midterm

搜索

全局搜索

问题的定义

1. 关键：把问题用一个统一的模型表示清楚，就能够用通用的搜索方法来求解
2. 什么问题可以搜索：
   - 初始状态S_0
   - 可行动作 给定状态s，$ACTIONS(s)$：同一个问题可以有不同的描述。建模很重要
   - 状态转移模型 $RESULT(s状态, a动作)$（1-3：构成状态空间）
   - 目标状态
   - 路径花费（衡量解的质量。最优解）
3. 问题例子：路径选择、八数码、八皇后……

搜索对问题求解

1. 搜索树。关键：分清开节点集（frontier/open list，访问过没有展开）和闭节点集（closed list/explored set，儿子已经展开）
2. 关键：树搜索（不记录已经搜索过的节点）、图搜索（从开节点集中拿元素后检验是否已在闭节点集中，避免重复搜索/死循环）
3. 搜索算法的评价标准/衡量指标：
   - 完备性：存在则找到
   - 最优性：找到最优解
   - 时间复杂度：用展开节点数目表示
   - 空间复杂度：存储最大节点数目估计
4. 问题的难度衡量：
   - 图规模：状态空间（|V|+|E|）
   - 树的规模：b分支数、p最浅目标深度

无信息搜索

1. 任务问题来了用相同的步骤进行搜索
2. 关键：不同策略的区别仅在于谁先从开节点集出来，不同的出来顺序使用不同的数据结构实现

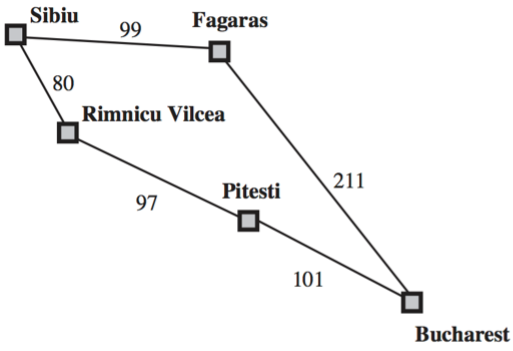
	广度优先	**深度优先**
定义	优先展开在树上层次最浅的节点，层数相同时，按照一个既定的顺序展开。	按照子结点的既定顺序，从左到右，优先展开最左边的子节点，一直到找到目标，或者没有子节点可以展开。此时返回到**上一级**节点，展开后边的儿子节点。
是否完备	是	否

	广度优先	深度优先
是否最优	是	否
时间复杂度	$O(b^d)$	$O(b^m)$
空间复杂度	$O(b^d)$	$O(bm)$（记录路径上的点）m最大深度。关键：深度优先更省空间，搜的更深
实现	开节点集：队列：push&pop；闭节点集：哈希表	开节点集：栈。递归调用。

3. 深→深度受限搜索（$O(b^L)$，$O(bL)$）
4. 深→迭代加深：一层一层但每一层用深搜。不知解在何处时最推荐
5. 宽→双向搜索 时间空间均为$O(b^{d/2})$
6. 宽+每步代价不一样→一致代价搜索：展开`当前路径花费 g（n）`最少的节点。记录初始状态到所有开节点开销，“用cost去排队”。使用优先队列存储开节点集。问题：初始到目标节点很远的时候可能很难收敛。

   注意：必须等所有代价更小的节点扩展完毕后，才能确定当前目标节点的代价是否最优。即每次只取最小者目标测试/展开关键：从开节点集出来时找到最优解，留下的花费更高

例：



顺序

1. 开节点集s，两个儿子R和F入开节点集；
2. 因为80<99，R出队，它的儿子P入队
3. 此时开节点集F（99）和P（80+97），因为99＜（80+97），F出队，它的儿子B入队
4. 此时开节点集P（80+97）和B（99+211），(80+97)＜（99+211），P出队，儿子B入队中，发现队列中已有B，但是新的花费更小，于是更改队列中B的花费为（80+97+101）
5. 此时开节点集中只有B（80+97+101），B出队，发现是目标，搜索结束，返回解。
6. 当B出队时，队列中已经没有比它更小的了，以后弹出的任何节点，再到B，路径都不会更短了。

