游戏人工智能

GameAI@NJUPT

陈兴国、徐修颖、杨光、吴毓双

2018-08-25

Related topics

- 大数据并行与交互式计算，陈国良
- 强化学习简介，俞 扬
- 多智能体强化学习中的博弈与均衡，高 阳
- AlphaGo原理与实现，朱天驰、刘 杰
- 卷积神经网络的前世今生，寇佳新

游戏（Game）

- 游戏定义
  - 游戏是一种具有某种功能的活动，具有两个最基本的特性：①以直接获得快感（包括生理和心理的愉悦）为主要目的。②主体参与互动。
- 总的来说游戏有四个特征：
  - 有趣
  - 不确定
  - 规则
  - 虚构

游戏分类

电脑游戏按内容分：

1. RPG角色扮演游戏（Role-playing Game）
2. ACT动作游戏（Action Game）
3. AVG冒险游戏（Adventure Game）
4. FPS第一人称视角射击游戏（First Personal Shooting Game）
5. FGT格斗游戏（Fighting Game）
6. SPT体育类游戏（Sports Game）
7. PZL益智类游戏（Puzzle Game）
8. RCG竞速游戏（Racing Game）
9. RTS即时战略游戏（Real-Time Strategy Game）
10. STG射击类游戏（Shoting Game）
11. SLG策略游戏（Strategic Simulation Game）
12. MUG音乐游戏（Music Game）
13. SIM生活模拟游戏（Simulation Game）
14. TAB桌面游戏（Table Game）
15. CAG卡片游戏（Card Game）

游戏分类

- Information
  - Perfect information： Chess, Go
  - Imperfect information: Card game
- Number of players
  - Single player: Tetris, 2048 Game
  - Multi players: Chess, Go, Card Game

目录

- 游戏的复杂度
- 玩游戏的人工智能
- 展望

进度

- 游戏的复杂度
- 玩游戏的人工智能
- 展望

游戏的复杂度

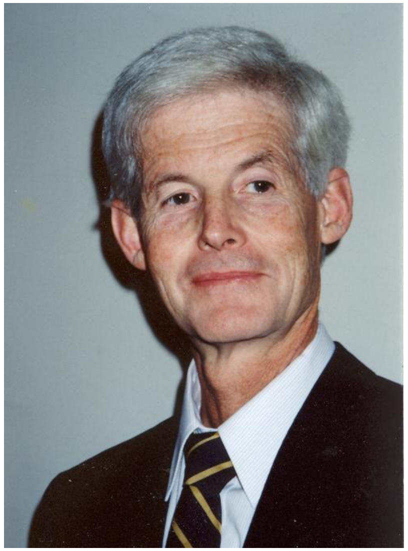
- 算法的复杂度
  - P
  - NP
  - NP-Complete
  - NP-hard
- 游戏的复杂度
  - Tetris的NP-Complete问题

算法的复杂度

- Class P
  - Decision problems that can be solved by a deterministic Turing machine that runs in polynomial time.
- Class NP
  - Decision problems that can be solved by a non-deterministic Turing machine that runs in polynomial time.
  - NP can be defined using deterministic Turing machines as verifiers.

P=NP?

- The hardest problem in NP
- Cook's theorem (1971)
  - The SAT problem (Boolean satisfiability)
  - The first NP-Complete problem
  - the problem of determining if there exists an interpretation that satisfies a given Boolean formula.

