon$.

3. 混合任务：纳什Q学习

纳什Q学习，与Minimax Q学习的区别是利用二次规划来更新行为值函数并找到均衡策略。收敛的条件为：每个状态处的博弈都有一个全局最优点或者鞍点。该条件太严格，在实际中很难达到。



Algorithm 3.2 Nash Q-learning algorithm

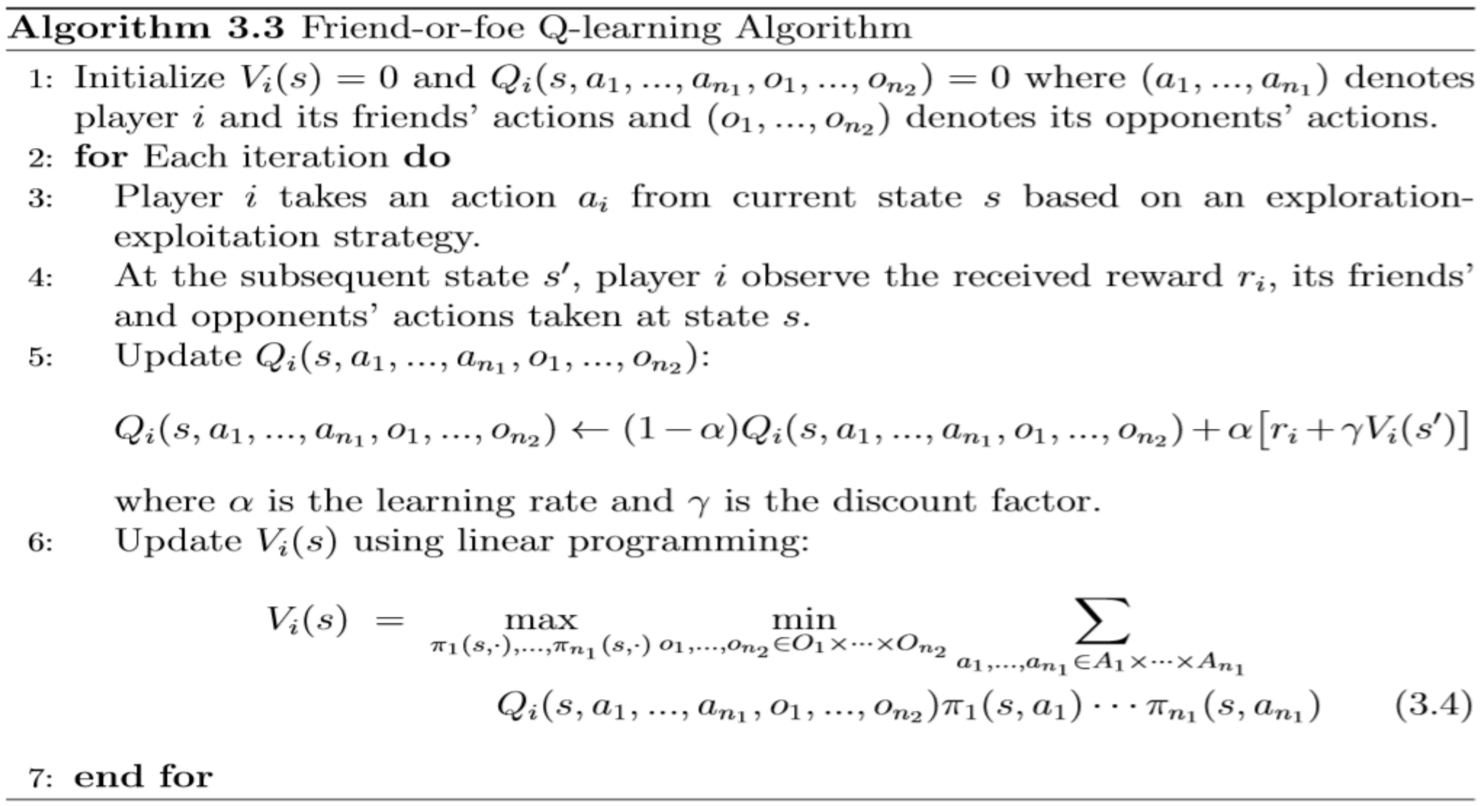
```
1: Initialize $Q_i(s, a_1, \ldots, a_n) = 0, \ \forall a_i \in A_i \ , i = 1, \ldots, n$
2: for Each iteration do
3:     Player $i$ takes an action $a_i$ from current state $s$ based on an exploration-
       exploitation strategy
4:     At the subsequent state $s'$, player $i$ observes the rewards received from all the
       players $r_1, \ldots, r_n$, and all the players' actions taken at the previous state $s$.
5:     Update $Q_i(s, a_1, \ldots, a_n)$:

           $Q_i(s, a_1, \ldots, a_n) \leftarrow (1 - \alpha) Q_i(s, a_1, \ldots, a_n) + \alpha \left[ r_i + \gamma \mathrm{Nash} Q_i(s') \right]$      (3.3)

       where $\alpha$ is the learning rate and $\gamma$ is the discount factor
6:     Update $\mathrm{Nash}Q_i(s)$ and $\pi_i(s, \cdot)$ using quadratic programming
7: end for
```