这里需要一种神奇的数据结构优先队列

有信息搜索：根据估值函数确定被选择展开的节点

1. 贪心（贪婪最佳）：一致代价中用f（节点到目标节点的最短路金估计值）取代g。
2. 一致代价（低效）+贪心（可能非最优）→A* 搜索：f=g+h

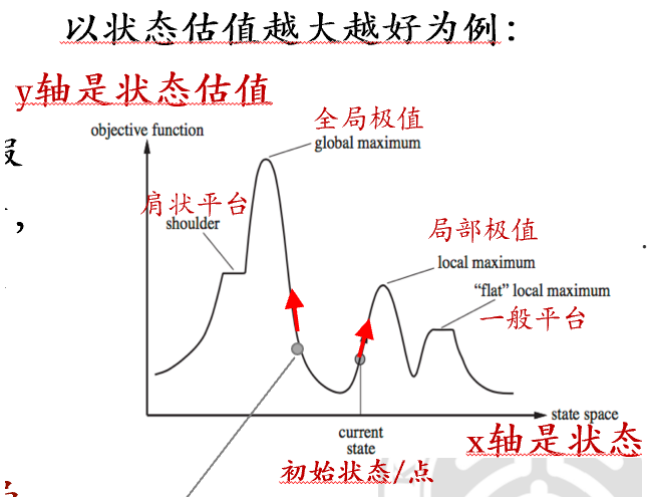
- h（n）关键：h(n)是可采纳的或者是一致的，A*就是最优的
  1. 可采纳：h≤实际花费。
  2. 树搜索（不会二次到达同一节点）：h（n）可采纳则A*最优。
  3. 一致的：$h(a) \leq cost(a,b) + h(b)$，则f在任意路径上是不下降的。图：最佳的。“即开始时悲观”。

     当你展开目标节点时，其他节点实际发生的和估计的之和都比它大。因为估计的是乐观的，所以其他节点如果真正展开，只会更大，所以找到的是最优解。

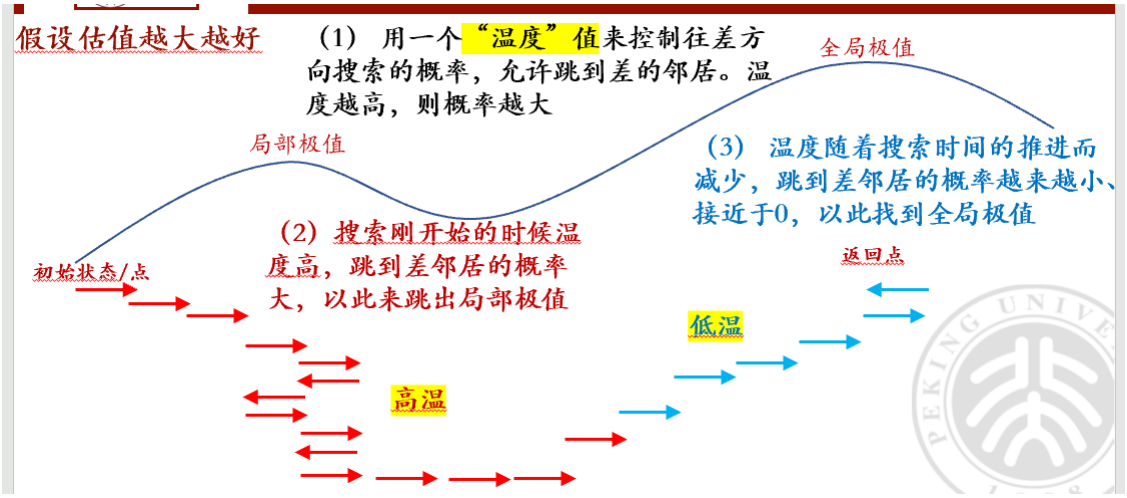
     极端情况：最好即贪心，最坏（全0）则一致代价
- 完备、最优、（针对f）最高效的。