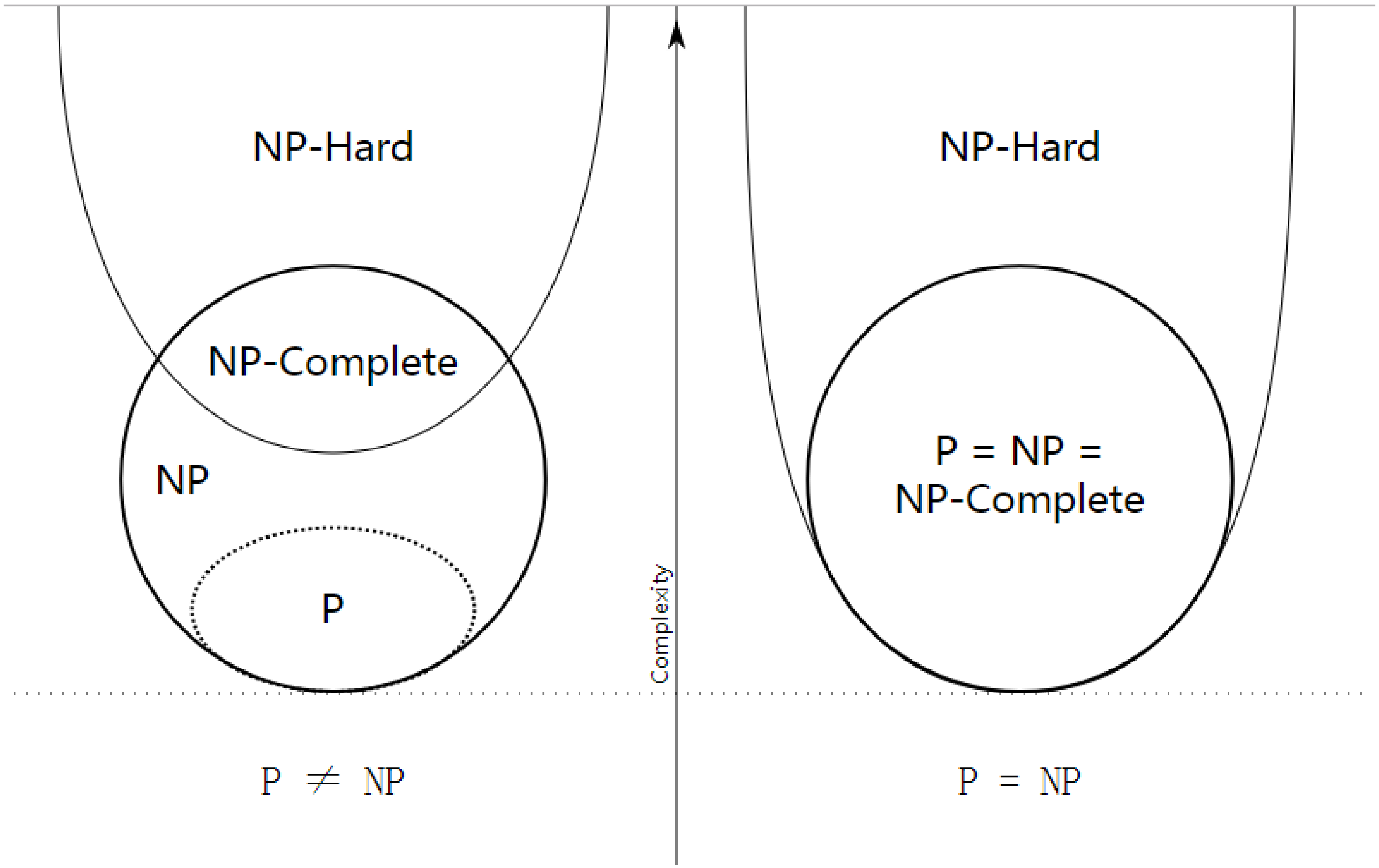


Stephen Arthur Cook (1939--)
1982年图 灵 奖

算法的复杂度

- Class NP-Complete
  - A decision problem c is NP-complete if
    - c is in NP
    - Every problem in NP is reducible to c in polynomial time.
- Class NP-Hard
  - A decision problem c is NP-hard if
    - Every problem in NP is reducible to c in polynomial time.

P=NP?


NP-Hard
NP-Complete
NP
P
Complexity
P ≠ NP
NP-Hard
P = NP =
NP-Complete
P = NP

经典问题

- 欧拉回路
  - 给定一个图，从图的某一个顶点出发，图中每条边走且仅走一次，最后回到出发点。
  - 充要条件：连通、所有节点度数为偶数
  - Fleury算法

P

- Hamilton回路
  - 给定一个图，问你能否找到一条经过每个顶点一次且恰好一次）最后又走回来的路。

NPC

3-Partition

- Can a integer set $A = \{t_1, t_2, \dots, t_{3s}\}$ be partitioned disjointly subsets A_1, A_2, … , A_s, with 3 integers in each subset

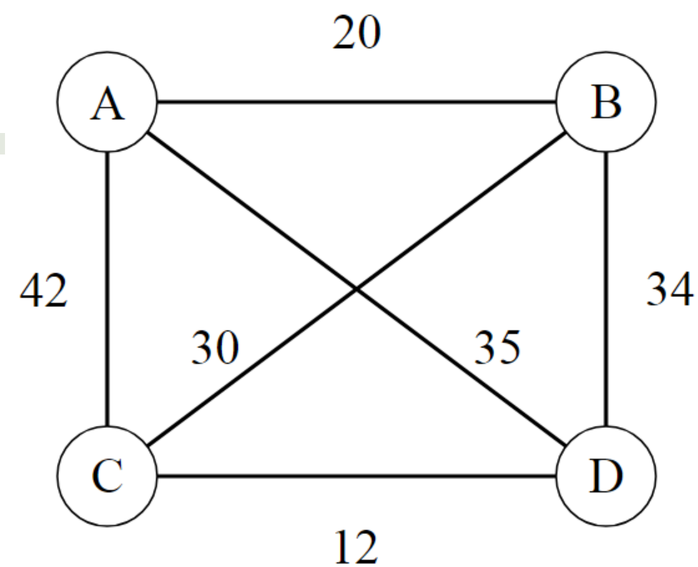
$(\forall A_j)\left(\sum_{t_i \in A_j} t_i = T\right)$?

- {**20, 23, 25, 45, 27, 40**}, $T = 90$
  - {**20, 25, 45**}
  - {**23, 27, 40**}

NPC

TSP

- TSP： The travelling salesman problem
  - Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?
  - Given the costs and a number *x*, decide whether there is a round-trip route cheaper than *x*?

