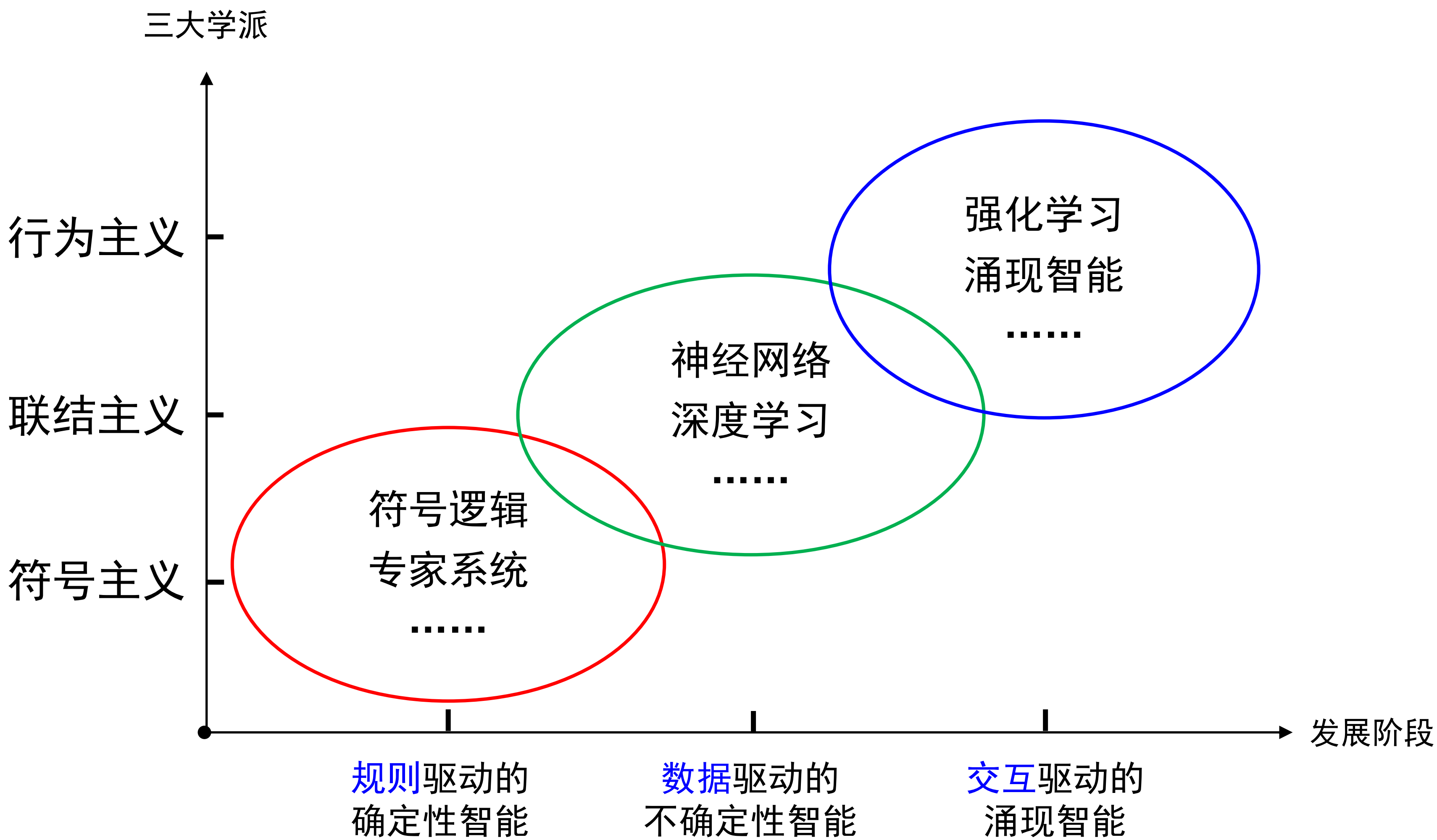
高级人工智能

沈华伟

中国科学院计算技术研究所

2022.12.29

课程总体结构

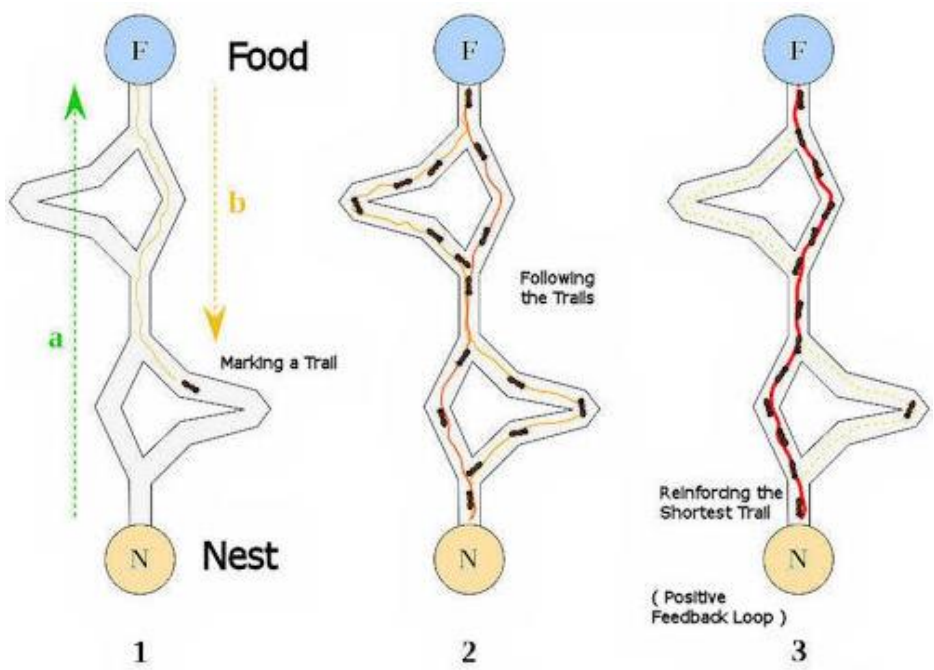

三大学派
行为主义
联结主义
符号主义
强化学习
涌现智能
……
神经网络
深度学习
……
符号逻辑
专家系统
……
发展阶段
规则驱动的
确定性智能
数据驱动的
不确定性智能
交互驱动的
涌现智能

集群智能

- 代表性方法
  - 蚁群优化算法
  - 粒子群优化算法

蚁群优化算法

- **ACO: Ant Colony Optimization**
  - 一种解空间搜索方法
  - 适用于在图上寻找最优路径



A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.

蚁群优化算法

- 形式化
  - 每个蚂蚁对应一个计算智能体
  - 蚂蚁依概率选择候选位置进行移动
  - 在经过的路径上留下“信息素”（Pheromone）
  - “信息素”随时间挥发
  - “信息素”浓度大的路径在后续的选择中会以更高的概率被选取

旅行商问题的蚁群优化求解

- 旅行商问题（TSP: Traveling Salesman Problem）
  - n个城市的有向图$G=(V,E)$

$$V=\{1,2,\cdots,n\} \qquad \mathrm{E}=\{(i,j)|i,j\in V\}$$

  - 城市之间的距离表示为

d_{ij} 为节点i和j之间的距离

  - 目标函数

$$f(w)=\sum_{l=1}^{n} d_{i_l i_{l+1}}$$

$w=(i_1,i_2,\cdots,i_n)$ 为TSP问题的任意可行解，其中 $i_{n+1}=i_1$

旅行商问题的蚁群优化求解

首先将m只蚂蚁随机放置在n个城市，位于城市i的第k只蚂蚁选择下一个城市j的概率为:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\left(\tau_{ij}(t)\right)^{\alpha}\left(\eta_{ij}(t)\right)^{\beta}}{\sum_{k\in allowed}\left(\tau_{ik}(t)\right)^{\alpha}\left(\eta_{ik}(t)\right)^{\beta}} & j\in allowed\\ 0, & otherwise\end{cases} \tag{1}$$

$\tau_{i,j}(t)$表示边(i,j)上的信息素浓度

$\eta_{i,j}(t)=1/d_{ij}$　是根据距离定义的启发信息

α和β反映了信息素与启发信息的相对重要性

旅行商问题的蚁群优化求解

当所有蚂蚁完成周游后，按以下公式进行信息素更新

$$\Delta\tau_{ij}^{k} = f(x) = \begin{cases} \dfrac{Q}{L_k}, & (i,j) \in w_k \\ 0, & otherwise \end{cases} \quad (2)$$

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t+1) + \Delta\tau_{ij}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

其中：Q为常数, w_k表示第k只蚂蚁在本轮迭代中走过的路径，L_k为路径长度，ρ为小于1的常数，反映信息素挥发速度

旅行商问题的蚁群优化求解

TSP问题蚁群算法流程

```
⑴初始化  随机放置蚂蚁，
⑵迭代过程
k=1
while k=<ItCount do (执行迭代)
      for i = 1 to m do  (对m只蚂蚁循环)
        for j = 1 to n - 1 do  (对n个城市循环)
            根据式(1)，采用轮盘赌方法在窗口外选择下一个城市j;
        将j置入禁忌表,蚂蚁转移到j;
     end for
  end for
  计算每只蚂蚁的路径长度;
  根据式(2)更新所有蚂蚁路径上的信息量;
  k = k + 1;
end while
⑶输出结果,结束算法.
```

旅行商问题的蚁群优化求解

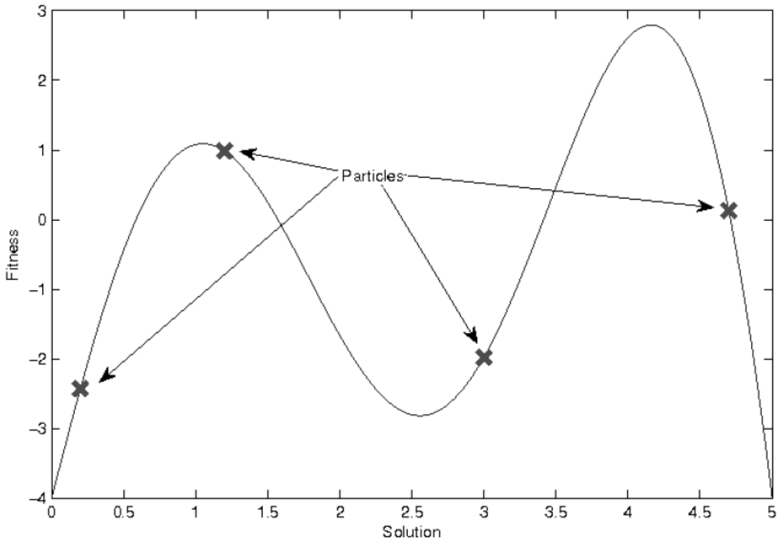
- 蚁群大小
  - 一般情况下，蚁群中的蚂蚁个数不超过TSP图中节点的个数
- 终止条件
  - 设定迭代轮数
  - 设定最优解连续保持不变的迭代轮数