3. 启发式函数
    1. 设计：松弛法：减少对动作的限制。状态图是原问题的超图（增加了边）→松弛问题的最优解是对原问题的一个可采纳的启发式函数
    2. 产生过程需简单
    3. 宽度优先 -> 迭代加深 -> 一致代价 -> A*

局部搜索

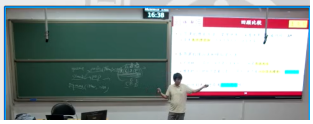
1. 只找一个可行解，不关心得到解的路径。局部优化。启发式函数→状态估值函数
2. 求解过程：当前状态→（状态转移）→邻居状态.....目标状态
3. 

4. 初始点的位置影响了能不能到全局最优的极值点。跳出局部极值？平台？
5. 完备的，如果目标存在则总能找到；最优的，算法能找到全局最优解。
6. 爬山法
    1. 最陡下降：只存储当前节点和一个估值函数
    2. 随机平移：可脱离肩状平台。防死循环设置次数上限
    3. 随机爬山：概率选择最优邻居。往往比最陡爬山法有更优解，但收敛速度变慢。
    4. 邻居多则估值表生成慢.....→第一选择爬山法：随机找一个邻居，计算它的估值，若它的估值比当前状态要好，则立马跳到那个邻居上；若这个邻居估值没当前的好，则随机再选择一个邻居做相同的操作。
    5. 不完备.....→随机重启爬山法"总有一个初始状态是有解的" 期望重启次数`1/p`
7. 爬山（不完备）+随机游走（低效）→模拟退火算法：如果领居更好肯定跳过去。如果领居没当前状态好，根据一个概率来跳转。



超参数：步幅（刚开始的温度）、温度下降速率、降到什么程度可以报告解

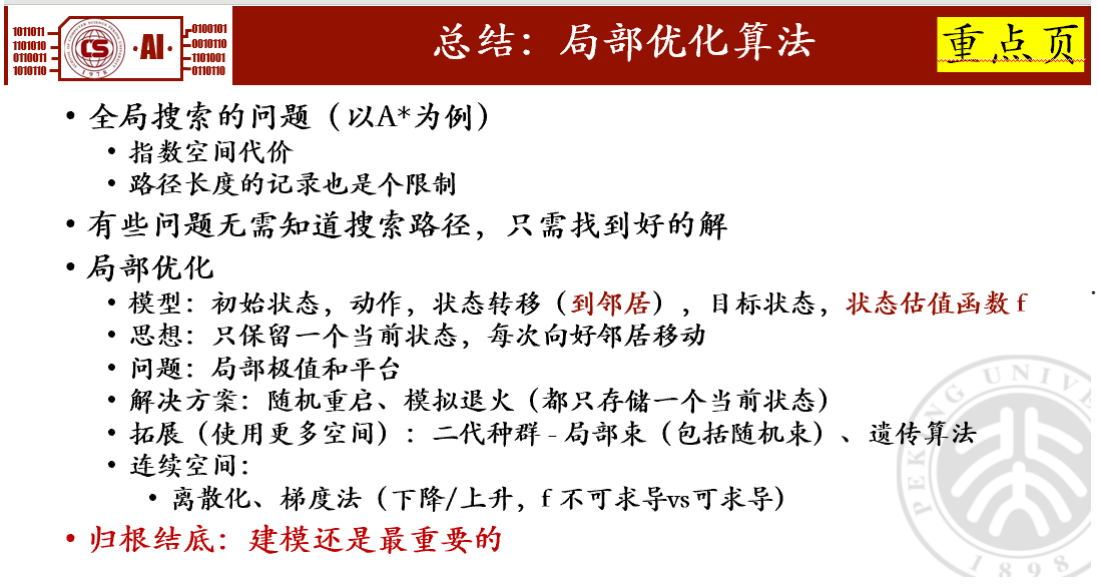
8.
- 上节课的搜索方法（宽度优先、深度优先、一致代价、A*）
  - 需要展开搜索树，占用内存太大，分支太多
- 本节课的爬山法和模拟退火法
  - 不用树了，节点之间不是父子关系了，而是邻居关系
  - 搜索时一个时刻只需要记录一个节点，则只用一个节点搜索 - 矫枉过正
- 问：如何用更多内存来加快搜索呢？