4. 混合任务：朋友或敌人Q学习

对于一般和随机博弈，对于玩家i，其他玩家或者是朋友或者是敌人。玩家的朋友被当成是合作伙伴，一起工作共同最大化玩家的值；而对于玩家的敌人，则敌人一起令玩家i的回报最小。N玩家一般和随机博弈能被认为是一个行为集扩展的两玩家零和博弈。



Algorithm 3.3 Friend-or-foe Q-learning Algorithm

```
1: Initialize $V_i(s) = 0$ and $Q_i(s, a_1, ..., a_{n_1}, o_1, ..., o_{n_2}) = 0$ where $(a_1, ..., a_{n_1})$ denotes
   player $i$ and its friends' actions and $(o_1, ..., o_{n_2})$ denotes its opponents' actions.
2: for Each iteration do
3:     Player $i$ takes an action $a_i$ from current state $s$ based on an exploration-
       exploitation strategy.
4:     At the subsequent state $s'$, player $i$ observe the received reward $r_i$, its friends'
       and opponents' actions taken at state $s$.
5:     Update $Q_i(s, a_1, ..., a_{n_1}, o_1, ..., o_{n_2})$:
           $Q_i(s, a_1, ..., a_{n_1}, o_1, ..., o_{n_2}) \leftarrow (1-\alpha)Q_i(s, a_1, ..., a_{n_1}, o_1, ..., o_{n_2}) + \alpha[r_i + \gamma V_i(s')]$
       where $\alpha$ is the learning rate and $\gamma$ is the discount factor.
6:     Update $V_i(s)$ using linear programming:
           $V_i(s) = \max_{\pi_1(s,\cdot),...,\pi_{n_1}(s,\cdot)} \min_{o_1,...,o_{n_2} \in O_1 \times \cdots \times O_{n_2}} \sum_{a_1,...,a_{n_1} \in A_1 \times \cdots \times A_{n_1}} Q_i(s, a_1, ..., a_{n_1}, o_1, ..., o_{n_2})\pi_1(s, a_1) \cdots \pi_{n_1}(s, a_{n_1})$   (3.4)
7: end for
```