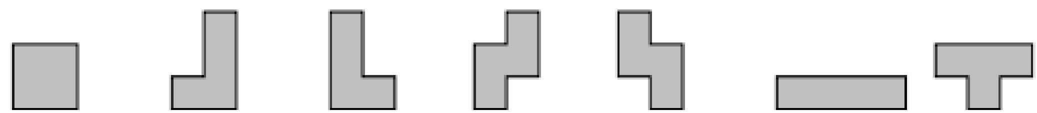


NP-hard

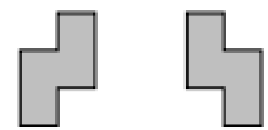
NPC

Tetris Game

- 七种方块等概率出现



- 以1的概率失败

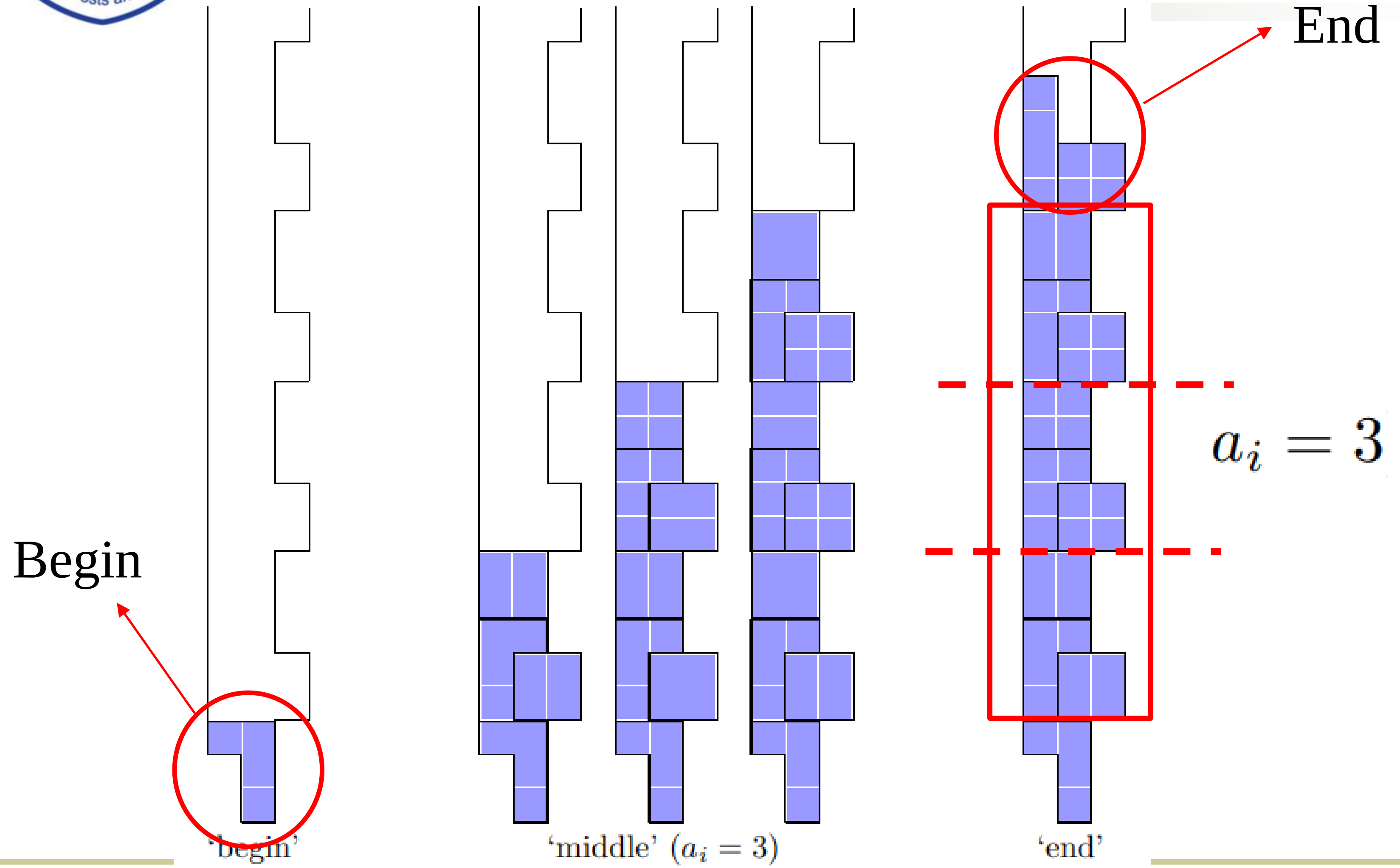


  - 轮流掉落

NP-Complete in Tetris

- 消除一行，有时候很简单，有时候很难
- 求解算法的目标：难
  - 给定初始形状
  - 给定顺序
- 找到一个带变量n的问题
- 给定一个解（下落位置和旋转的序列）
  - NP
  - Reduction?
    - From 3-partition

Reduction



Tetris Game

- NP-complete
  - Maximizing the number of rows cleared while playing the given piece sequence
- NP-hard
  - Given an initial gameboard and a sequence of p pieces, for any constant $\epsilon > 0$, it is NP-hard to approximate to within a factor of $p^{1-\epsilon}$ the maximum number of pieces that can be placed without a loss, or the maximum number of rows that can be cleared.