集群智能

- 代表性方法
  - 蚁群优化算法
  - 粒子群优化算法

粒子群优化算法

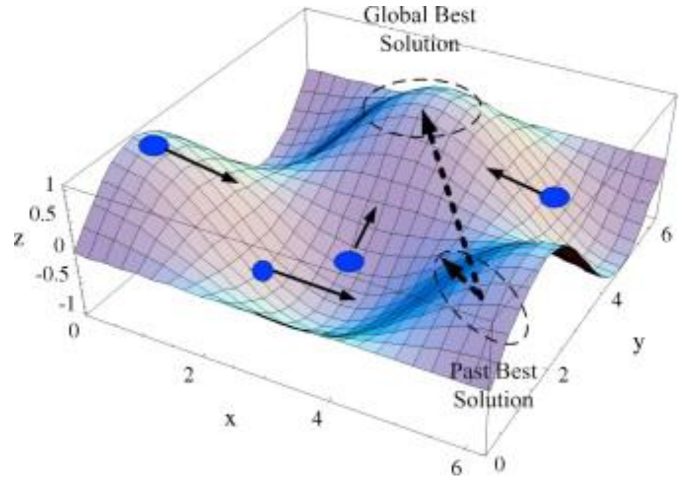
- **PSO: Particle Swarm Optimization**
  - 一种随机优化方法
  - 通过粒子群在解空间中进行搜索，寻找最优解（适应度最大的解）



James Kennedy and Russell Eberhart. Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948, Piscataway, NJ, 1995.

粒子群优化算法

- 构成要素
  - 粒子群
    - 每个粒子对应所求解问题的一个可行解
    - 粒子通过其位置和速度表示
      - 粒子i在第n轮的位置：$x_n^{(i)}$
      - 粒子i在第n轮的速度：$v_n^{(i)}$
  - 记录
    - $p_{best}^{(i)}$：粒子i的历史最好位置
    - g_{best}：全局历史最好位置
  - 计算适应度的函数
    - 适应度：$f(x)$



粒子群优化算法

- 算法过程描述
  - 初始化
    - 初始化粒子群：每个粒子的位置和速度，即$x_0^{(i)}$和$v_0^{(i)}$
    - $p_{best}^{(i)}$和g_{best}
  - 循环执行如下三步直至满足结束条件
    - 计算每个粒子的适应度：$f\left(x_n^{(i)}\right)$
    - 更新每个粒子历史最好适应度及其相应的位置，更新当前全局最好适应度及其相应的位置
    - 更新每个粒子的速度和位置

$$v_{n+1}^{(i)} = v_n^{(i)} + c_1 * r_1 * \left(p_{best}^{(i)} - x_n^{(i)}\right) + c_2 * r_2 * \left(g_{best} - x_n^{(i)}\right)$$

$$x_{n+1}^{(i)} = x_n^{(i)} + v_{n+1}^{(i)}$$

粒子群优化算法

- 粒子速度更新公式解读

$$v_{n+1}^{(i)} = \boxed{v_n^{(i)}} + c_1 * r_1 * \boxed{\left(p_{best}^{(i)} - x_n^{(i)}\right)} + c_2 * r_2 * \boxed{\left(g_{best} - x_n^{(i)}\right)}$$

①惯性项

保持原速度不变的倾向

②记忆项

回到历史最好位置的倾向

③社会项

走向粒子群全局最好位置的倾向

粒子群优化算法

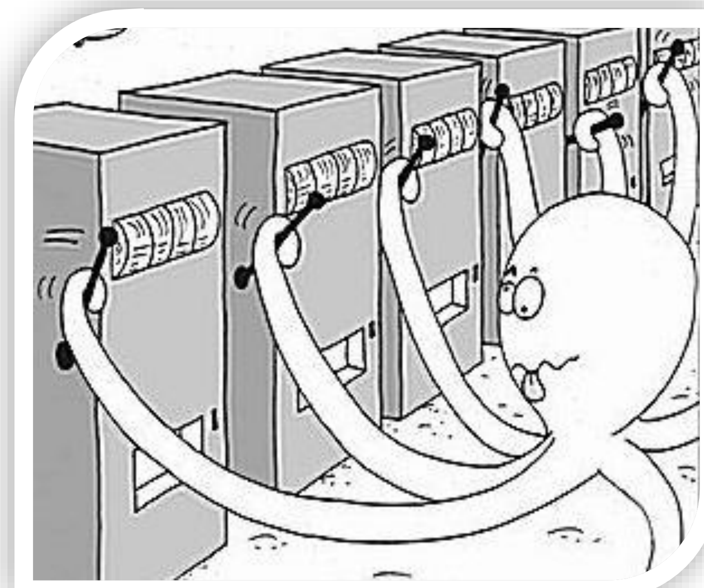
- 算法终止条件
  - 迭代的轮数
  - 最佳位置连续未更新的轮数
  - 适应度函数的值到达预期要求

大纲

- 强化学习
- 多臂赌博机
- 马尔科夫决策过程

多臂赌博机 (Multi-armed bandit)

- 一台赌博机有多个摇臂，每个摇臂摇出的奖励（reward）大小不确定，玩家希望摇固定次数的臂所获得的期望累积奖励最大



问题形式化

- 行为：摇哪个臂
- 奖励：每次摇臂获得的奖金
- A_t表示第t轮的行为，R_t表示第t轮获得的奖励
- 第t轮采取行为a的期望奖励为：

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$$

Question:

假如摇臂T次，那么按照什么策略摇臂，才能使期望累积奖励最大呢？

Answer:

当$q_*(a)$已知时，每次都选择$q_*(a)$最大的a（贪心策略）

贪心策略

- 一般情况下， $q_*(a)$ 对于玩家而言是未知的或具有不确定性，此时该采用什么样的策略呢？

- 玩家在第t轮时只能依赖于当时对$q_*(a)$的估值$Q_t(a)$进行选择

- 此时，贪心策略在第t轮选择$Q_t(a)$最大的a

在$q_*(a)$未知时，贪心策略还是最好的策略吗？

利用和探索

- 利用：Exploitation
  - 按照贪心策略进行选择，即选择$Q_t(a)$最大的行为a
  - 优点：最大化即时奖励
  - 缺点：由于$Q_t(a)$只是对$q_*(a)$的估计，估计的不确定性导致按照贪心策略选择的行为不一定是$q_*(a)$最大的行为

- 探索：Exploration
  - 选择贪心策略之外的行为（non-greedy actions）
  - 缺点：短期奖励会比较低
  - 优点：长期奖励会比较高，通过探索可以找出奖励更大的行为，供后续选择

✓ 每步选择在“利用”和“探索”中二选一

✓ 如何平衡“利用” 和“探索” 是关键

贪心策略和ε贪心策略

- 贪心策略形式化地表示为

$$A_t \doteq \arg\max_a Q_t(a)$$

当有多个行为的$Q_t(a)$同时为最大时，随机选择一个

- **ε**贪心策略
  - 以概率$1-\boldsymbol{\varepsilon}$按照贪心策略进行行为选择——Exploitation
  - 以概率**ε**在所有行为中随机选择一个——Exploration
  - **ε**的取值取决于$q_*(a)$的方差，方差越大**ε**取值应越大

行为选择策略

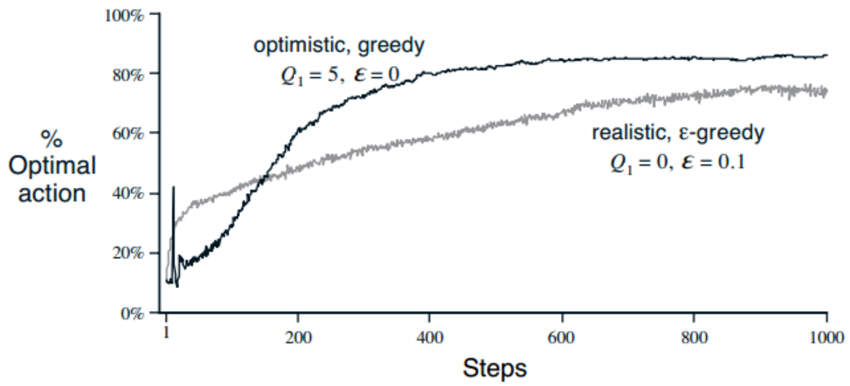
- 如何制定合适的行为选择策略？
  - 贪心策略：选择当前估值最好的行为
  - ε贪心策略：以一定的概率随机选择非贪心行为（non-greedy actions），但是对于非贪心行为不加区分

- 行为选择策略
  - 平衡exploitation和exploration，应对行为估值的不确定性
  - 关键：确定每个行为被选择的概率

乐观初值法

- 行为的初始估值
  - 前述贪心策略中，每个行为的初始估值为0
  - 每个行为的初始估值可以帮助我们引入先验知识
  - 初始估值还可以帮助我们平衡exploitation和exploration

- 乐观初值法：Optimistic Initial Values
  - 为每个行为赋一个高的初始估值
  - 好处：初期每个行为都有较大机会被explore



UCB行为选择策略

- UCB：Upper-Confidence-Bound

$$A_t \doteq \underset{a}{\arg\max}\left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}\right]$$