→局部束搜索：保留k个状态，生成kb个后继，再选其中最优的k个继续。

9. 仍然容易都到局部极值……→随机束搜索“自然选择”
10. 仍然没有解决局部极值问题……→遗传算法（随机束算法的变种）：两个父状态结合产生后继节点。
    - （a）中的状态（称“人口”）的得分，后被用来作为该状态被选择产生子代的概率
    - 根据任务要额外设计交叉配对Crossover&变异Mutation算法
    - 模式schema：某些字符串可以有一部分空置着：如 `246*****`，具体的串成为模式的一个实例，如 `24613578`
11. 连续状态空间中的局部搜索

    经验梯度（不可求导时）；梯度下降



强化学习

1. 斯金纳箱实验揭示了**通过反馈信号迭代调整策略**的机制，直接对应强化学习的核心框架：
    1. **智能体（Agent）**：对应实验中的动物，通过与环境互动学习行为策略。
    2. **环境（Environment）**：对应斯金纳箱，包含状态、动作和奖励信号。
    3. **奖励函数（Reward Function）**：对应实验中的强化物或惩罚物，指导智能体调整行为。
    4. **策略（Policy）**：对应动物通过试错习得的 “按压杠杆” 等行为模式，即从状态到动作的映射。

    现代强化学习算法（如 Q-learning、策略梯度法）均基于 “行为 - 奖励” 的反馈机制，可视为斯金纳理论在计算机领域的延伸。

2. 环境：
   1. 初始状态 S_0(state)
   2. 当前玩家C（current player(s)）
   3. 动作 A(action)
      - 智能体在某个状态下的合法动作集合。
   4. 状态转移 P(transition)　$P(S_{t+1}|S_t,A_t)$ 用以表示环境
      - 衡量一个环境的**复杂程度**：某个状态下，智能体采取某个动作后，转移到下一状态的状态转移模型。可能到达的所有状态构成了**状态空间**（state space）。所有状态下可行动作，构成**动作空间**（action space）。
   5. 终止状态 S_T(terminate state)
   6. 奖励 R(reward) $R_t \leftarrow S_t, A_t$
      - 某个状态下，智能体采取某个动作后得到的分数。

3. 智能体：
   - 策略 π：$A_t \leftarrow \pi(S_t)$ 用以表示智能体
     - 状态 S 到动作 A 的映射关系，给出了智能体在状态 S 下如何选择动作 A 的决策方法。
   - 注意：策略π是全局性的，任何状态下都要能够给出动作选择
   - 目标（问题的解）：
     - 寻找**最优策略π**，使得从初始状态 S_0 到终止状态 S_T的累积收益 G（gain）$=\sum_{i=1}^{T} R_i$ 最大

4. 小结：
   - 将对手建模在环境里；每次采取动作后面临的状态都是对手执行完它的动作后的新状态；（也可以建模成多智能体博弈问题，有一个对手决策模型，轮到对手落子时让对手模型决策）
   - 用值函数表存储状态估值/值函数表V(S)
   - 通过不断对弈更新值函数表
   - 根据值函数表、可以得到贪心选最优动作π*
5. 问题模型的泛化（带概率的P,R,π）、状态价值V_π和动作价值Q_π
6. 寻找最优策略的思路（多臂老虎机为例）
   1. 计算动作价值q(s,a)，更新
   2. 贪心（选最高的）→ϵ-贪心（大部分时间贪心，偶尔随机选）→乐观初值贪心（鼓励探索，只需要贪心算法）
   3. UCB：当前估值与新鲜程度取加权和　$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$，
      $N_t(a)$：a 在时间t前被选中的次数，$c > 0$ 控制探索的度。
   4. 梯度下降：给每个动作a一个数值的优先度，影响动作选择概率。softmax归一化。
7. 强化学习中最大的挑战：
   - 平衡探索和利用的关系
   - 要最大化某个目标，就要选择好的动作，
   - 但是要发现这些好的动作，又要去尝试那些未知收益的新动作。
   - 数学家们已经在这个问题上探索了很久，依旧没有解决这个问题。

