第一节：

多智能体是为了找到对多个智能体来说比较一致的决策。

Agent的特点，自动（不需要人）、反应（需要对环境场景及时作出反应）、前瞻性（提前意识到问题并进行优化）

为什么自动很重要呢？不需要人来给每一个行为初始化，人可能还没准备好时也需要react

反应性：有明确的输入和输出，需要改变环境，需要根据环境的改变来改变自己的行为，需要及时反应。为什么很重要呢？有时候没有反应比坏反应效果更差

前瞻性，需要提早考虑，给出比较优的结果。

什么是rational agents，与程序与算法不同，程序与算法总是干相同的事，而rational agents则是干正确的事，比如在电价低谷时洗衣服这些。

简单动作空间，优化+学习

复杂空间，符号推理，满足限制，计划

自动化任务：只是单纯重复

强化学习，通过观察以及与环境交互来提高表现。

自动智能体，自己的目标永远是最重要的（self-interested），但是在保证自己的目标的前提下，也可以考虑其他的目标。

多智能体系统中的问题，不同agent之间的协作，使自私的智能体合作，分布式实现

高耦合用集中式，低耦合用分布式。

一个状态如果越无序，那么是有代价的。

在没有沟通的情况下应该怎么处理多智能体问题呢，可以用博弈论。

在有沟通的情况下，可以用投票，或者让第三方来居中裁判，或者彼此沟通协商

为什么这一点很重要，因为有的agent可能不会公布真实想法，所以集中式的合作可能并不会起作用。也能在保护不被恶意agent伤害。

分布式，如果某些agent需要公用同一数据结构，那么agnet可以被不同的人设计，并且彼此间可以独立，

为什么分布式很重要，因为可以根据现有信息做局部决策。没有中心决策者。

第二节：反应智能体

理想的理性智能体应该在任何情况下都能最大化表现（基于其感知序列（比如病人的问诊，环境变化）和内置知识做出决策）。

需要实时作出反应，世界动态变化，输入很多，**如果在一个动态变化的世界中应该用哪种机器人？**在面临许多不同的不确定状态转移和奖励时，深思熟虑代理的决策过程变得相当复杂。

传统的MDP需要完全知道转移矩阵的信息，而Q learning不用，因为可以通过估计来算出来。

如果动作空间很多很大的话应该用反应agent

Reactive agent：不需要考虑太多，反映时间快，考虑眼前的情况就行，与deliberative agent形成鲜明对比。比如自动避障车。需要实时反应，反应太慢的话甚至比错误反应还worse。

图示

描述已自动生成

Stimulus-response

无记忆、不会随时间planning，不考虑环境，仅根据stimulus.比如温度感知机器人，

文本

描述已自动生成

在MDP中，Assumption: Observations determine state with certainty.

而在用value iteration 或者 policy iteration的时候需要在有限空间内，也即状态和动作空间都是有限的。

怎么解决一个MDP问题，用value iteration，怎么解决呢？

MDP总结：状态转移是随机的，且每次到一个状态都能够确定当前状态。

在转移矩阵（包括状态以及reward）已知的前提下，可以计算价值函数

**文本, 信件

描述已自动生成**

可以看出来，价值函数是和策略挂钩的，不同策略下的价值函数肯定不同，在给定策略的情况下，怎么计算价值函数呢，一个可以用矩阵形式来直接进行计算，但是由于要计算逆矩阵，计算量太大，所以可以使用迭代，取一个V0，然后不断迭代得到Vk+1