回顾

- 游戏的复杂度
  - 算法复杂度p, np, np-complete, np-hard
  - 游戏复杂度
- 启示?
  - 玩游戏 v.s.“不正经”？

进度

- 游戏的复杂度
- 玩游戏的人工智能
- 展望

玩游戏的人工智能

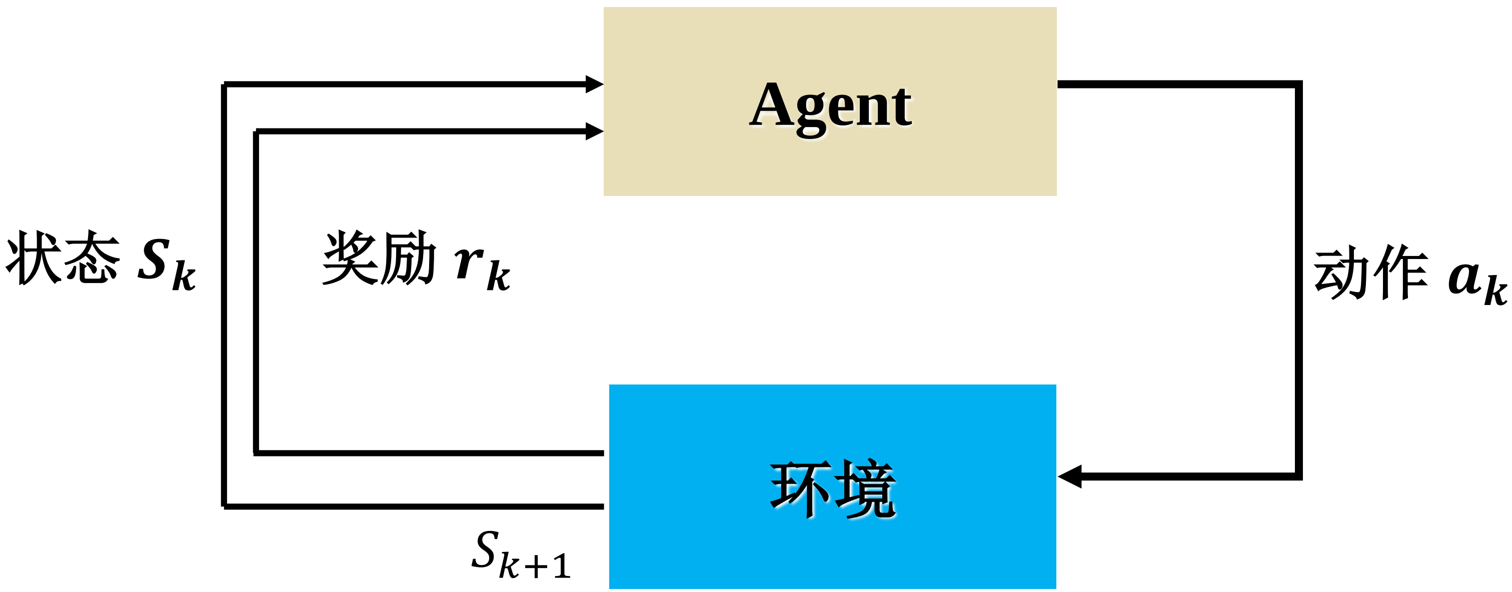
- 形式化：游戏的常见模型
- 游戏怎么玩?
  - 看得远
  - 看得准

游戏的常见模型

- 单个agent
  - Markov Decision Process
- 多个Agent
  - Markov Game

Markov Decision Process

MDP的流程



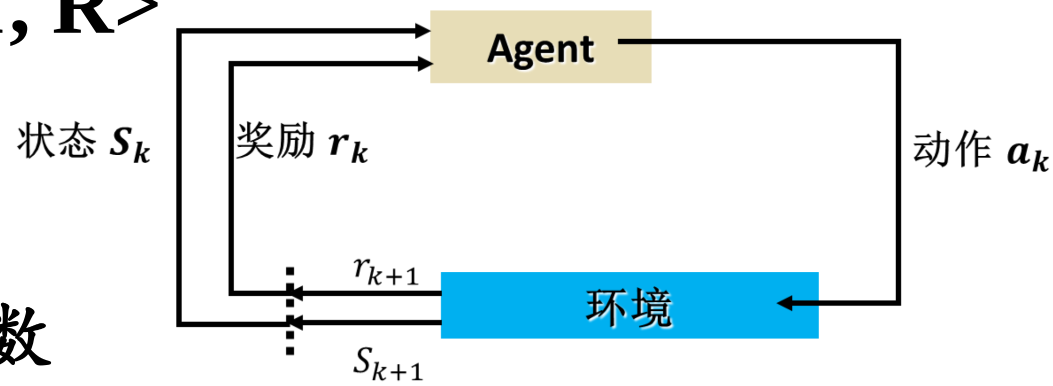
Markov Decision Process

- 四元组：<S, A, T, R>
  - S：状态空间
  - A：动作空间
  - T：状态转移函数

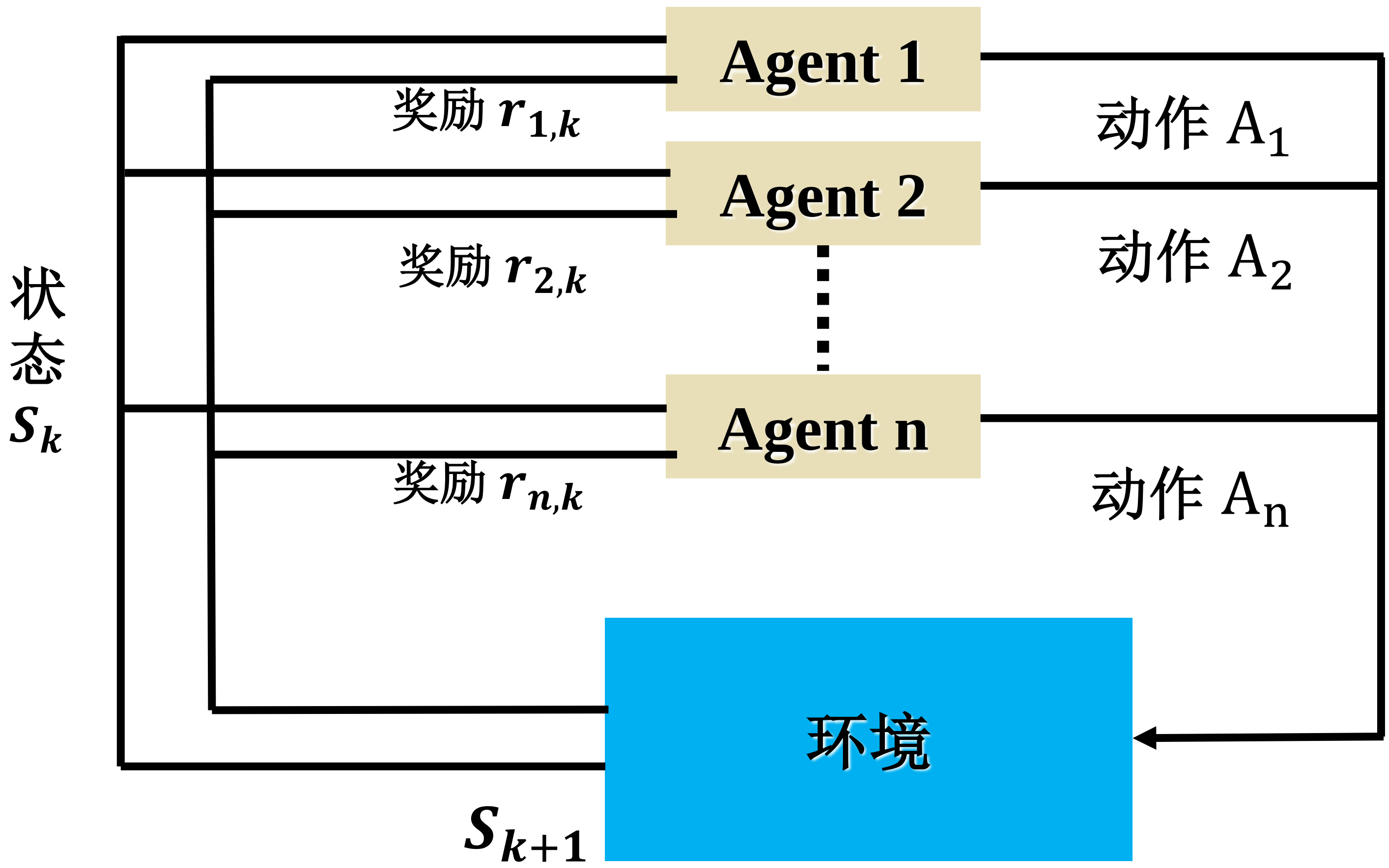
    $$T: S \times A \times S \rightarrow [0,1]$$

  - R：奖赏函数

    $$R: S \times A \times S \rightarrow \mathbb{R}$$



Markov Game


Agent 1
Agent 2
Agent n
奖励 $r_{1,k}$
奖励 $r_{2,k}$
奖励 $r_{n,k}$
动作 A_1
动作 A_2
动作 A_n
状态 s_k
s_{k+1}
环境