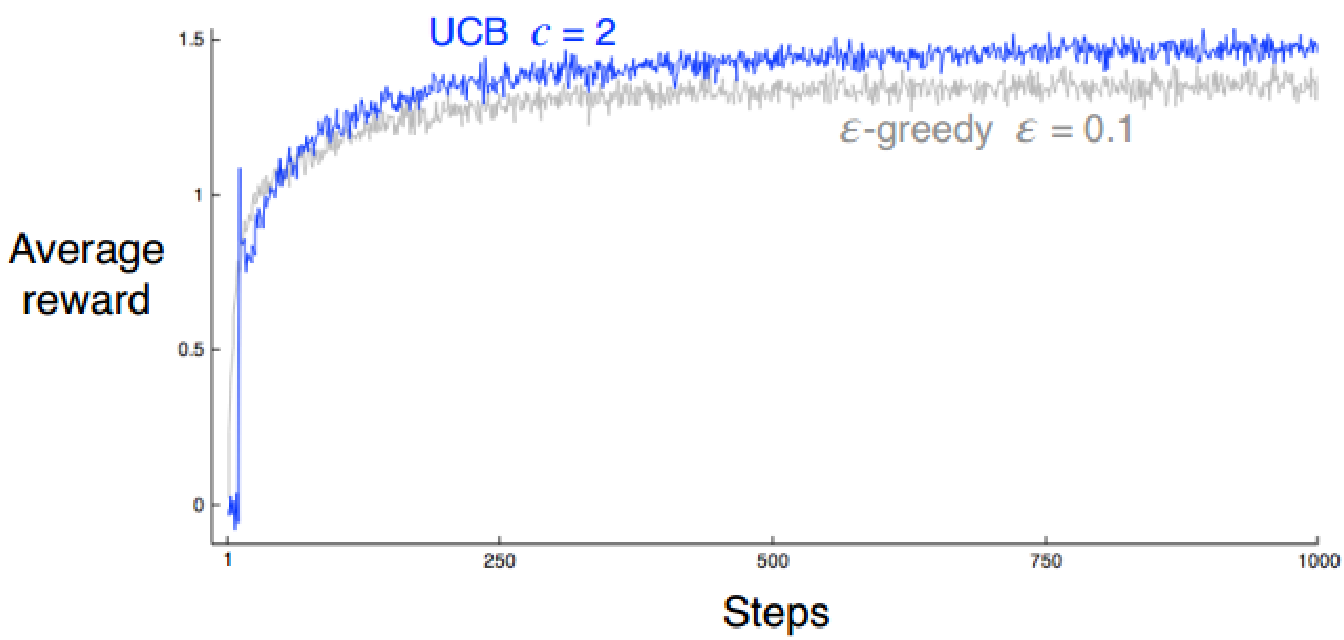
$N_t(a)$表示时刻t之前行为a被选择的次数

公式解读

- ✓ 选择潜力大的行为：依据估值的置信上界进行行为选择
- ✓ 第一项表示当前估值要高，e.g., 接近greedy action
- ✓ 第二项表示不确定性要高，e.g., 被选择的次数少
- ✓ 参数c用来控制exploration的程度

UCB策略 vs. ε贪心策略

UCB策略一般会优于ε贪心策略，不过最初几轮相对较差

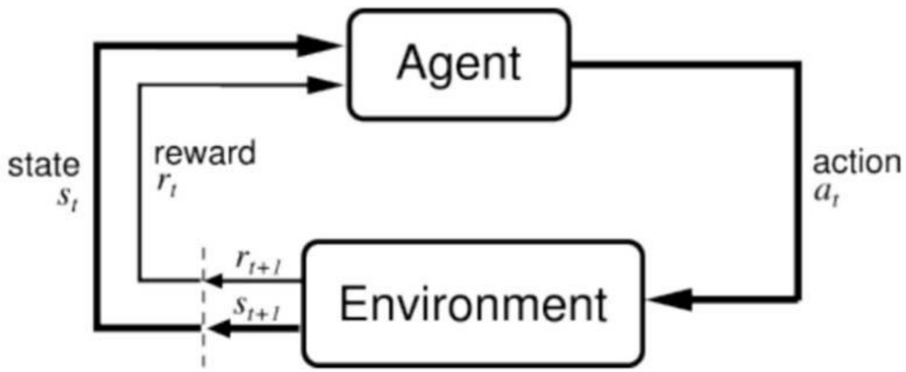


UCB策略实现起来比ε贪心策略要复杂，在多臂赌博机之外的强化学习场景中使用较少

马尔科夫决策过程的要素

- 智能体和环境按照离散的时间步进行交互

  $t = 0,1,2,3,\cdots$



- 形式化记号
  - 智能体在时间步t所处的状态记为$S_t \in S$
  - 智能体在时间步t所采取的行为记为$A_t \in A(s)$
  - 采取行为A_t后智能体转到状态S_{t+1}，并获得奖励$R_{t+1} \in R$
  - 马尔科夫决策过程产生的序列记为

    $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$

奖励假设

- 目标和奖励
  - 目标：长期的或最终的
  - 奖励：即时的
- 奖励假设：reward hypothesis

> That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)

- 如何设置奖励呢？
  - 奖励是我们与智能体进行交互的途径

策略和状态估值函数

- 策略
  - 状态到行为的映射
  - 随机式策略：$\pi(a|s)$
  - 确定式策略：$a = \pi(s)$
- 给定策略π，状态估值函数(state-value function)定义为

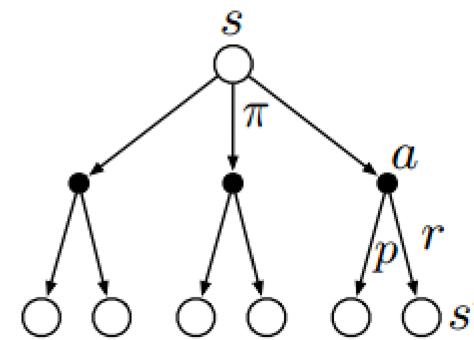
$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S}$$

- 给定策略π，行为估值函数(action-value function)定义为

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

贝尔曼方程

- 状态估值函数的贝尔曼方程



Backup diagram for v_π

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a)\Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']\Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big], \quad \text{for all } s \in \mathcal{S}
\end{aligned}
$$

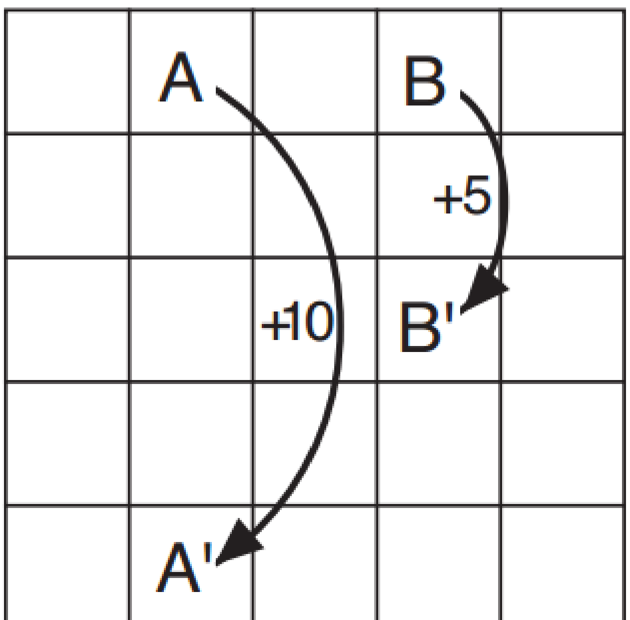
Bellman equation expresses a relationship between the value of a state and the values of its successor states

贝尔曼方程的作用

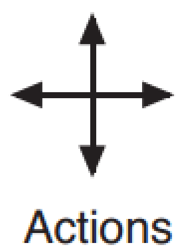
- **贝尔曼方程**定义了状态估值函数的依赖关系
  - 给定策略下，每个状态的估值视为一个变量
  - 所有状态（假如有n个）的估值根据贝尔曼方程形成了一个具有n个方程和n个变量的线性方程组
  - 求解该方程组即可得到该策略下每个状态的估值

例子：Gridworld

- 游戏规则：正常移动一次的奖励为0；出界时回到当前位置，奖励为-1；A位置选择任意方向都到达A‘，奖励为+10；B位置选择任意方向都到达B’，奖励为+5；折扣率$\gamma = 0.9$。
- 假定策略为：从每个格子等概率选择四个移动方向（行为）



游戏规则



策略

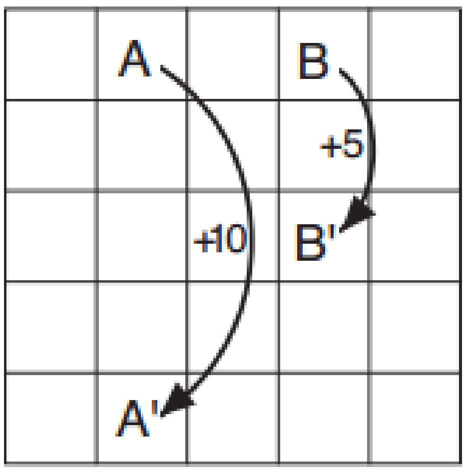
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

策略对应的状态估值函数

注意：这里的计算精度为小数点后一位

例子：Gridworld

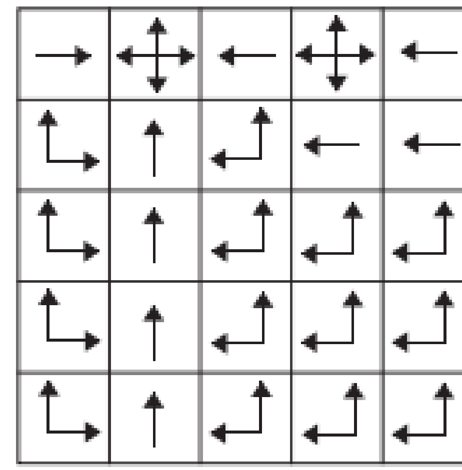
- 最优状态估值函数和最优策略



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

最优策略对应的状态估值函数

最优策略

大纲

- 策略学习
  - 动态规划方法
  - 蒙特卡洛方法
  - 时序差分方法
  - 参数近似方法

策略估值

- 给定策略π，该策略下的状态估值函数满足

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big]
\end{aligned}
$$

- 假如环境$p(s',r|s,a)$已知，状态估值函数的求解方式有
  - 求解线性方程组
    - 计算开销大
  - 寻找不动点——迭代策略估值

策略提升

- 策略估值的目标是为了寻找更优的策略（策略提升）
  - 策略估值根据策略π计算其估值函数v_π

- 策略提升
  - 根据当前策略的估值函数，寻找更优的策略（如果存在），逐步寻找到最优策略
    - 根据策略π的估值函数v寻找更优策略π'
  - 提升方法
    - 给定一个确定策略π，在状态s下选择行为a，后续按照策略π行动所得的估值$q_\pi(s,a)$是否高于完全按照策略π行动得到的估值$v_\pi(s)$

$$
\begin{aligned}
q_\pi(s,a) &\doteq \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big]
\end{aligned}
$$