**当然，仅能计算价值函数是肯定不足够的，要计算最优的价值函数。**

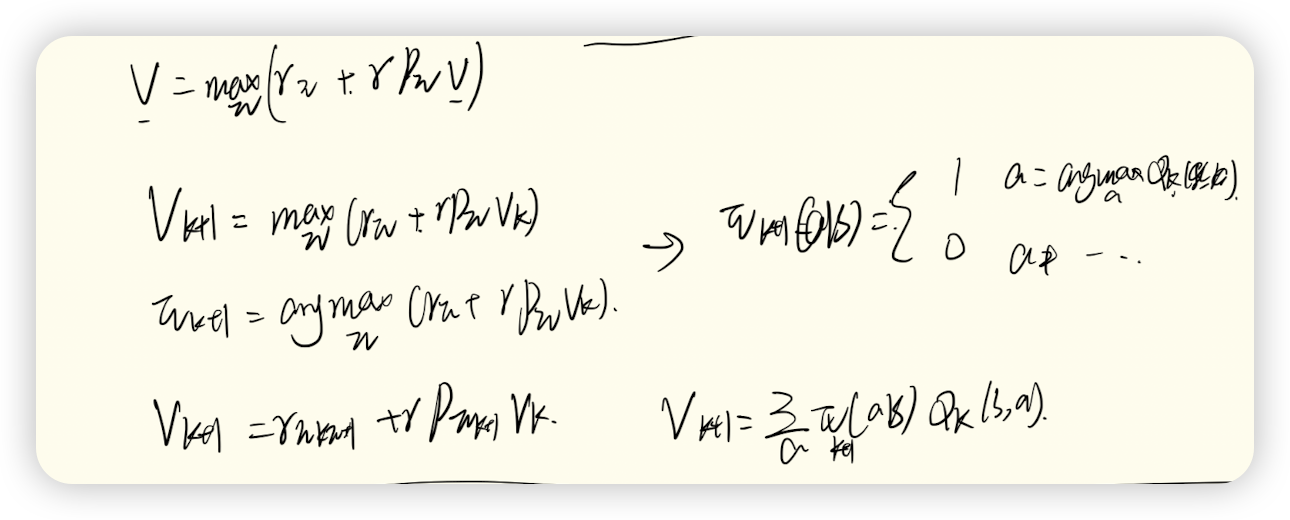
**什么是最优价值函数呢，就是找寻最好的策略，根据这个最好的策略算出来的价值函数最大。**

怎么计算最优价值函数呢？最优从何而来，自然是策略最优，直觉理解策略就是在给定当前状态的情况下选择s的概率分别为多少？

最优状态价值函数和最优动作价值函数在数值上是相等的，因为非最优的动作会得到0的概率。

通常来说，有value iteration和policy iteration。

Value iteration介绍，其实与以上计算状态函数的步骤有点类似，但是这边取了一个最大值



直觉理解就是每次选择最大的动作价值函数，初始化一个V0，然后不断取最好的策略，得到下一步v1，再继续迭代。

为什么可以保证收敛呢？不动点定理

Policy iteration

首先从一个random的policy开始，因为动态性质已知，所以能够算出动作价值函数，然后最大化动作价值函数。

以上内容都需要建立在动态特性已知的情况下，但是很多情况下，动态特性是不知道的，这个时候就需要使用DP算法，也即replay。

Q learning就是DP的一种，可以在并不知道动态特性的情况下使用

POMDP

也即部分可观测的MDP，为什么是部分可观测呢，因为不能准确地识别出当前所处的状态，实际应用中包括自动驾驶汽车，即使包括摄像头也不能百分百确定自己所处的状态。这就是与MDP最大的区别所在。