Markov Game

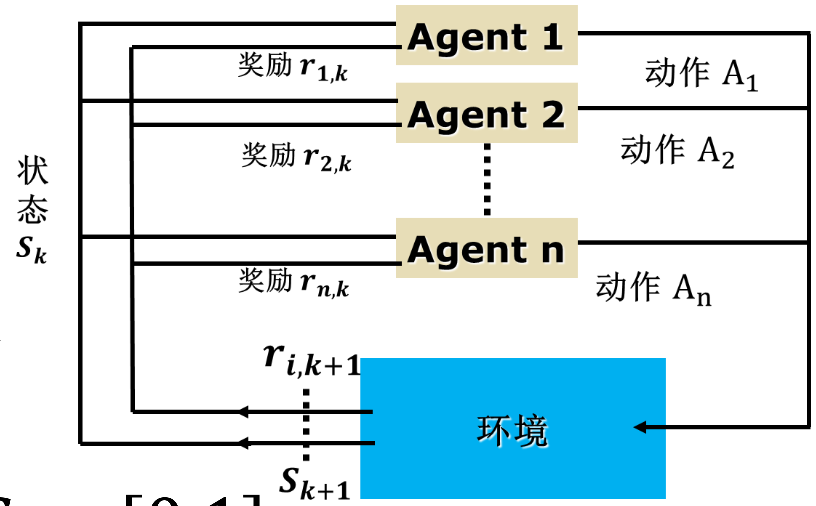
■Markov Game

- ◆<S, A, T, R>
- ◆S： 状态空间
- ◆A： 动作空间 $A_1, \cdots, A_n$
- ◆T： 状态转移函数

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

- ◆R： 奖赏函数

$$R: S \times A_1 \times \cdots \times A_n \times S \rightarrow \mathbb{R}^n$$



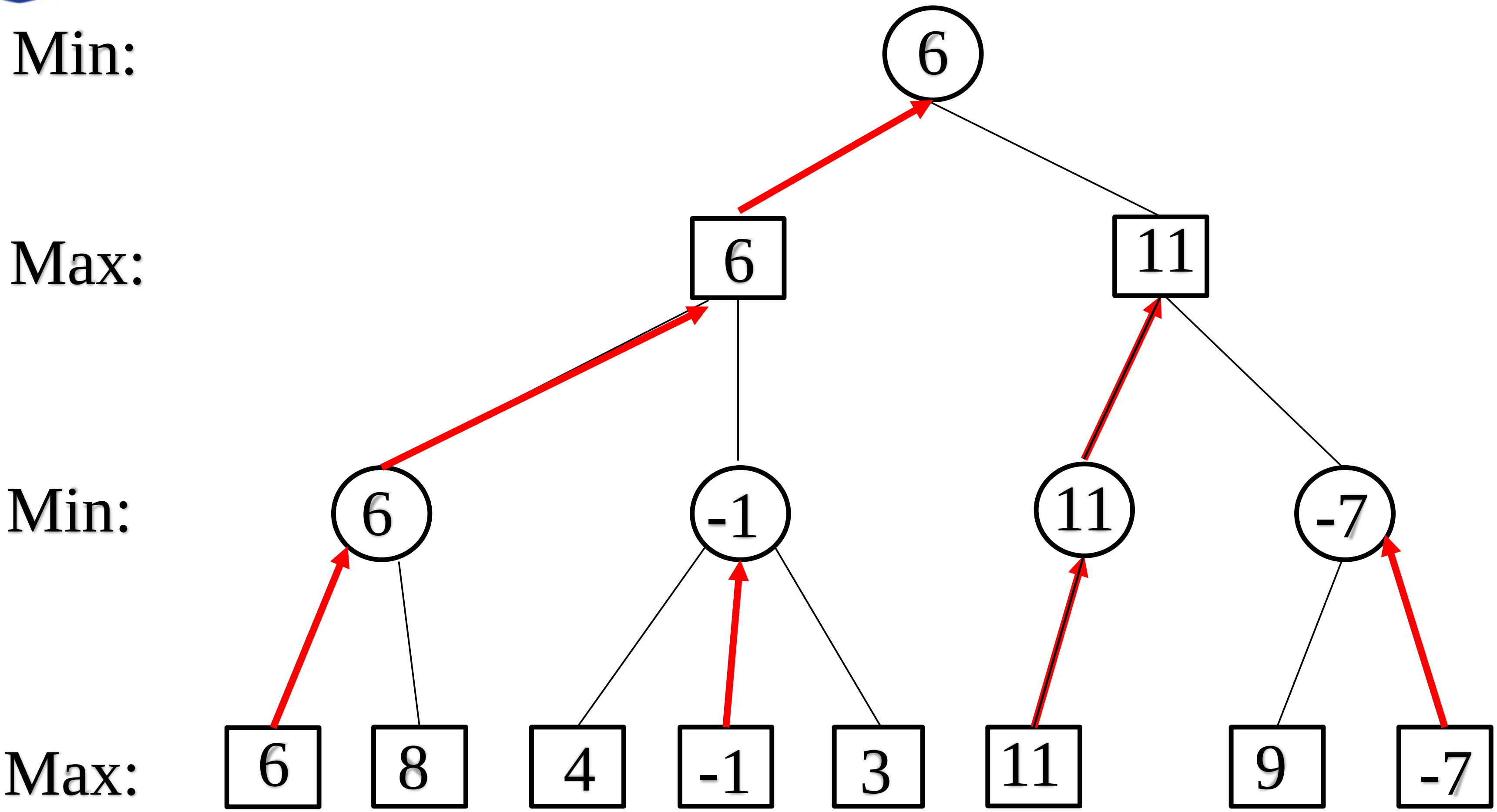
游戏怎么玩?

- 看得远
- 看得准

Search

- Mini-Max search
- $\alpha-\beta$ pruning
- Monte-Carlo method
- Monte-Carlo tree search