策略迭代

- 从初始策略π_0开始，迭代进行“策略估值(E)”和“策略提升(I)”，最终得到最优策略π_*

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Policy iteration (using iterative policy evaluation)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\quad \Delta \leftarrow 0$
   $\quad$ For each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

估值迭代

- 从初始状态估值v_0开始，进行估值迭代，找到最优状态估值v_*，进而根据v_*按照贪心方式得到最优策略π_*

Value iteration

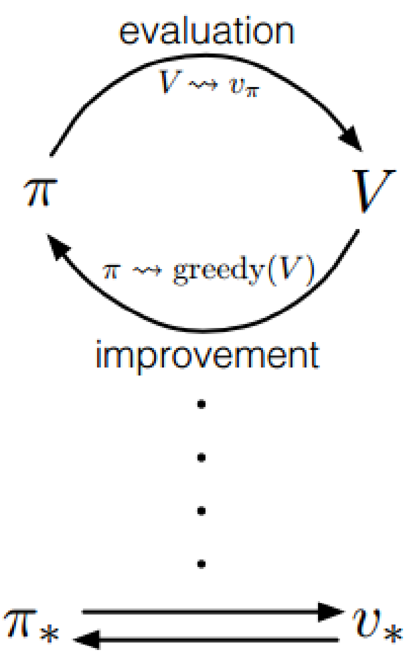
Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

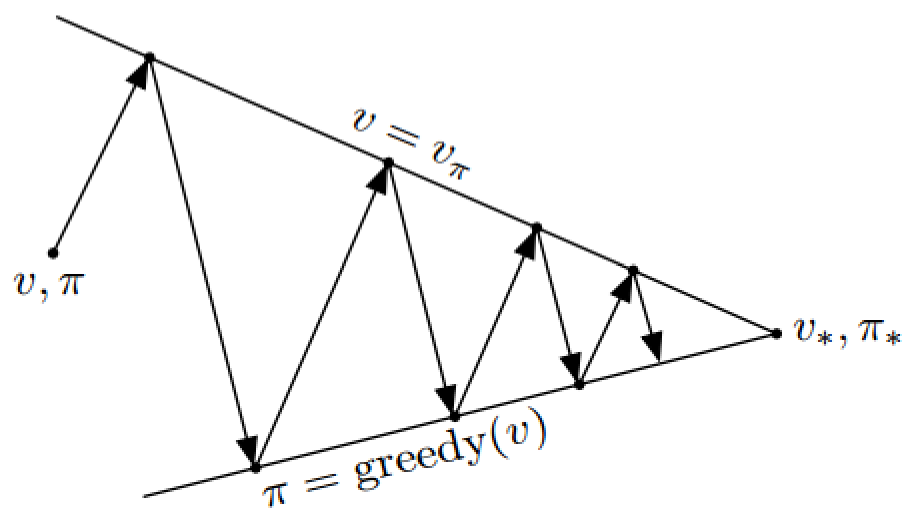
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

广义策略迭代

- 迭代进行两个阶段：
  - 策略估值：让新的估值和当前的策略保持一致
  - 策略提升：根据当前估值，得到相应的贪心策略





大纲

- 策略学习
  - 动态规划方法
  - 蒙特卡洛方法
  - 时序差分方法
  - 参数近似方法

蒙特卡洛方法的动机

- 动态规划方法需要知道关于环境的完整模型
- 大部分情况下没有关于环境的完整模型，或模型过于复杂
- 蒙特卡洛方法
  - 从真实或者模拟的经验(experience)中计算状态（行动）估值函数
  - 不需要关于环境的完整模型

蒙特卡洛方法

- 状态估值
  - 从某个状态s出发，使用当前策略π通过蒙特卡洛模拟的方式生成多个episode，使用这些episode的平均收益（return）近似状态估值函数$v_\pi(s)$

First-visit MC prediction, for estimating $V \approx v_\pi$

Initialize:
  $\pi \leftarrow$ policy to be evaluated
  $V \leftarrow$ an arbitrary state-value function
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:
  Generate an episode using π
  For each state s appearing in the episode:
    $G \leftarrow$ the return that follows the first occurrence of s
    Append G to $Returns(s)$
    $V(s) \leftarrow \text{average}(Returns(s))$

收敛速度大约为：$1/\sqrt{n}$（n表示蒙特卡洛模拟次数）

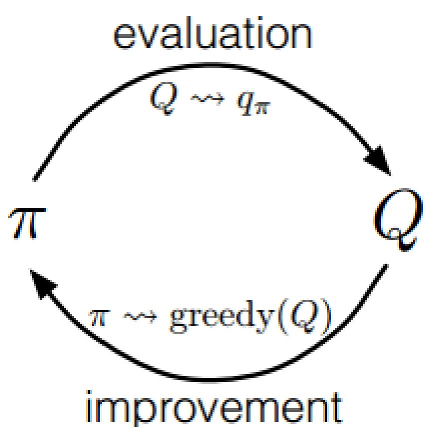
蒙特卡洛方法的优势

- 直接根据真实经验或模拟经验计算状态估值函数
- 不同状态的估值在计算时是独立的
  - 不依赖于“自举”方法

基于蒙特卡洛方法的策略迭代

- 在完整的环境模型未知时，仅有状态估值$v_\pi(s)$无法得出策略π

- 大多数情况下，直接使用蒙特卡洛方法计算行为估值函数$q_\pi(s,a)$，进而采用贪心方法得到策略π



$$\pi_0 \xrightarrow{\text{E}} q_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_*$$

$$\pi(s) \doteq \arg\max_a q(s,a)$$

大纲

- 策略学习
  - 动态规划方法
  - 蒙特卡洛方法
  - 时序差分方法
  - 参数近似方法

时序差分方法

- 非平稳情形下的蒙特卡洛方法（恒定步长）

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big]$$

G_t表示第t轮蒙特卡洛模拟的收益

$V(S_t)$表示第t轮对状态S_t的估值

- 时序差分方法（Temporal Difference: TD）

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$$

不需要根据完整的episode计算G_t，只需要模拟几步（这里是1步）之后更新状态估值

时序差分方法分析

- 时序差分方法是强化学习中最核心的策略学习方法
- TD和蒙特卡洛方法的联系和区别
  - 联系：都是从经验中学习
    - 非平稳情形下的蒙特卡洛方法是TD的特例
  - 区别：蒙特卡洛方法需要episode完整的信息，TD只需要episode的部分信息
- TD和动态规划方法的联系和区别
  - 联系：TD和动态规划方法都采用自举的方法
  - 区别：动态规划方法依赖于完整的环境模型进行估计，TD依赖于经验进行估计

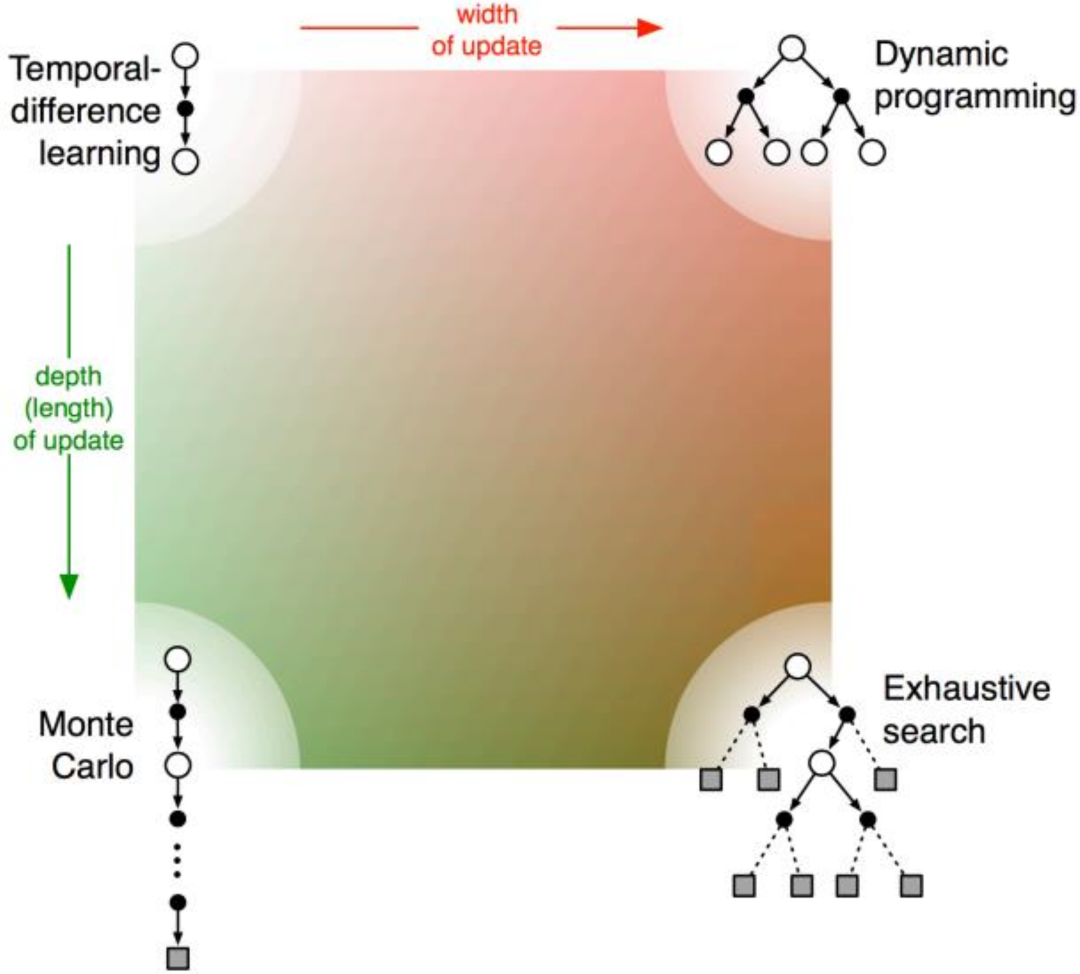
时序差分方法分析

- 时序差分方法是“learn a guess from a guess”