4. 混合任务：WoLF 策略爬山算法

前面三种强化学习算法需要智能体维持它们的Q函数。每个玩家的Q函数包括其他玩家的信息。

WoLF-PHC算法仅仅需要每个玩家自己的动作，约减空间从 $|S|\cdot|A|^n$ 到 $|S|\cdot|A|$

该算法组合了两个方法：

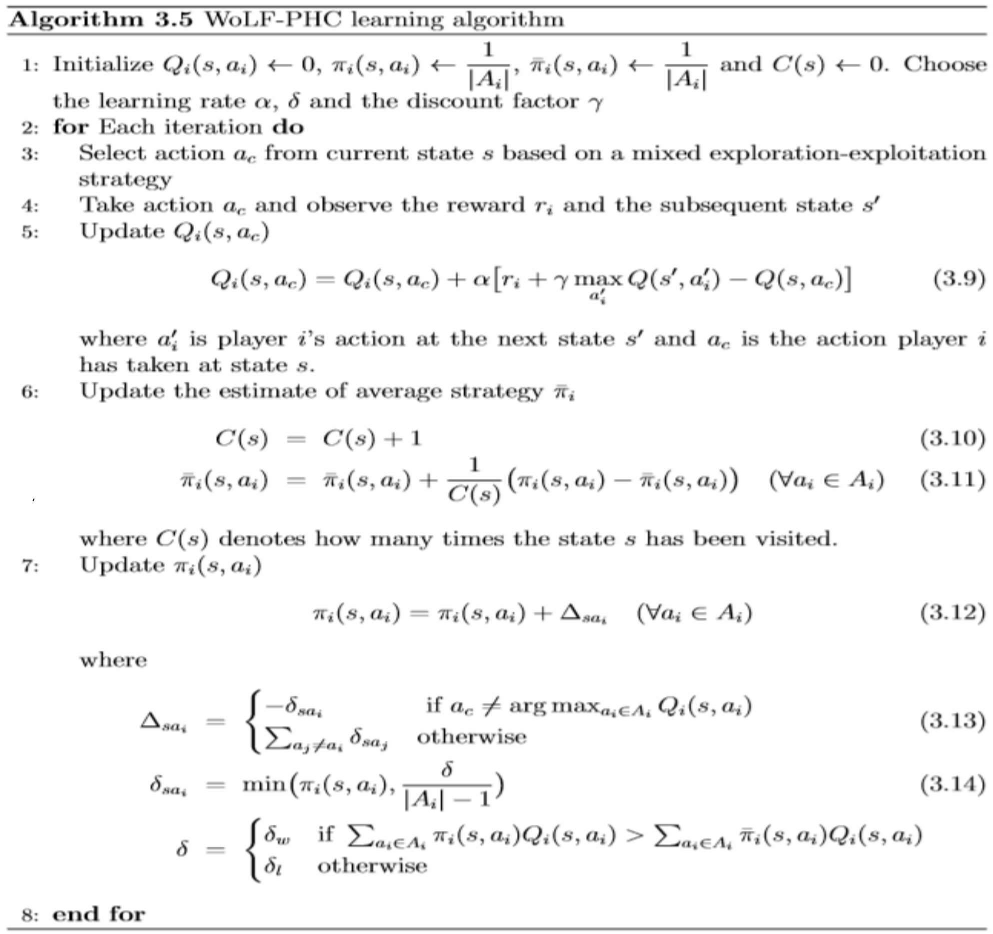
“Win or Learn Fast”：当学习者做得比期望的差很多时，需要快速学习（快速调整权重）；