**可以将POMDP转换成belief MDP,再利用MDP的方法来求解。  
什么是POMDP，就是当你到一个状态时候不能很准确地清楚所在的状态，只能观察到一部分情况再根据观察到的内容来推断你所在的情况。**

**也即原来是P(s\_prime|a,s)变成了P（s\_prime|a,o,b），其中o是观察而b是belief。**

**文本

描述已自动生成**

**给定a和b的情况下转移到下一个信念b\_prime的概率，等于给定a和b的情况下观察到o的概率的加和。**

**在处理POMDP时，如果有n个状态则信念状态是连续且通常是n维的，且连续在01间**

**文本, 信件

描述已自动生成**

**Belief mdp的价值函数是分段线性的，也就是在不同阶段可以有不同的斜率。**

**可以用值迭代和策略迭代来解决belief mdp**

**MDP存在不少问题，只适用于解决比较简单（即状态空间小）的问题。并且需要动态特性已知。**

 **模型类型**：

* **MDP**：是一个完全的模型，定义了状态、动作、状态转移概率和奖励函数。它提供了一个确定的框架来描述环境的动态变化。
* **Q-learning**：是一种无模型（model-free）方法，不需要事先了解环境的转移概率和奖励函数。Q-learning 通过与环境交互来学习 Q 值，从而找到最优策略。

 **学习过程**：

* **MDP**：通常假设智能体可以通过动态规划等方法，利用已知的模型来计算最优策略（如价值迭代或策略迭代）。
* **Q-learning**：通过不断地与环境交互、观察结果并更新 Q 值来学习策略。它使用探索-利用策略来平衡学习新知识和利用已知信息。

 **更新机制**：

* **MDP**：在已知状态转移和奖励的情况下，可以直接计算值函数或策略。

**Exploitation and exploration中间有一个折衷。**

Q learning是一种model free的方法，也即不需要知道动态特性和奖励函数，不段更新就行。

**Exploration是为了减少未来的regret，而exploitation是为了减少当前regret。Regret是当前行动产生的价值与最优policy的payoff，最优policy是固定的，只有一个，也即只能和一个baseline做比较。**

**每个状态下的动作都不变，也即一直选择那一个动作。**

**为什么需要Q-table，因为动态形状不清楚，所以只能通过观察来不断更新参数。**

**MAB问题，有很多根杆子，每个杆子的分布不一样，应该怎么拉动杆子得到最多的钱？**

**Exploitation和exploration。**

**通常有好几种方法来解决，比如epsilon-greedy，UCB,TPS**

**epsilon-greedy**

**累积regret (O(cεt)).**

**UCB，其实就是期望加一个置信区间，置信区间怎么计算？**

**如果某一个动作的上限比另一个动作的下限还低，那么就可以把这个动作丢弃掉，因为只在很小的概率内这个动作可能更优**

**并且出现次数比较小的动作更容易被play**

**在UCB中，上限其实也是被bounding住的，**

**好处，不需要知道其他杆子分布是什么样的，只需要观察自己这个杆子的回报就行。**

**文本

描述已自动生成**

**直观理解就是，大于平均值加一个数的概率很小**

**累积遗憾与次数成log关系，一直是大于0的，比如在MAB问题中，如果有一根杆子就是显著优于其他棒子，那么理论上就应该每次都选这根杆子。**

**汤普森采样**

**为什么需要这个采样，因为UCB算法过于简单，同种类型的动作应该会有比较类似的reward，所以可以将协方差考虑进来，但是UCB不能做到这一点。**

**汤普森采样是是一种贝叶斯方法。其实就是先对每个动作先采一部分样本，然后之后每次都从已经采样的样本中再采样一个，选取回报最大的动作。理论上的Bound不如UCB，但是实际上表现地更好。**

**例子，一个人看了很多电影，并给出了评分，那么就对这些数据建模，并从模型中预测一个最有可能的电影给他。**

**如果reward是可变的呢，也即对抗性算法。**

**需要将这种对抗setting当成worst case**

**对抗环境下最好使用随机性算法，因为不易被预测，如果你使用确定性算法的话，在对手知道你的规律的情况下你再无赚钱可能。**

**完全随机的话，你的遗憾大概是（k-1）/k,对手也无从预测，但是这样遗憾又太大了**

**可以使用遗憾匹配，就是计算选择动作a相比于选择当前动作的差值，并加总就得到了累积遗憾。**

#### 1. 反事实后悔

* **定义**：
  + 对于动作 aaa 在时间 ttt 的反事实后悔定义为：

l(a)t=r(a)t−r(πt)tl(a)\_t = r(a)\_t - r(\pi\_t)\_tl(a)t​=r(a)t​−r(πt​)t​ 其中：

* + r(a)tr(a)\_tr(a)t​ 是在时间 ttt 采取动作 aaa 时获得的奖励。
  + r(πt)tr(\pi\_t)\_tr(πt​)t​ 是根据当前策略 πt\pi\_tπt​ 获得的奖励。
* **解释**：
  + 这个反事实后悔度量了如果代理在时间 ttt 选择动作 aaa 而不是根据策略 πt\pi\_tπt​ 所采取的动作时，可能获得的额外奖励。