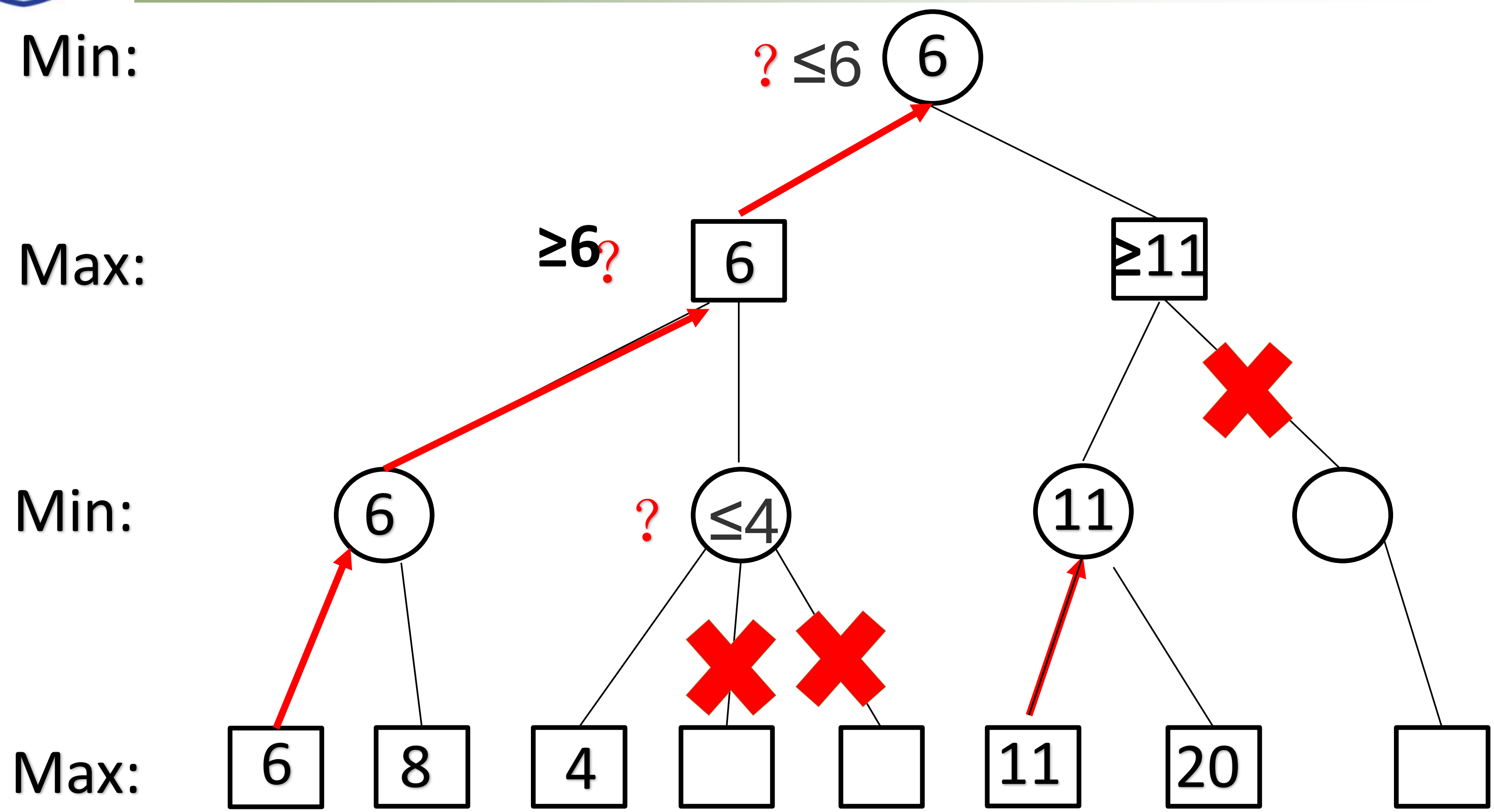
极大极小搜索



Minimax search

- A worst-case approach
  - In zero-sum games, Nash equilibrium
- Maxi-min value
  - $v = \max_{a} \min_{b} v(a, b)$

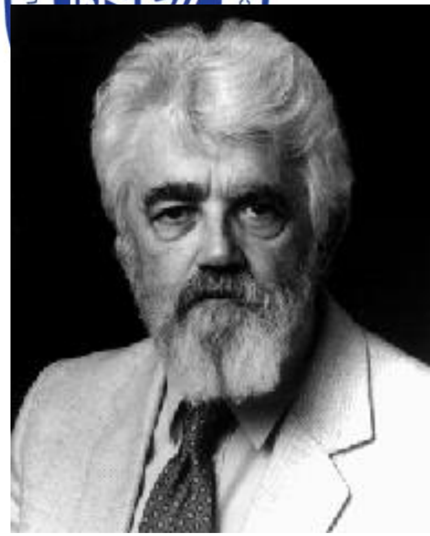
$\alpha-\beta$剪枝搜索



Donald Ervin Knuth（1938-）

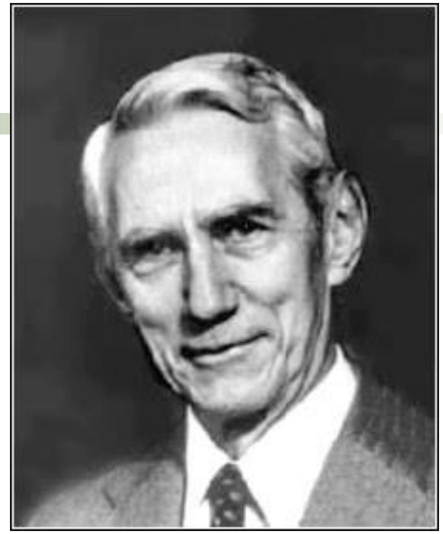
- 1974年图灵奖获得者
  - Art of computer programming
  - TeX《具体数学》
- Knuth B D E, Moore R W. An analysis of alpha beta pruning, Artificial Intelligence 6(4): 293-326, 1975.

1956 Dartmouth Conference: The Founding Fathers of AI



John McCarthy

Marvin Minsky



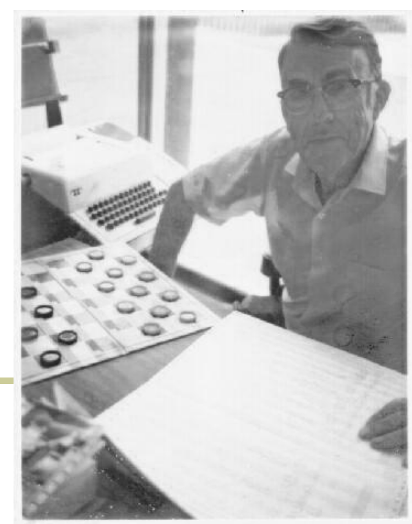
Claude Shannon

Ray Solomonoff

Alan Newell

Herbert Simon

Arthur Samuel



And three others…

Oliver Selfridge
(Pandemonium theory)

Nathaniel Rochester
(IBM, designed 701)

Trenchard More
(Natural Deduction)