- 时序差分方法收敛吗？
  - 收敛

- 时序差分方法和蒙特卡洛方法收敛速度哪个快？

动态规划/蒙特卡洛/时序差分对比



课程内容

- 博弈
  - 基本概念
  - 纳什均衡
  - 机制设计
- 两个经济学的应用
  - 拍卖
  - 讨价

博弈的要素

- 局中人（Player）
  - 在博弈中有权决定自己行动方案的博弈参加者
  - 局中人不一定是具体的人
    - 如球队、军队、企业
  - 博弈中利益完全一致的参与者只能看成一个局中人
    - 如桥牌中的南北方和东西方
- 重要假设：局中人是自私的理性人
  - 不存在侥幸心理
  - 不可能利用其它局中人的失误来扩大自己的利益
  - 以最大化个人利益为目的

博弈的要素

- 策略集合(Strategy Set)
  - 策略：博弈中可供局中人选择的行动方案
  - 参加博弈的局中人i的策略集合记为A_i
  - 所有局中人的策略形成的策略组，称为局势，记作S
  - 多人博弈中假定有 n个局中人，每个局中人从自己的策略集合中选择一个策略s_i， $s_i \in A_i$ ，这样就形成了一个局势$S = \{s_1, s_1, \cdots, s_n\}$

- 田忌赛马中田忌的策略集合
  - {上中下、上下中、中上下、中下上、下上中、下中上}

博弈的要素

- 效用函数(Payoff)
  - 通常用U来表示
  - 对每个参与博弈的局中人，都有一个相应的效用函数
  - 效用函数在静态博弈中一般是局势的函数
  - 在动态博弈中效用函数可能是局势的函数，也可能还有其它因素，比如时间
  - 每个局中人的目的都是最大化自己的效用

博弈案例

- 田忌赛马
  - （田）忌数与齐诸公子驰逐重射。孙子见其马足不甚相远，马有上、中、下辈。于是孙子谓田忌曰："君弟重射，臣能令君胜。"田忌信然之，与王及诸公子逐射千金。及临质，孙子曰："今以君之下驷与彼上驷，取君上驷与彼中驷，取君中驷与彼下驷。"既驰三辈毕，而田忌一不胜而再胜，卒得王千金。

	第一场	第二场	第三场	获胜方
齐王	上	中	下	
田忌1	上	中	下	齐王
田忌2	上	下	中	齐王
田忌3	中	上	下	齐王
田忌4	中	下	上	齐王
田忌5	下	上	中	田忌
田忌6	下	中	上	齐王

囚徒困境

- 局中人
  - 两个囚徒
- 策略
  - 抗拒
  - 坦白
- 效用函数矩阵

囚徒B

囚徒A

	抗拒	坦白
抗拒	-1,-1	-10,0
坦白	0,-10	-3,-3

剪刀-石头-布(Rock-paper-scissors)

- 局中人
  - 两个玩家
- 策略
  - 剪刀、石头、布
- 效用函数矩阵

玩家二

玩家一		剪刀	石头	布
	剪刀	0,0	-1,1	1,-1
	石头	1,-1	0,0	-1,1
	布	-1,1	1,-1	0,0

纳什均衡

- 最佳应对
  - 假设s是局中人1的一个选择策略， t是局中人2的一个选择策略； $U_1(s,t)$ 是局中人1从这组决策中获得的收益， $U_2(s,t)$ 是局中人2从这组决策中获得的收益
  - 针对局中人2的策略t，若局中人1用策略s产生的收益大于或等于其任何其他策略，则称策略s是局中人1对局中人2的策略t的最佳应对
    - $U_1(s,t) \geq U_1(s',t)$，s'是局中人1除s外的其它策略
  - 如果一个局中人的某个策略对其它局中人的任何策略都是最佳应对，那么这个策略就是该局中人的占优策略

纳什均衡

- 纳什均衡
  - 定义：如果一个局势下，每个局中人的策略都是相对其他局中人当前策略的最佳应对，则称该局势是一个纳什均衡
- 纳什均衡就是博弈的一个均衡解
- 是一个僵局
  - 即给定其他人不动，没有人有动的积极性
  - 谁动谁吃亏

纳什均衡

- 例子：囚徒困境
  - 纳什均衡：双方都坦白
    - 一方保持策略不变（坦白），另一方如果改变策略（抗拒），其效用会降低（从-3变成-10）

囚徒B

囚徒A

	抗拒	坦白
抗拒	-1,-1	-10,0
坦白	0,-10	-3,-3

混合策略纳什均衡

- 例子：剪刀-石头-布
  - 玩家一的策略选择分布记为$p = \{p_1, p_2, 1 - p_1 - p_2\}$，玩家二的策略选择分布记为$q = \{q_1, q_2, 1 - q_1 - q_2\}$
  - 假设玩家一的策略分布不变，玩家二策略选择的效用为
    - 剪刀：$0 * p_1 + (-1) * p_2 + 1 * (1 - p_1 - p_2) = 1 - p_1 - 2p_2$
    - 石头：$1 * p_1 + 0 * p_2 + (-1) * (1 - p_1 - p_2) = 2p_1 + p_2 - 1$
    - 布：$(-1) * p_1 + 1 * p_2 + 0 * (1 - p_1 - p_2) = p_2 - p_1$
  - 令玩家二的各个策略的效用相等，得到$p_1 = p_2 = \frac{1}{3}$
  - 同理可得$q_1 = q_2 = \frac{1}{3}$
- 剪刀-石头-布的混合纳什均衡态
  - 每个玩家各以1/3的概率选择剪刀、石头和布
  - 期望收益为0

玩家二

玩家一

	剪刀	石头	布
剪刀	0,0	-1,1	1,-1
石头	1,-1	0,0	-1,1
布	-1,1	1,-1	0,0

社会最优示例

- 囚徒困境案例

囚徒B

囚徒A		抗拒	坦白
	抗拒	-1,-1	-10,0
	坦白	0,-10	-3,-3

帕累托最优的决策组合一共有3个，分别是（坦白，抗拒），（抗拒，坦白）和（抗拒，抗拒），纳什均衡策略组合（坦白，坦白）不是帕累托最优，社会最优策略组合是（抗拒，抗拒）

课程内容

- 博弈
  - 基本概念
  - 纳什均衡
  - 机制设计

- 应用案例
  - 拍卖
  - 讨价

首价密封报价拍卖

- 纳什均衡
  - 每个竞拍者的报价低于其对商品的估价
- 解读
  - 共有n个竞拍者，竞拍者i的估价记为v_i，报价记为b_i，其他竞拍者的估价服从$[a, b]$区间上的均匀分布，且诚实出价
  - $b_i < a$时，竞标失败，收益为0
  - 竞拍者i赢得竞拍的概率为 $\left(\frac{b_i - a}{b - a}\right)^{n-1}$
  - 竞拍者的期望收益是 $f(b_i) = (v_i - b_i)\left(\frac{b_i - a}{b - a}\right)^{n-1}$

首价密封报价拍卖

- 解读（续）
  - 期望收益

$$f(b_i) = (v_i - b_i)\left(\frac{b_i - a}{b - a}\right)^{n-1}$$

  - 期望收益关于报价b_i的梯度为

$$f'(b_i) = -\left(\frac{b_i - a}{b - a}\right)^{n-1} + (n-1)(v_i - b_i)\left(\frac{b_i - a}{b - a}\right)^{n-2}\frac{1}{b - a}$$

$$= \left(\frac{b_i - a}{b - a}\right)^{n-2}\left(\frac{-nb_i + a + (n-1)v_i}{b - a}\right)$$

  - 最优报价为

$$b_i^* = \frac{a + (n-1)v_i}{n}$$

✓ 最优报价低于估价

✓ 竞拍者越多，报价越接近于估价

次价密封报价拍卖

- 纳什均衡
  - 每个竞拍者会倾向于采用其对商品的估价进行报价

- 解读
  - 给定一个竞拍者，其估价记为v，报价记为b，其他竞拍者的最高报价记为b^*
  - 理性行为假设下，报价不会高于估价，即$b \leq v$
  - 此时，根据b^*的取值有三种情形
    - $b^* > v$：收益为0；将报价从b提高到v，收益不变
    - $b^* < b$：收益为$v - b^*$；将报价从b提高到v，收益不变
    - $b \leq b^* \leq v$：收益为0；将报价从b提高到v，收益变为$v - b^*$

课程内容

- 博弈
  - 基本概念
  - 纳什均衡
  - 机制设计
- 应用案例
  - 拍卖
  - 讨价

讨价

- 卖家和买家之间的博弈
- 讨价的对象是双方对商品估价之差
  - 假设所有因素都体现在估价中
    - 时间、情感、眼缘等
  - 例子：
    - 衣服进价80，标价200
    - 卖家对衣服的估价在80和200之间，譬如120
    - 买家的估价假如为160
    - 讨价的对象是双方的估价之差，即160-120=40
- 后续的讨论中，将讨价对象视为整体1
  - 卖家的估价为0，买家的估价为1

课程内容

- **maxmin策略和minmax策略**
- 匹配市场
- 中介市场
- 议价权

maxmin策略

- maxmin策略
  - 最大化自己最坏情况时的效用（收益）

$$\arg\max_{s_i}\min_{s_{-i}} u_i(s_i, s_{-i})$$

局中人i的效用函数

局中人i的策略

除i之外其他局中人的策略

maxmin策略示例

- 性别大战
  - 妻子的策略：以概率p选择韩剧，以概率$1-p$选择体育
  - 丈夫的策略：以概率q选择韩剧，以概率$1-q$选择体育
- 妻子的期望收益

$$u_w(p,q) = 2pq + (1-p)(1-q) = 3pq - p - q + 1$$

- 妻子的期望收益关于丈夫的策略q是单调的
  - 最小值的可能取值点：$q=0$或$q=1$
- 妻子的最坏期望收益

$$\min_q u_w(p,q) = \min(1-p, 2p)$$

- 妻子的maxmin策略为

$$\arg\max_p \min_q u_w(p,q)$$

妻子

丈夫		韩剧	体育
	韩剧	1,2	0,0
	体育	0,0	2,1

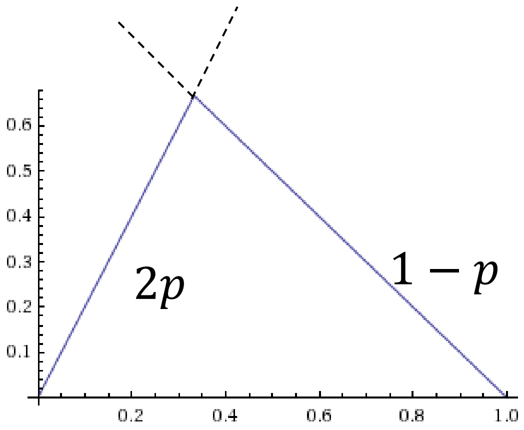
丈夫: h　　妻子: w

maxmin策略示例

- 性别大战

$$\arg\max_p \min_q u_w(p,q) = \arg\max_p \min(1-p, 2p)$$

- 解得
  - $p = \frac{1}{3}$



- 妻子的maxmin策略
  - 1/3概率选择韩剧，2/3概率选择体育

- 同理，丈夫的maxmin策略
  - 2/3概率选择韩剧，1/3概率选择体育

minmax策略

- minmax策略
  - 最小化对手的最大收益（收益）

$$\arg\min_{\boxed{s_i}}\max_{\boxed{s_j}}\boxed{u_j\left(s_i, s_j\right)}$$

对手j的效用函数

局中人i的策略

对手j的策略

minmax策略示例

- 性别大战
  - 妻子的策略：以概率p选择韩剧，以概率$1-p$选择体育
  - 丈夫的策略：以概率q选择韩剧，以概率$1-q$选择体育
- 丈夫的期望收益（注意：妻子的minmax策略考虑到是丈夫的收益）