**也就是说无论你选择哪个动作，相比当前的策略来说的regret都是一样的，也就是不会被针对。无论选择哪个动作，当前动作的累积遗憾不会变，其他的可能变，要根据正回报选择概率最大的那个。**

**文本

中度可信度描述已自动生成**

**Regret需要观察到所有的回报**

**MWU:需要观察到所有回报，而EWU只需要当前动作的回报就行，且更重视哪些选择概率小的动作**

**Q learning需要所有的动作和状态都被充分访问到，如果存在状态不能被访问到，则Q learning可能会有问题。**由于状态转移的性质，观察到的状态之间不是独立的。状态 st+1s\_{t+1}st+1​ 的值依赖于状态 sts\_tst​ 的值和所采取的动作。这种依赖性破坏了随机梯度下降的假设，导致在学习过程中可能出现收敛性问题。

**Context bandits**

**将好几个特征组合一起成为一个context，给每个context一个老虎臂，可以减少需要的数据。图片包含 文本

描述已自动生成 Q learning 和 SARSA**

**图形用户界面, 网站, 日程表

描述已自动生成**

**第4节**

**Reactive agent，反应快，不需要plan over time，但是相应的对每个state都需要有相应的反应存进来，但是现实生活中有很多都是不需要的。**

**Deliberative solution of dp,仅考虑当前状态以及后继，局部搜索可能不一定能找到best path,需要去遍历所有的可能性，也说明状态空间可能会指数型增加。**

**DFS和BFS的伪代码，可以看出来只有队列那一步不一样，BFS是先进先出，而DFS是后进先出**

**文本

描述已自动生成**

**但是DFS也有问题，就是如果在一条错误的路上一直走下去就会浪费时间，所以现在再加一个DLS，就是没多探索一步的话将限制I减去1。当I为0的话就不把那个点加到list里面了。**

**还有个问题，应该怎么确定I的大小呢？**

**用迭代算法，初始化一个I,然后每次增加I的大小，看能否实现目标，这样的话虽然复杂度会增加，但是也不会超过两倍，比你盲目猜的效果会好很多。**

**A星算法的最优性证明**

**也就是H永远都比实际情况要小，不能比真实世界中的距离还长。为什么，比如直接将h设置成0，那么就可以保证肯定是最短距离，**

**启发式函数不能高估cost，就能保证最优。**

**但是那个Q可能会比较长，但是尽管比较长也不能使用beam search或者迭代增加个数，因为还是可能会错过最优方案。**

**Minmax算法**

**从最底下一层来推，自己取max，对手取min**

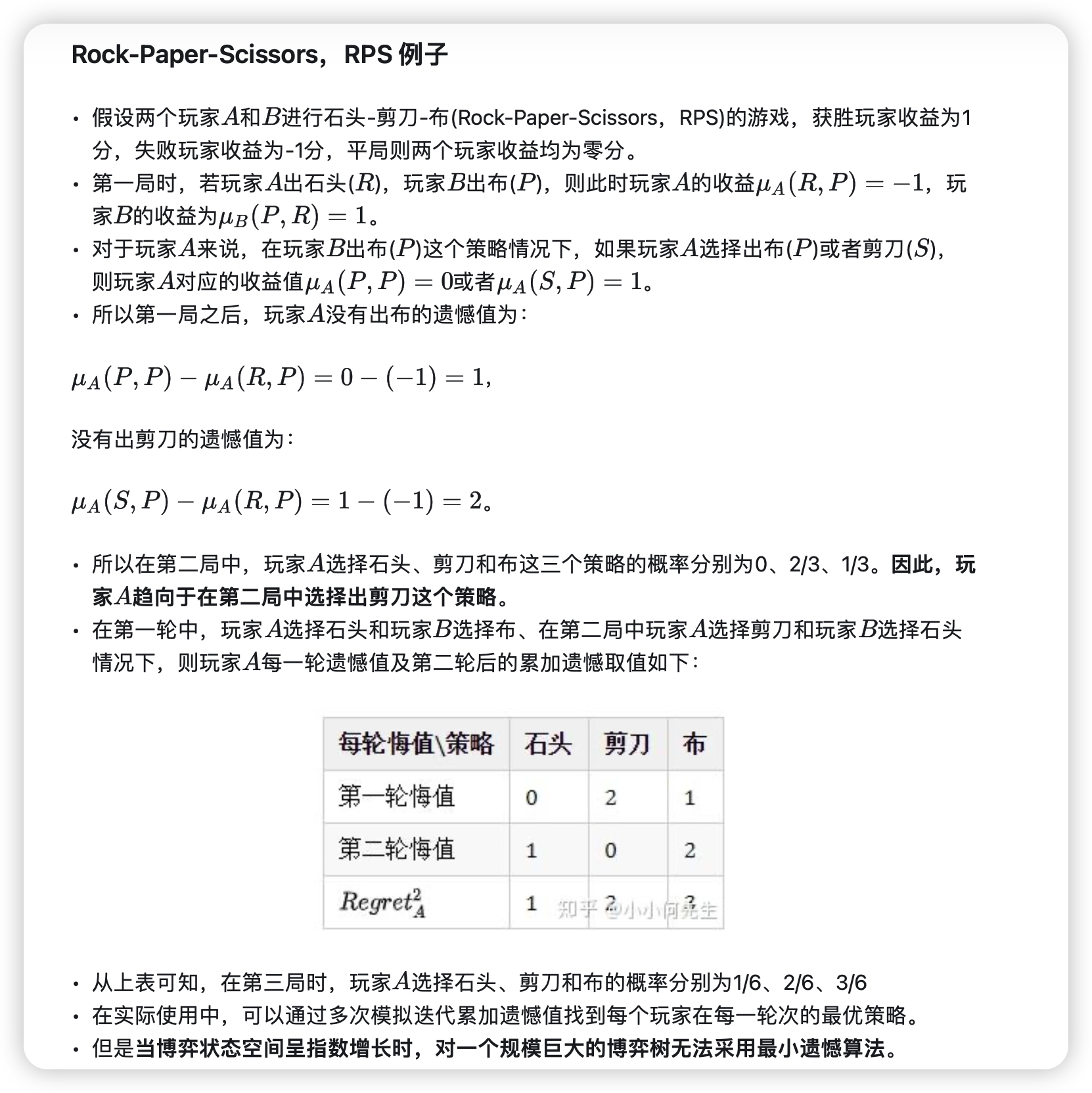
**但是这样的话有个问题，当步数太多时，需要存储的空间太大，需要进行alpha beta剪枝，看PPT**