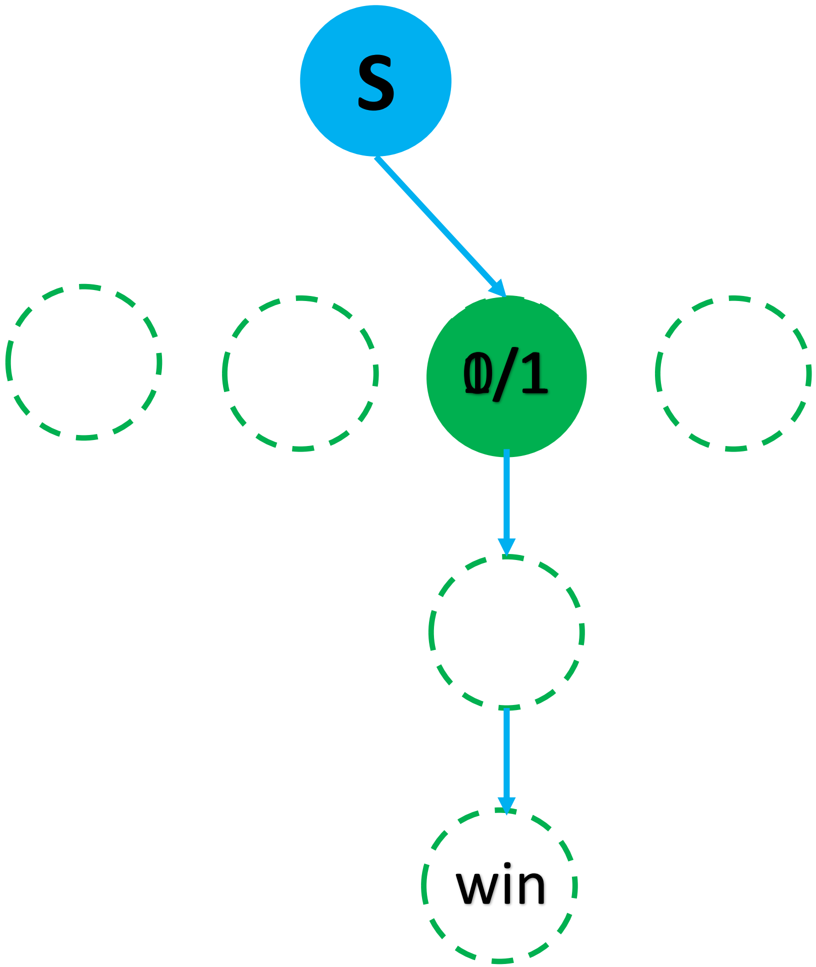
伪代码

```
int AlphaBeta(int depth, int alpha, int beta)
{
    if (depth == 0) return Evaluate();
    GenerateLegalMoves();
    while (MovesLeft())
    {
        MakeNextMove();
        val = -AlphaBeta(depth-1,-beta,-alpha);
        UnmakeMove();
        if (val >= beta)  return beta;
        if (val > alpha)  alpha = val;
    }
    return alpha;
}
```

如何看得准?

- 评估函数

Monte Carlo Search

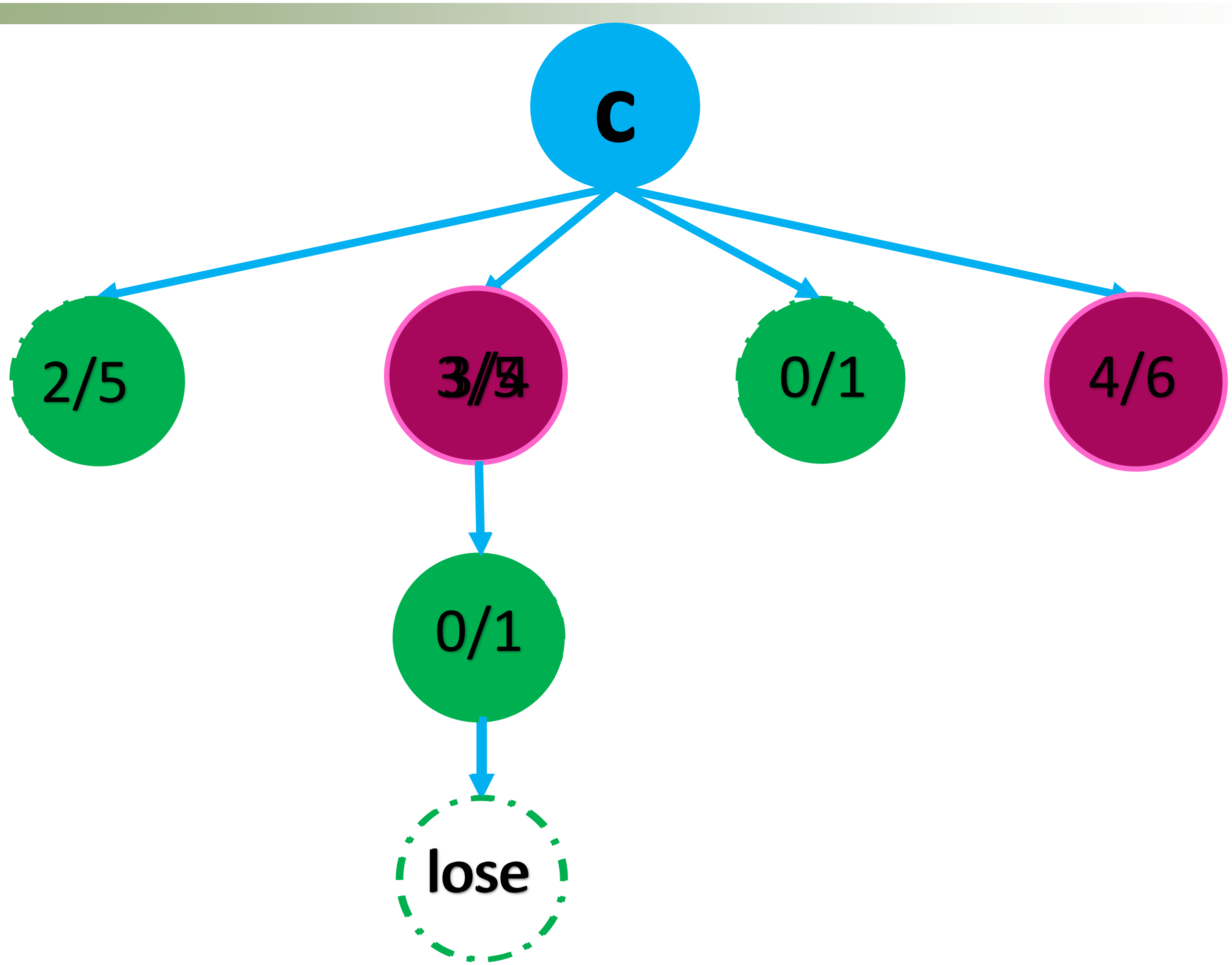


Monte Carlo Tree Search

- Selection/ expansion/ simulation/ backpropagation

- $score = v_{child} + C \cdot \sqrt{\frac{\log(N_{parent})}{N_{child}}}$

Monte Carlo Tree Search



机器学习分类（反馈）

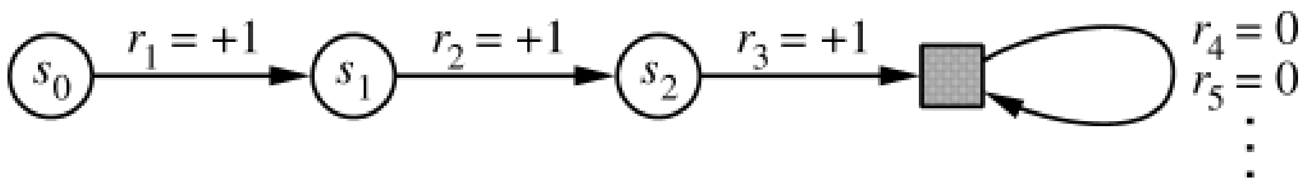
- Supervised learning （监督学习）
  - Training data: (input, output)
- Semi-supervised learning （半监督学习）
- Unsupervised learning （无监督学习）
  - No information at all about given output
- Reinforcement learning （强化学习、游戏）
  - Agent receives no examples and starts with no model of the environment and no utility function. Agent gets feedback through rewards, or **reinforcement**.