$$u_h(p,q) = pq + 2(1-p)(1-q) = 3pq - 2p - 2q + 2$$

- 丈夫的期望收益关于其策略q是单调的
  - 最大值的可能取值点：$q=0$或$q=1$
- 丈夫的最好期望收益

$$\max_q u_h(p,q) = \max(2-2p, p)$$

- 妻子的minmax的策略为

$$\arg\min_p \max_q u_h(p,q)$$

妻子

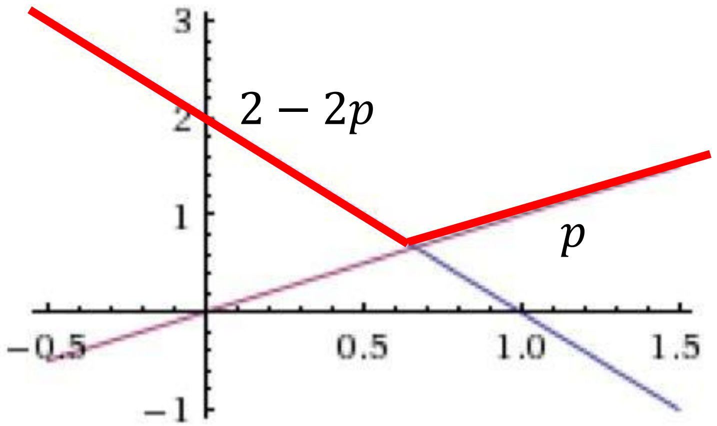
丈夫		韩剧	体育
	韩剧	1,2	0,0
	体育	0,0	2,1

丈夫: h　　妻子: w

minmax策略示例

- 性别大战

$$\arg\min_{p}\max(2-2p,p)$$



- 解得
  - $p=\frac{2}{3}$
- 妻子的minmax策略
  - 2/3概率选择韩剧，1/3概率选择体育
- 同理，丈夫的minmax策略
  - 1/3概率选择韩剧，2/3概率选择体育

maxmin策略和minmax策略

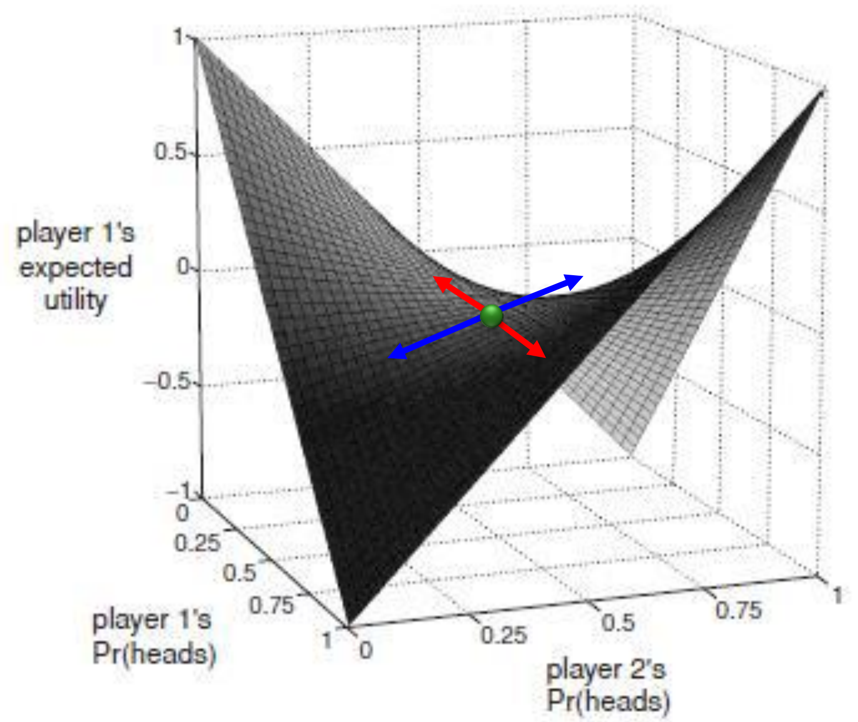
- 零和博弈情况下
  - minmax和maxmin是对偶的
  - minmax策略和maxmin策略等价于纳什均衡策略

玩家二

玩家一

	Heads	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

零和博弈的例子

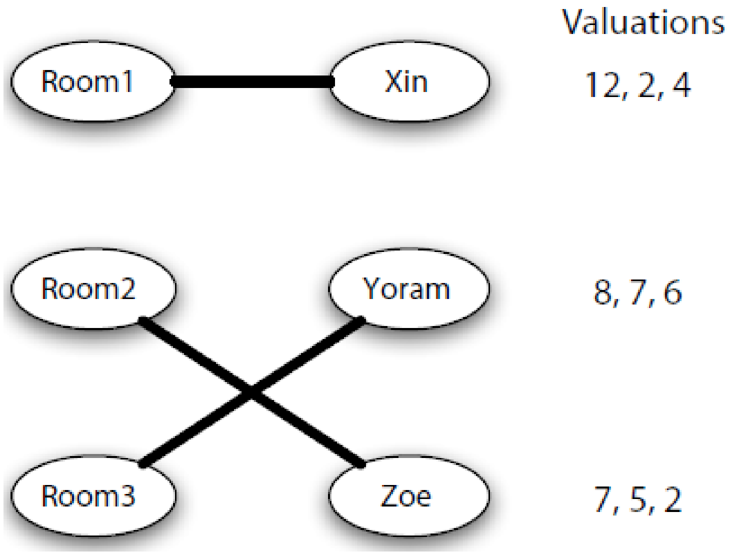


课程内容

- maxmin策略和minmax策略
- **匹配市场**
- 中介市场
- 议价权

最优匹配

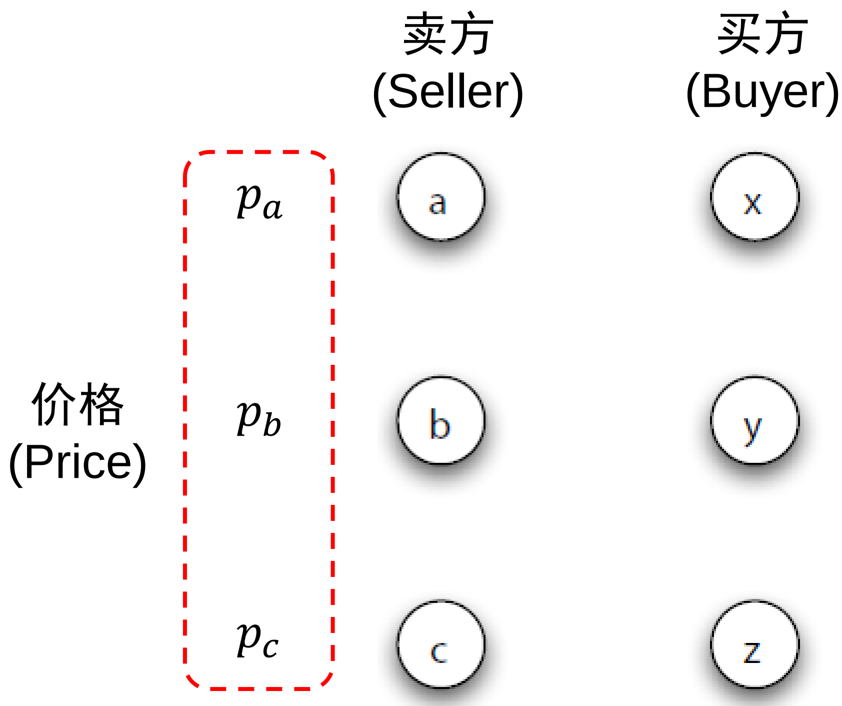
- 匹配的效用
  - 成功匹配的估价之和，称为匹配的效用
- 最优匹配
  - 效用最大的匹配
- 最优匹配对于个体而言不一定最优，甚至是最差的
  - Yoram和Zoe的最优选择是Room1
  - Yoram的最差选择是Room3



最优匹配
效用：12+6+5=23

价格导向的匹配

- 价格导向的匹配问题的形式化表示

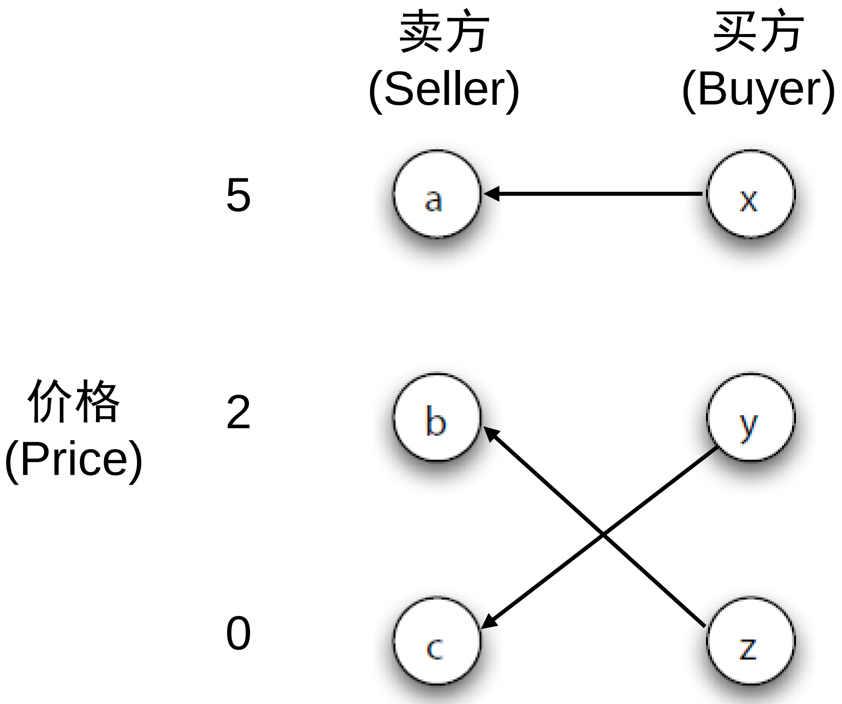


估价
(Evaluation)

$$v = \begin{bmatrix} v_{x,a} & v_{x,b} & v_{x,c} \\ v_{y,a} & v_{y,b} & v_{y,c} \\ v_{z,a} & v_{z,b} & v_{z,c} \end{bmatrix}$$