**在下棋的时候可以用蒙特卡洛搜索来模拟下一步可能出现的情况，但是有很多步都是无用的，所以需要用更有效的方法，来选择出那些更加promising的。**

**蒙特卡洛树搜索，不需要穷举所有的结果，而是随机跑完一次结果，然后重复很多次，在处理大规模数据时有优势。**

**反事实遗憾最小化算法，其实就是用来更新下一个动作的概率。**

**反事实遗憾最小化算法用来解决随机性较强的问题的。**

****

**遗憾最小化算法与反事实遗憾最小化算法的区别是什么？**

**反事实遗憾最小化通常考虑以往的所有信息，如上。**

**遗憾最小化只考虑当前步骤。**

**什么是信息集？**

**就是有些信息只有一个agent知道。**

**第5节**

**为什么需要factored representation（分解表示）**

**能够更有效地解决复杂问题。**

**在之前那个例子中，6个行李，加上机器人，每个object都可能有6个位置，再加上持不持有，状态数目就很多了，但是实际情况下，我们发现有很多状态是不能达到的。**

**那么就可以进行分解，只care那些对我们当前目标比较重要的factors。这能够很大程度上降低复杂度。**

**首先是对状态转移进行factor，只选取重要的进行factor**

**文本, 信件

描述已自动生成**

**然后可以将策略进行分解，比如有些策略很复杂，但是分解一下之后就简单了。**

**回报函数也可以进行分解：pos(1), pos(A), destination(1)**

**状态价值函数分解：**

**图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成**

**然后也能用价值迭代来计算最优策略，但是需要等式的个数大于未知数。**

**什么是情况演算，就是只考虑重要的那些状态**

**举个例子，将包裹A从一个地方移动到另一个地方**

**此时其他包裹的位置就不重要，可以将他们看成等同状态。**

**最小承诺原则，就是对action的顺序不做规定，保留灵活性。**

**图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成**

**非线性planning，**

**就是有些动作需要按照特定顺序，但是其他的不需要。比如穿鞋需要在穿袜子之后，但是穿袜子和穿裤子就没有顺序。可以并行**

**怎么使这种非线性planning更有效呢？**

**就是得首先识别出这种有顺序的group，对可以并行的就先并行。**

**怎么识别呢，依靠pre和postcondition**

**通常偶数时间表示状态，奇数时间表示动作**

**第6节**

**强化学习不够优秀，因为显示情况中状态以及动作空间太大，所以可以考虑用深度强化学习。**

**怎么估计value function呢？**

**如果状态可以观察到的话，就直接将观察到的东西和选择的动作当作输入CNN来预测下一个状态。**

**如果状态只是部分可观察的话，需要用RNN，因为单独用当前的输入是不够的，还需要之前的信息**

**有时候奖励要在整个trajectory都完成后才能观察到，所以可以用MC方法来进行模拟。**

**策略梯度，其实就是给策略一个参数，使得其可以被求导，再用梯度下降来求解。**

**怎么采样trajectory呢？**

**只需要采样一步，得到即时reward.**

**Actor-critic**

**文本

描述已自动生成**

**还可以用多任务学习，用同样的观察与动作来预测value与下一步动作。通过共享特征与权重能够很有效地提高效果。**

**用SGD来对策略做梯度下降可能效果不会太好，直觉理解就是希望变化较小**

**文本

低可信度描述已自动生成**

**并且通常training data不会包括比较差的状态，所以我们同样需要offline，包括重要性采样（从已有分布中采样出未知分布）以及外推，**

**第7节**

**多智能体系统，就是有多个智能体的系统，确实有这个需要，一个agent不太够。**

**中心化的决策机构太脆弱了，比如会有贿赂这些，**

**中心化：所有的agent在一个平台上运行并且共享内存**

**Mediator：各agents通过一个居中的mediator交换信息**

**分布式，各agents通过各自的方式交换信息**

**Joint plan first incentives later**

**两种优化理念1，任何agent的最小奖励;2所有的agent的奖励之和。**

**也有online和offline两种方式，**

**Offline，每个agent算出自己的各个动作的奖励，然后用线性规划组合**

**Online，各agent还考虑其他agent的动作，最后选个最优的动作组合。**

**中心的缺点：缺乏并发性（concurrency）**

**博弈论**

**Game，就是许多agents根据组合的action来获得收益**

**有的策略中，某个选择的收益就是会比较高，这就是主导策略**

**图示

中度可信度描述已自动生成**

**（严格）支配策略**：对于其他玩家的每个行动，该策略总是严格优于任何其他策略。  
**弱支配策略**：对于其他玩家的每个行动，该策略至少与任何其他策略一样好，并且在至少一种情况下严格优于其他策略。  
**非常弱支配策略**：对于其他玩家的每个行动，该策略至少与任何其他策略一样好。