当做得比期望的好时，谨慎学习（谨慎调整权重）。

Policy hill-climbing: 策略爬山算法，该算法满足合理性要求。

策略爬山算法仅仅能保证在静态环境下对于单智能体收敛到最优策略。

4. 混合任务： WoLF 策略爬山算法



Algorithm 3.5 WoLF-PHC learning algorithm

```
1: Initialize $Q_i(s,a_i) \leftarrow 0$, $\pi_i(s,a_i) \leftarrow \frac{1}{|A_i|}$, $\bar{\pi}_i(s,a_i) \leftarrow \frac{1}{|A_i|}$ and $C(s) \leftarrow 0$. Choose
   the learning rate $\alpha$, $\delta$ and the discount factor $\gamma$
2: for Each iteration do
3:     Select action $a_c$ from current state $s$ based on a mixed exploration-exploitation
       strategy
4:     Take action $a_c$ and observe the reward $r_i$ and the subsequent state $s'$
5:     Update $Q_i(s,a_c)$
           $Q_i(s,a_c) = Q_i(s,a_c) + \alpha[r_i + \gamma \max_{a_i'} Q(s',a_i') - Q(s,a_c)]$    (3.9)
       where $a_i'$ is player $i$'s action at the next state $s'$ and $a_c$ is the action player $i$
       has taken at state $s$.
6:     Update the estimate of average strategy $\bar{\pi}_i$
           $C(s) = C(s) + 1$    (3.10)
           $\bar{\pi}_i(s,a_i) = \bar{\pi}_i(s,a_i) + \frac{1}{C(s)}(\pi_i(s,a_i) - \bar{\pi}_i(s,a_i)) \quad (\forall a_i \in A_i)$    (3.11)
       where $C(s)$ denotes how many times the state $s$ has been visited.
7:     Update $\pi_i(s,a_i)$
           $\pi_i(s,a_i) = \pi_i(s,a_i) + \Delta_{sa_i} \quad (\forall a_i \in A_i)$    (3.12)
       where
           $\Delta_{sa_i} = \begin{cases} -\delta_{sa_i} & \text{if } a_c \neq \arg\max_{a_i \in A_i} Q_i(s,a_i) \\ \sum_{a_j \neq a_i} \delta_{sa_j} & \text{otherwise} \end{cases}$    (3.13)
           $\delta_{sa_i} = \min(\pi_i(s,a_i), \frac{\delta}{|A_i|-1})$    (3.14)
           $\delta = \begin{cases} \delta_w & \text{if } \sum_{a_i \in A_i} \pi_i(s,a_i)Q_i(s,a_i) > \sum_{a_i \in A_i} \bar{\pi}_i(s,a_i)Q_i(s,a_i) \\ \delta_l & \text{otherwise} \end{cases}$
8: end for
```

5. 基于微分对策略

运动学方程：

$$\begin{cases} \dot{x} = f(x) + g(x)u(x) + k(x)d(x) \\ y = h(x) \\ z = [y^T \quad u^T]^T \end{cases}$$

性能指标函数： $J(x(0),u,d) = \int_0^{\infty} (Q(x) + u^T Ru - \gamma^2 \|d\|^2) dt \equiv \int_0^{\infty} r(x,u,d) dt$

定义双人-零和微分对策： $V^*(x(0)) = \min\limits_u \max\limits_d J(x(0),u,d)$

得到Hamilton-Jacobi-Isaacs等式：

$$0 = Q(x) + \nabla V^T(x) f(x) - \frac{1}{4} \nabla V^T(x) g(x) R^{-1} g^T(x) \nabla V(x) + \frac{1}{4\gamma^2} \nabla V^T(x) k k^T \nabla V(x)$$