LLM

1. R1是推理模型。重复问题→理解问题→初步思考→自我提问→回忆以往策略→提出质疑→猜测/探索→举例初步验证，但存疑→先考虑能否实现→重复念叨→整理思路（分层次）→存疑，所以举例验证→发现错误，再举一例，再错误→探索新方案并质疑→得出验证结论→再换情况举例，得出验证结论→尝试总结
2. 第一个阶段：Pretraining：学会语言

   构造：给前面n-1词构造第n个词；给前后构造中间

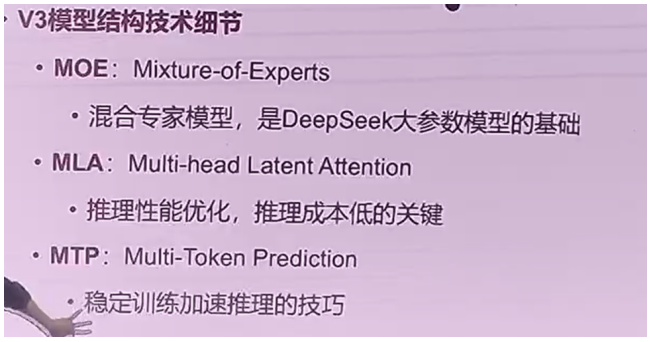
不可能穷举所有的人类语言……大致对其输入语言与输出语言

$$m\text{个输出词序列} = f_\theta(n\text{个输入词序列})$$

3. 第二个阶段：Post-training：适配到人类迎用场景

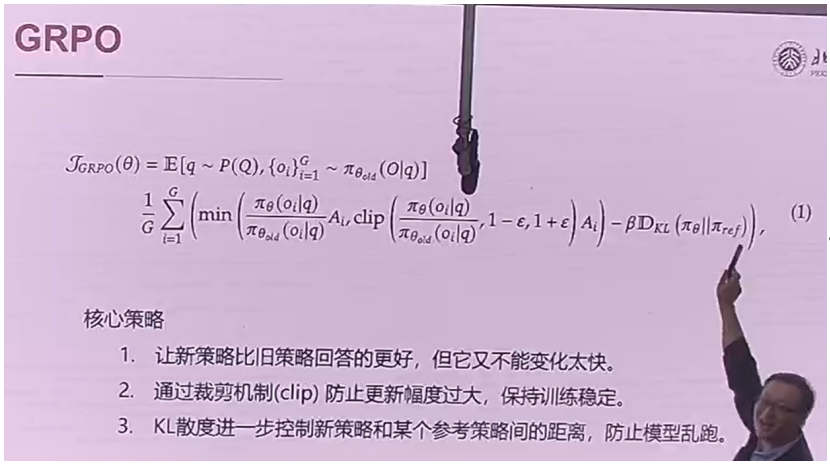
（1）使LLM更加适合于各种人类沟通场景中的表述习惯；
（2）使LLM更加对齐到人类的期望和价值观；
（3）让LLM在一定程度内“学”某些领域的“知识”；
（这个目的能否达到，是存在争议的）
（4）对模型进行进一步处理，如在保持性能的同时缩减参数等；

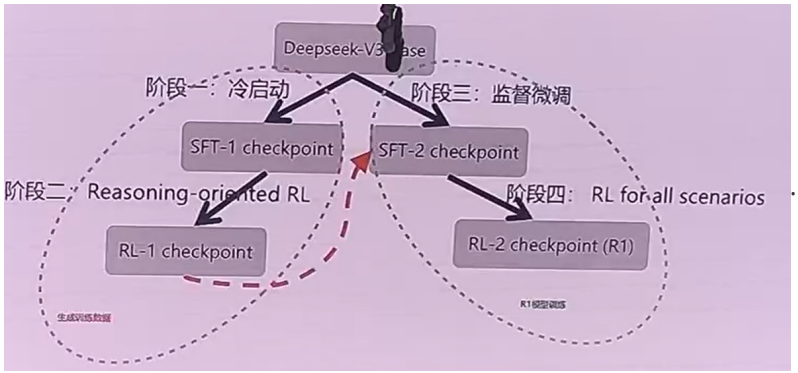
4. 第三个阶段：TTS（Test Time Scale/Compute
5. GPT-3 的训练过程……
6. DeepSeek-V3模型：