**A** 和 **B** 的选择在这个状态下互相稳定，没有任何一方能通过改变自己的策略来获得更好的结果。

纯策略，就是确定性的动作，我只选一个动作，那么另一个也只能选一个动作，并且不能双方都不能达到更优了

无纯策略，就是并不100%选这个动作

**在0合游戏中，如果都用minmax的话，那么一方会亏另一方会赚，且这个绝对值相等。**

**如果一个动作的预期收益小于v，那么不会选择这个工作，因为选择其他动作能拿到v。**

**人们有时候是风险厌恶的，有时候又是风险爱好的，很正常。**

**支付是客观的，而效用是主观的，效用确实和支付相关，但也并不是决定性因素，还取决于风险厌恶程度。**

**拍卖理论**

**有好几种价值类型**

**Private:只和agent的preference有关，比如haircut**

**Common:完全由市场决定，比如金价**

**Correlated:部分由自己，部分由市场，比如毕加索的画**

**表格

描述已自动生成**

**图片包含 文本

描述已自动生成**

**文本

描述已自动生成**

**图片包含 文本

描述已自动生成**

**一定要是私人价值，就是很确定知道价值。**

**第二价格能保证以真实价格出价，为什么？**

**风险厌恶的竞标者** 更倾向于选择那些不涉及投机的拍卖形式，如 **荷兰式拍卖** 和 **差别定价拍卖**，因为这些拍卖形式的风险较低

收益不等性

文本

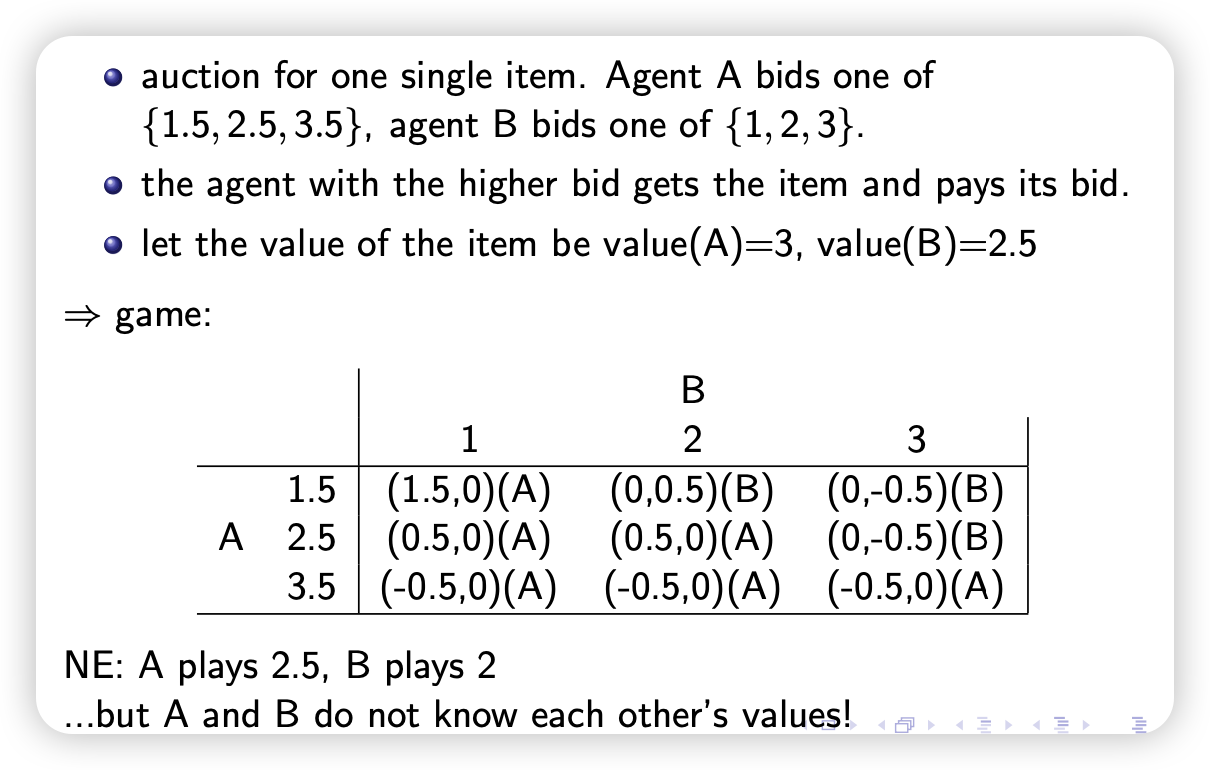
中度可信度描述已自动生成

多units维客利拍卖理论，

如果有n个物品要出售的话，每个agent都出第n+1的价格

真实世界博弈

如果在知道对方的效用的情况下，是可以很简单的达到纳什均衡的

****

**比如在这个例子中，不论A选什么，B都会选2，在B选2的情况下，A肯定会选2.5。**

**但是假如在不清楚对方效用的情况下呢？**

**Ex ante,其实就是根据期望来算平均效用**

**表格

中度可信度描述已自动生成**

**Ex post，信息在游戏结束后完全知道，那么第二拍卖价格是一个纳什均衡。**

**文本

描述已自动生成**

**注意：纳什均衡不一定会是最优的策略。**

**通常来说，合作的话会更容易实现最优策略，但是相应的，双方需要互相信任，并且有些时候还需要引入惩罚机制或者建立契约。**

**比如囚徒困境，纳什均衡是双方都坦白，但是合作的话是双方都背叛。**

**所以需要签订契约，如果不遵守的话要被punish，被punish的价格需要大于因为不遵守契约而获得的收益。**

**可以要求一个mediater来行动。**

**表格

描述已自动生成**

**Mediater是知道双方信息的。**

**还有些问题是没有纳什均衡的，比如石头剪刀布，只要我知道你的策略我就是可以赢你。这时候就需要混合策略纳什均衡。**

**文本

低可信度描述已自动生成**