第九次作业

1. 分组阅读多智能体强化学习论文

01 Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments

02 Actor-Attention-Critic for Multi-Agent Reinforcement Learning

03 Counterfactual Multi-Agent Policy Gradients

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

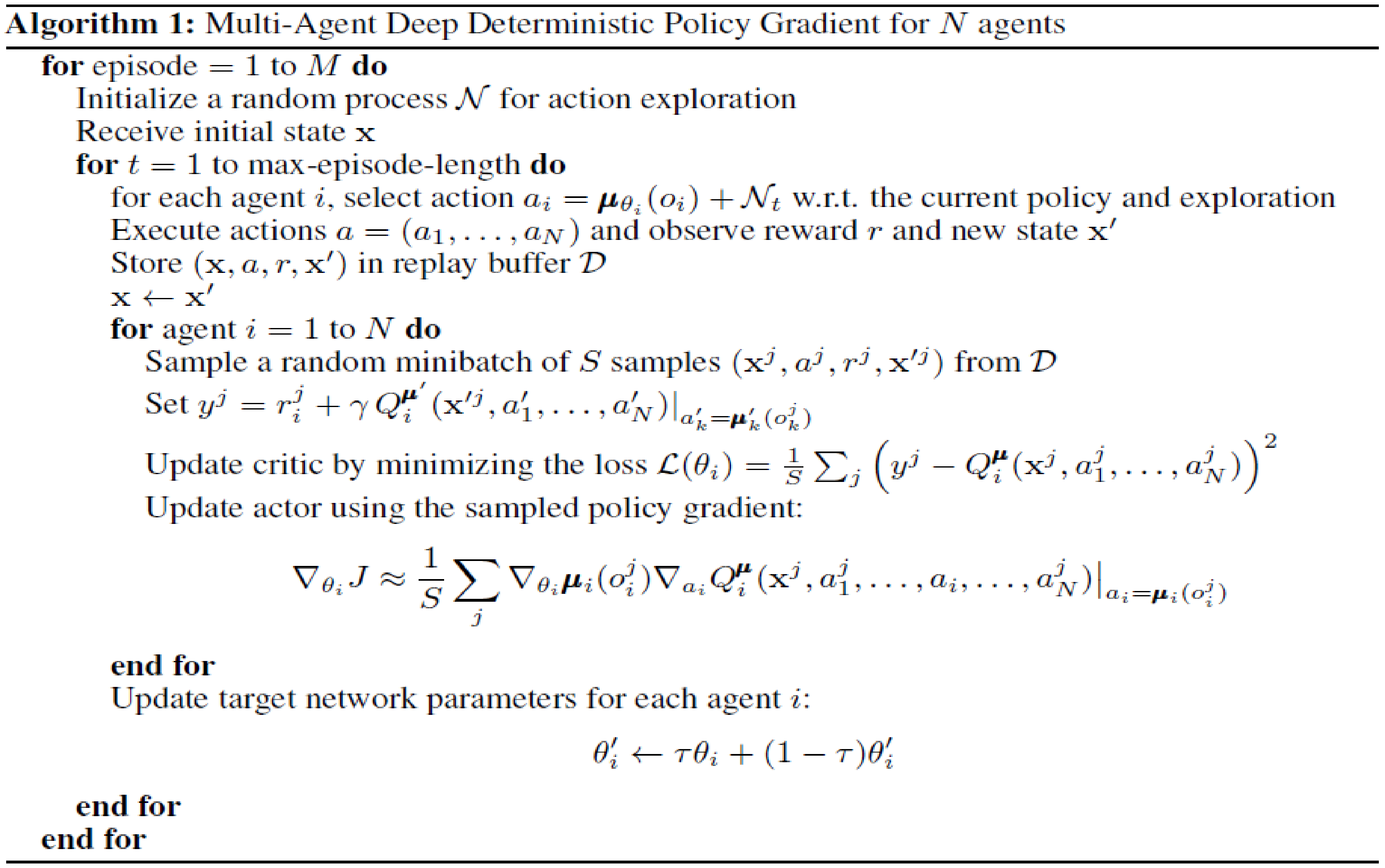
05 Grandmaster level in StarCraft II using multi-agent reinforcement learning`