价格导向的匹配

- 示例



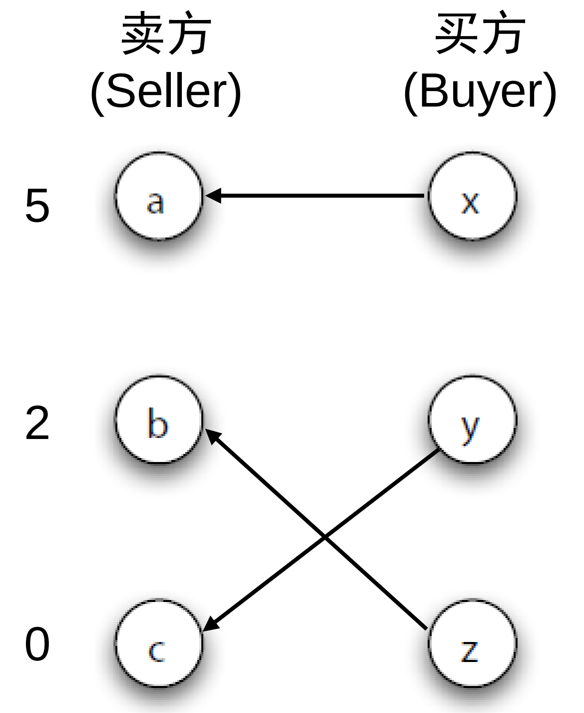
买方偏好图

估价
(Evaluation)

$$
v = \begin{array}{c} \\ x \\ y \\ z \end{array}
\begin{array}{c}
\begin{array}{ccc} a & b & c \end{array} \\
\left[\begin{array}{ccc} 12, & 4, & 2 \\ 8, & 7, & 6 \\ 7, & 5, & 2 \end{array}\right]
\end{array}
$$

市场结清价格

- 市场结清(Market-Clearing)
  - 每个卖方和买方都成交了
- 给定买方报价的情况下，如果卖方的某种价格使得对应的买方偏好图中存在完全匹配，则称卖方的这组价格为市场结清价格



买方偏好图

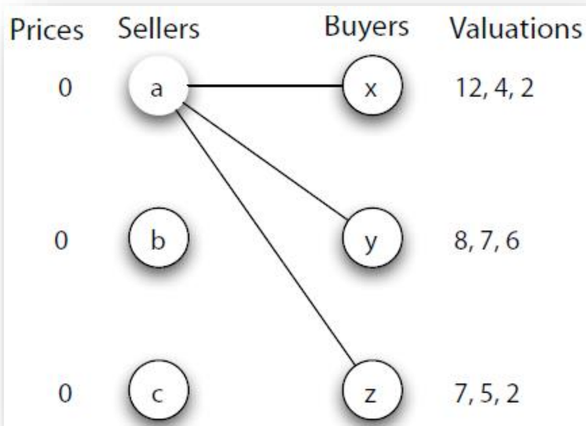
市场结清价格的存在性

- 寻找市场结清价格的过程
  - 步骤1：初始时，所有卖方的价格为0
  - 步骤2：构建买方偏好图，检查其是否存在完全匹配
    - 如果存在，当前价格是市场结清价格
    - 如果不存在，从图中找到一个受限集S（一定是买方）及其邻居$N(S)$，让 $N(S)$ 中的每个卖家的价格增加1
  - 回到步骤2（当所有价格都为正时，可以通过让所有价格减去最低价格，使最低价格为0，此操作不影响结果）
- 收敛性
  - 买卖双方的总收益有限
  - $|N(S)| < |S|$，总收益下降，但不会小于0

市场结清价格

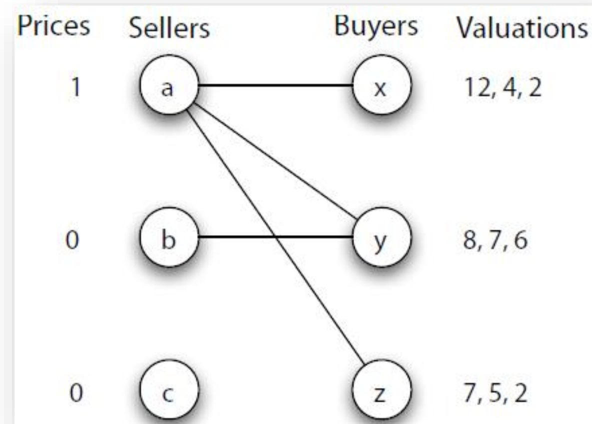
- 寻找市场结清价格的过程示例

第一轮



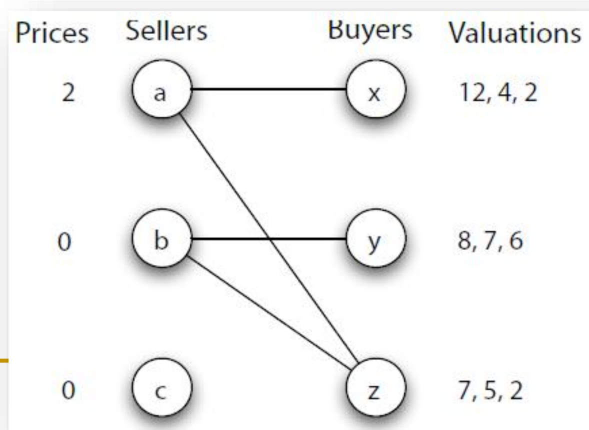
受限集S={x,y,z}，N(S)={a}

第二轮



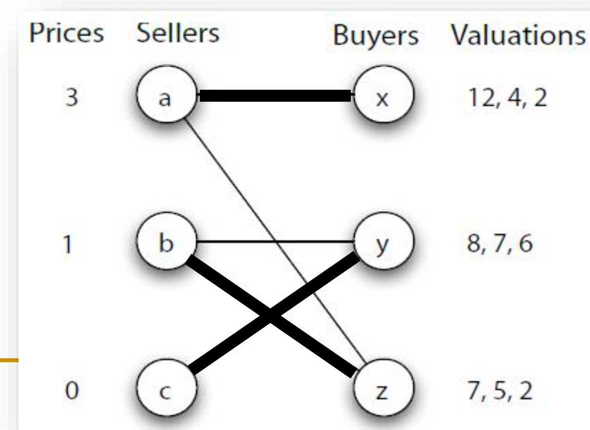
受限集S={x,z}，N(S)={a}

第三轮



受限集S={x,y,z}，N(S)={a,b}

第四轮



市场结清价格的存在性

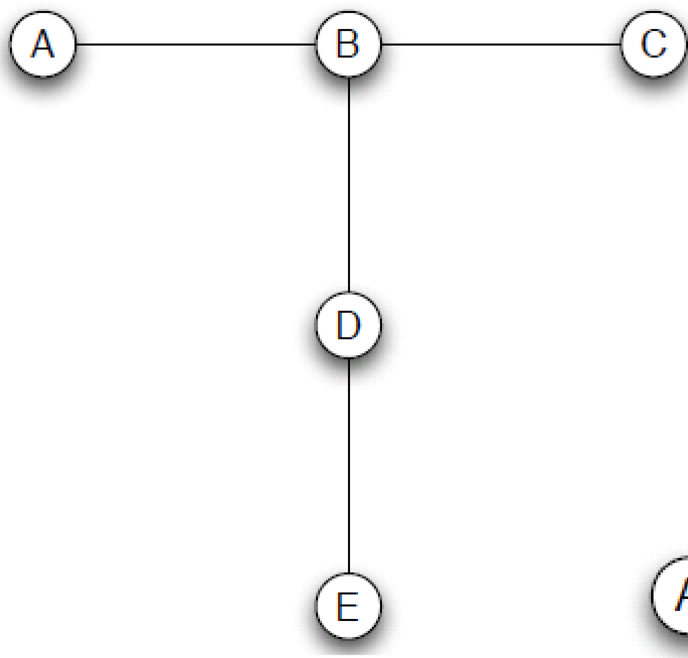
- 小结
  - 完全匹配是否存在可以通过寻找受限集来判断
  - 价格能够引导市场优化配置
  - 市场结清价格总是存在
  - 市场结清价格使得买卖双方总效用最优