   - MOE：每次激活模型的一部分（称为Expert）dense model→sparse model。
   - MLA（KV Cache的高强度压缩）
   - MTP（多次预测）：每次预测3个token的生成概率（只再过1个block预测再下一个词（next next one token）），获得约1.8倍加速。

7. 先V3经过纯强化学习（准确性奖励+格式奖励）训练DeepSeek-R1-Zero，再经过4步（见下）得到R1



8. 

   1. SFT_1 冷启动
   2. RL_1 Reasoning Oriented
   3. SFT_2 Rejection Sampling 提升推理能力，对V3base有监督微调
   4. RL_2 提升有用性&无害性

9. distill模型：精华的数据用于训练

10. TTS的例子：自己出题自己证明；SAT表达式控制题目难度……

具身智能

1. 定义：被赋予物理身体的智能体，通过感知理解物理世界、做出规划决策 并与环境交互从而完成任务或解决问题的能力。
2. 难点
    - 任务变：从开环变为闭环
    - 赋能基础变：从数据集到数据集+仿真环境
    - 标准化难

    莫拉维克悖论
3. 研究目标：构建具身智能多模态大模型与大数据，实现具身智能Scaling Law

~~不学了。~~

- AI总结：
    - **具身智能概述**
        - 定义：物理智能体感知、决策、交互能力
        - 发展历程
            - 机器人1.0：专用机器人时代
            - 过渡期：向通用过渡，多模态大模型融合
            - 机器人2.0：具身智能时代，端到端多模态大模型驱动
        - 核心难点
            - 任务转变：开环到闭环，马尔可夫决策过程
            - 数据获取：机器人数据成本高，依赖仿真环境
            - 标准化挑战：本体、控制、感知标准差异大
            - 莫拉维克悖论：低阶技能计算复杂
        - 研究目标：构建多模态大模型与大数据，实现Scaling Law
    - **多模态大模型技术基础**
        - 算法支撑
            - 模仿学习：数据驱动，学习状态-动作映射
            - 强化学习：奖励驱动，优化策略
        - 关键模型
            - Transformer：LLM主流结构，Decoder-Only
            - 视觉Transformer（ViT）：图像分块处理
            - CLIP：图文对比预训练，跨模态关联
            - Diffusion模型：正向扩散与逆向去噪
            - 多模态大模型（LLaVA）：处理多模态数据
    - **具身智能多模态大模型**
        - 大小脑协同框架
            - 大脑（慢系统）：高层认知，任务规划、感知、轨迹预测
            - 小脑（快系统）：低层控制，执行具体动作

        - 具身大脑大模型（RoboBrain）
            - 能力：任务规划、可操作区域感知、轨迹预测
            - 训练：四阶段训练，近千万条数据
            - 性能：超越基线模型，具身任务规划能力强
        - 传统VLMs挑战：长程操作能力不足，需专用数据集
    - **具身智能数据**
        - 数据金字塔
            - 底层：互联网数据，预训练基础模型
            - 中层：仿真数据，低成本生成，存在Sim2Real差距
            - 顶层：真机数据，质量高，采集成本高
        - 采集方案
            - 仿真生成：RoboGen、MimicGen
            - 遥操作采集：同构遥操作、异构遥操作
        - 大规模数据集：ShareRobot、Open X-Embodiment、RH20T
    - **具身智能大小脑系统**
        - 端云协同
            - 云端大脑：复杂推理、多机协作
            - 终端小脑：本地控制，实时反馈
        - 跨本体协作：RoboOS统一控制接口，多机器人协同
        - 应用案例：任务规划、可操作区域感知、轨迹预测
        - 开源资源：RoboBrain、ShareRobot、RoboOS