**但是有些时候混合策略纳什均衡不一定能实现最优。**

**相关策略：<https://zhuanlan.zhihu.com/p/193127814>**

**可以通过谈判达到比纳什均衡更好的效果，但是需要支付side payment.**

**如果只有一轮谈判的话，那么propose方是有优势的。**

**如果有多轮谈判的话，那么最后一次propose的有优势，之前的都无关紧要。**

**文本

描述已自动生成**

**Multi agent的问题**

**协调（中心化和去中心化，强化学习，投票）**

**合作**

**distributed implementation（程序有共同的数据结构与算法，并且很显然每个agent都是分布的。能够保持分布性和多样性）**

**怎么处理self-interest,**

**Voting**

**Mechanisims:让第三方来参与，比如拍卖**

**Negotiation，协商**

**什么是POA，就是不协调的成本与协调的成本之比**

**实时机器人**

**反应太慢比不反应更差**

**MDP（动态转移已知）**

**POMDP（对当前状态不确定）**

**Q-LEARNING(动态转移不知)**

**Deliberative agent(动作和状态很多)**

**记得把公式推导一遍**

**需要做exploration和expoitation的trade off,**

Here is the empirical result, we applied clustering on both synthetic data and a real-world dataset to compare the methods.

First, we tested our model on the moons dataset. As you can see, it's non-convex, which makes it a great dataset to show off the advantages of spectral clustering. We got the results we were hoping for, with high AMI and low variance. This means the model not only performs well, but it's also stable.

Secondly, since real-world problems can be more complex, we chose MNIST, a dataset of handwritten digits. We selected 5 digits and then split it into training, validation, and test sets with equal sizes. To make the results more visible, we transformed the pixel images into 2D data using t-SNE. As we can see, the shape of the clusters is irregular.