06 Superhuman AI for multiplayer poker

07 Let's Play Mahjong!

08 https://openai.com/blog/openai-five/

6. MADDPG

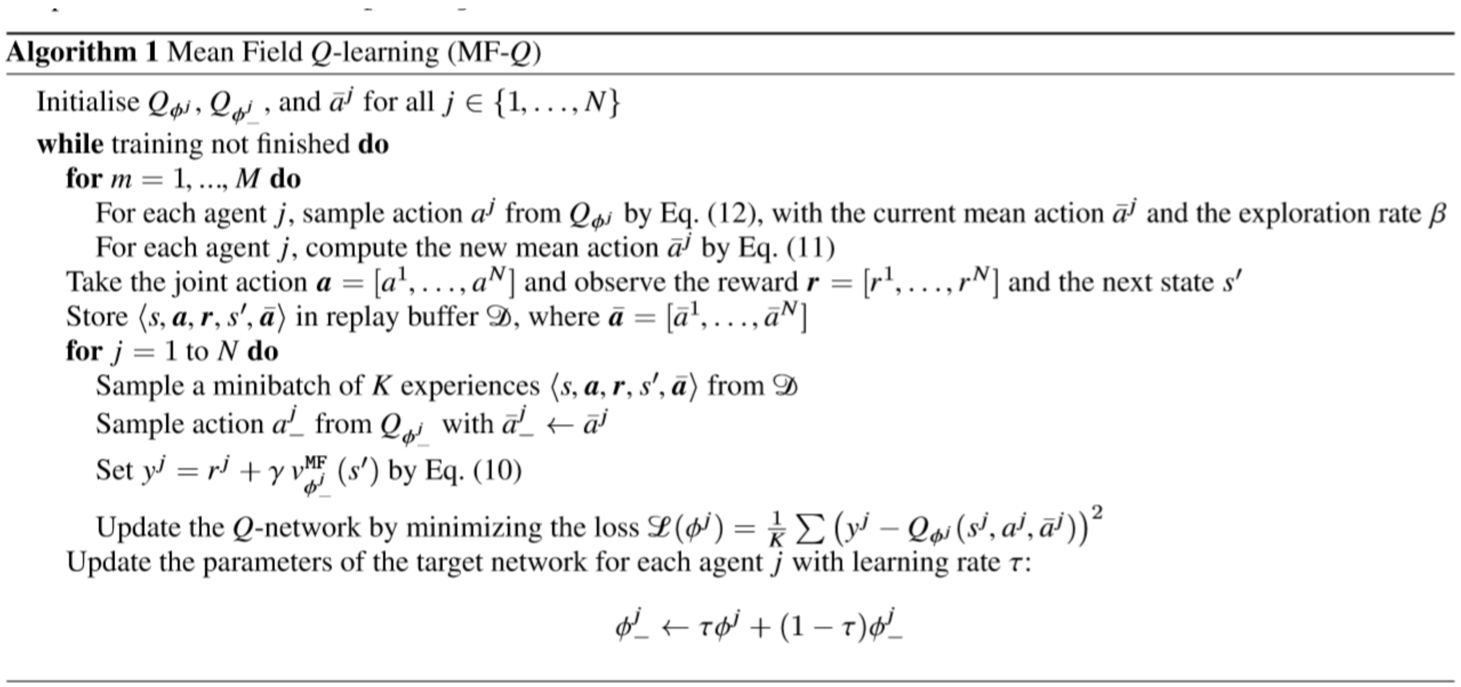


Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

```
for episode = 1 to $M$ do
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length do
    for each agent $i$, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward $r$ and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to $N$ do
      Sample a random minibatch of $S$ samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma\, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:
          $\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i(o_i^j)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$
    end for
    Update target network parameters for each agent $i$:
          $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
  end for
end for
```

7. 基于平均场MF-Q

利用平均场建模智能体之间的交互



Algorithm 1 Mean Field Q-learning (MF-Q)

```
Initialise $Q_{\phi^j}$, $Q_{\phi^j_-}$, and $\bar{a}^j$ for all $j \in \{1, \ldots, N\}$
while training not finished do
  for $m = 1, ..., M$ do
    For each agent $j$, sample action $a^j$ from $Q_{\phi^j}$ by Eq. (12), with the current mean action $\bar{a}^j$ and the exploration rate $\beta$
    For each agent $j$, compute the new mean action $\bar{a}^j$ by Eq. (11)
  Take the joint action $\boldsymbol{a} = [a^1, \ldots, a^N]$ and observe the reward $\boldsymbol{r} = [r^1, \ldots, r^N]$ and the next state $s'$
  Store $\langle s, \boldsymbol{a}, \boldsymbol{r}, s', \bar{\boldsymbol{a}} \rangle$ in replay buffer $\mathcal{D}$, where $\bar{\boldsymbol{a}} = [\bar{a}^1, \ldots, \bar{a}^N]$
  for $j = 1$ to $N$ do
    Sample a minibatch of $K$ experiences $\langle s, \boldsymbol{a}, \boldsymbol{r}, s', \bar{\boldsymbol{a}} \rangle$ from $\mathcal{D}$
    Sample action $a^j_-$ from $Q_{\phi^j_-}$ with $\bar{a}^j_- \leftarrow \bar{a}^j$
    Set $y^j = r^j + \gamma\, v^{\mathrm{MF}}_{\phi^j_-}(s')$ by Eq. (10)
    Update the $Q$-network by minimizing the loss $\mathcal{L}(\phi^j) = \frac{1}{K} \sum \big(y^j - Q_{\phi^j}(s^j, a^j, \bar{a}^j)\big)^2$
  Update the parameters of the target network for each agent $j$ with learning rate $\tau$:
      $\phi^j_- \leftarrow \tau \phi^j + (1 - \tau)\phi^j_-$
```

https://github.com/mlii/mfrl