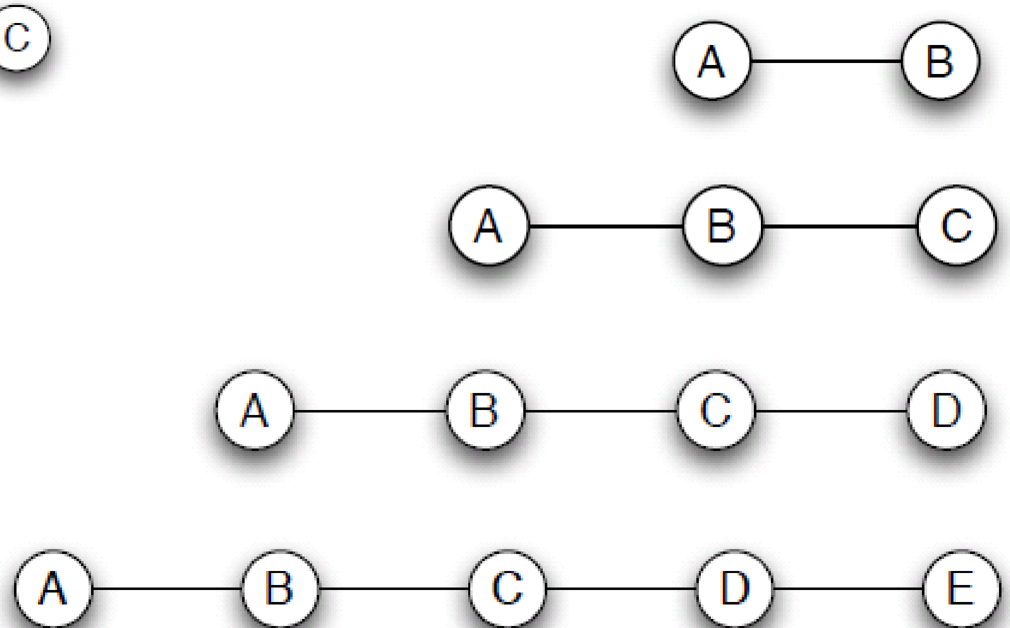
课程内容

- maxmin策略和minmax策略
- 匹配市场
- 中介市场
- 议价权

网络中节点位置的重要性

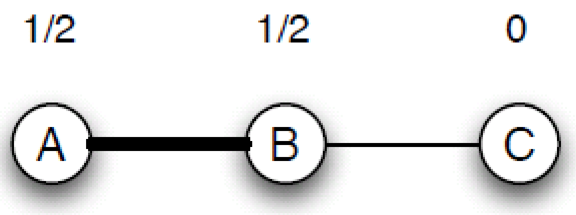
- 节点在网络中所处的位置不同，导致其在博弈中的权利不同



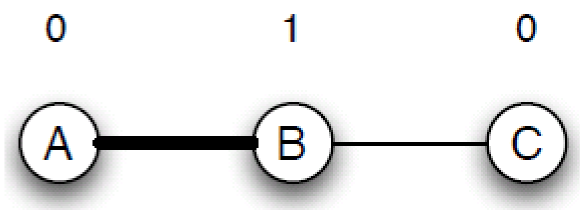


结局的稳定性

- 不稳定边
  - 对于结局中**未参与配对的边**，如果边的两个端点获得的收益之和小于1，则称这条边为不稳定边
  - 不稳定边的存在意味着其两个端点可以通过改变报价而改变结局
- 稳定结局(stable outcome)
  - 如果一个结局中不存在不稳定边，则称该结局为稳定结局



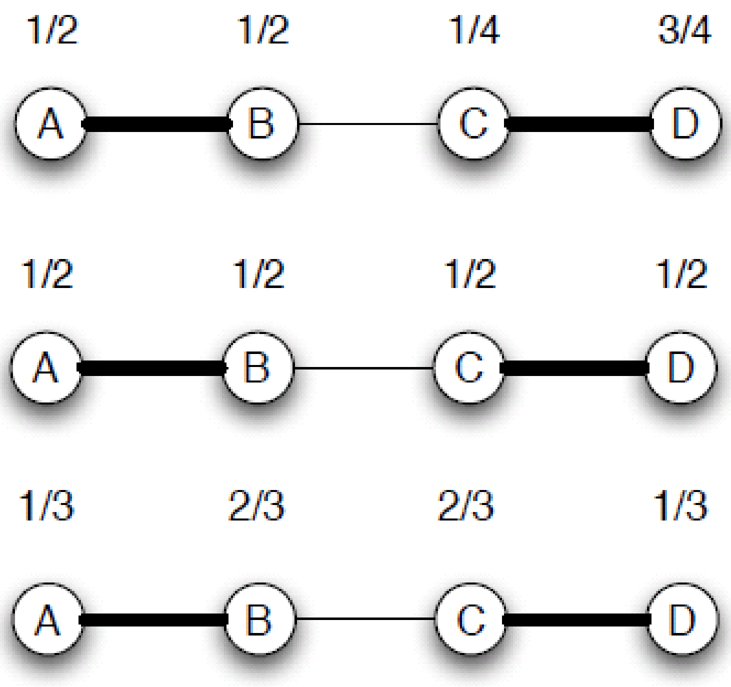
不稳定的结局
B和C的收益之和小于1



稳定的结局

结局的稳定性

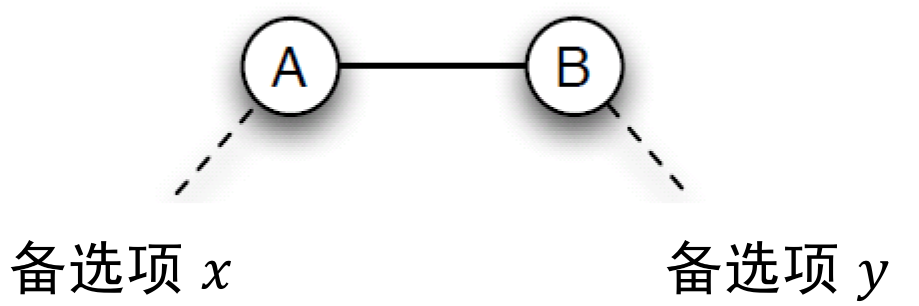
- 下列哪些结局是稳定的？



- 上面的例子中，后面两个稳定结局哪个更体现节点的议价权呢？如何判断呢？

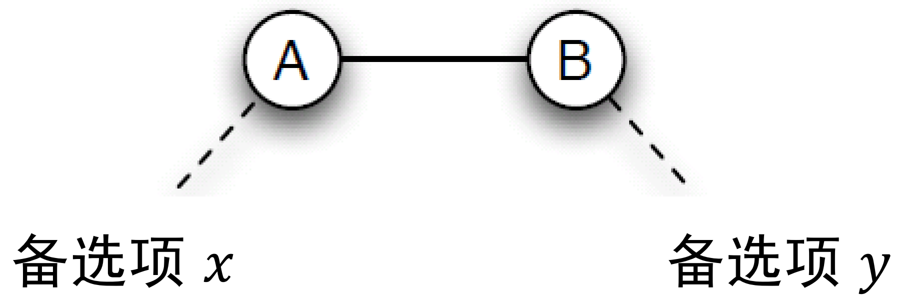
网络中的议价权

- 有备选项的议价
  - A和B二人议价，确定分配比例
  - A的备选项收益为x
  - B的备选项收益为y
  - 要求： $x + y \leq 1$；否则A和B达不成交易



纳什议价解

- 议价的对象
  - 如何分配“剩余价值”$s = 1 - x - y$



- 纳什议价解
  - A的收益是： $x + \frac{s}{2} = \frac{1+x-y}{2}$
  - B的收益是： $y + \frac{s}{2} = \frac{1+y-x}{2}$

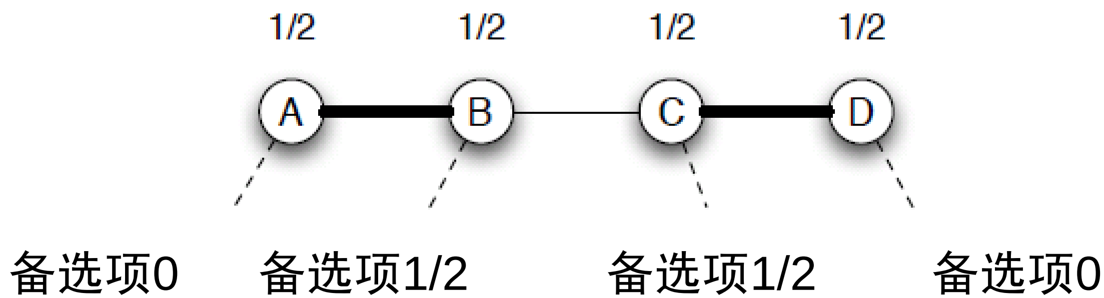
均衡结局

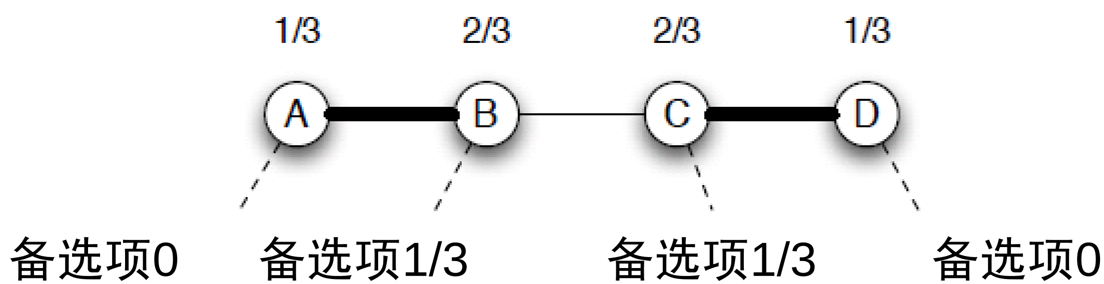
- 均衡结局（balanced outcome)
  - 给定一个结局，如果结局中的任意一个**参与配对的边**都满足纳什议价解的条件，则称该结局是均衡结局

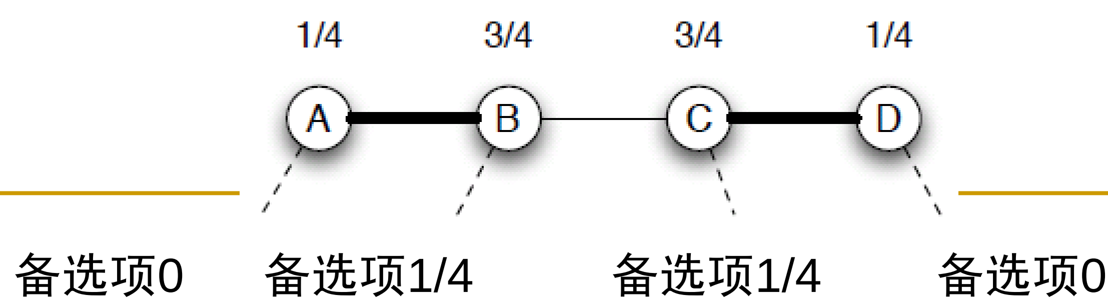
- 注意：均衡结局一定是稳定结局
  - 因此，在寻找均衡结局时，可以先寻找稳定结局，进而确定均衡结局

均衡结局

- 下列哪些结局是均衡结局？







总结

- 群体智能之博弈
  - 博弈的要素：局中人、策略、效用矩阵
  - 纳什均衡策略和混合纳什均衡策略
  - maxmin和minmax策略
  - 议价和拍卖
  - 匹配问题中价格的作用
  - 纳什议价解和均衡结局