学习用的数据

- 监督学习中样例Instance $< x, y >$
- 游戏的过程
  - Markov Decision Process
    - 经验（Experience）：$< s, a, r, s' >$
    - $< s, a, r, s', a', r', s'', \dots, s^{\mathrm{T}} >$
  - Markov Game
    - $< s, a_1, a_2, \dots, a_n, r_1, r_2, \dots, r_n, s' >$
    - $< s, a_1, a_2, \dots, a_n, r_1, r_2, \dots, r_n, s',$
      $a'_1, a'_2, \dots, a'_n, r'_1, r'_2, \dots, r'_n, s'',$
      $\dots$
      $s^{\mathrm{T}} >$

Reward与Return

- 有限任务:
  - $R_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_{\mathrm{T}} = \sum_{k=0}^{\mathrm{T}} r_{t+k+1}$
  - 其中T表示terminal state的时刻
- 连续任务:
  - $R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$
  - 其中γ为折扣率，$0 \leq \gamma \leq 1$
- 统一



目标与值函数

- 累计奖赏
  - $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$
- 状态值函数
  - $\boldsymbol{V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\}}$
- 状态动作值函数
  - $\boldsymbol{Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\}}$

策略

- 策略 $\boldsymbol{\pi}: \boldsymbol{S} \times \boldsymbol{A} \rightarrow [\mathbf{0}, \mathbf{1}]$
- 两种方式
  - 利用状态动作值函数

$$\mathrm{a} = \max_{\mathrm{a} \in A} \mathrm{Q}(\mathrm{s}, \mathrm{a})$$

  - 利用游戏的after-state

$$a = \max_{a \in A} [r + \gamma V\big(as(s, a)\big)]$$

Bellman等式

- Bellman 等式

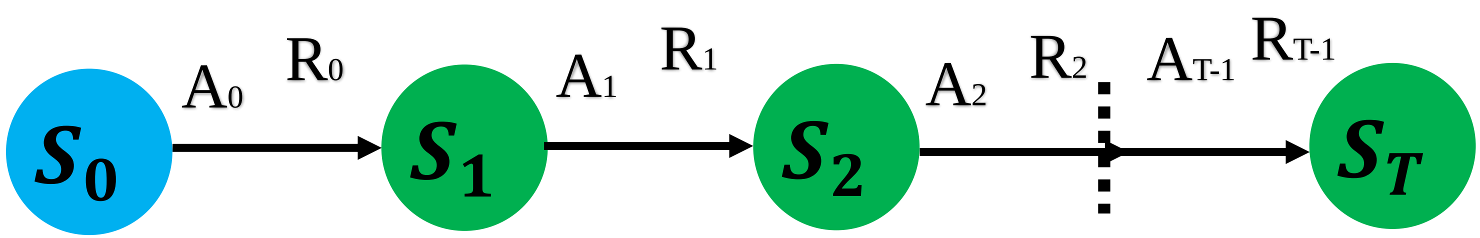
$$
\begin{aligned}
\underline{V^{\pi}(s)} &= E_{\pi}\{R_t | s_t = s\} \\
&= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \\
&= E_{\pi}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\} \\
&= E_{\pi}\{\underline{r_{t+1} + \gamma V^{\pi}(s_{t+1})} | s_t = s\}
\end{aligned}
$$

训练的数据

- 监督学习中样例Instance :$< x, \ y >$
- MDP中的数据
  - 一步：$< V(s), r + \gamma V(s') >$
    - 目标最小化$MSE = \sum_{s \in S}\left(r + \gamma V(s') - V(s)\right)^2$
  - 两步：$< V(s), r + \gamma r' + \gamma^2 V(s'') >$
  - 多步：$< V(s), r + \gamma r' + \cdots + \gamma^k V\left(s^k\right) >$

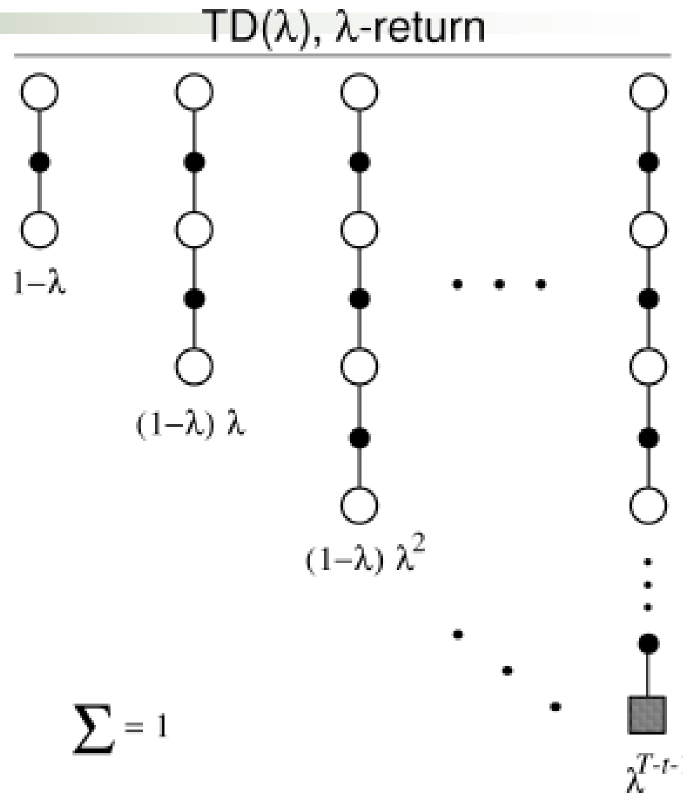
Monte Carlo

- 基于当前策略评估值函数
- $A_i = \pi(S_i)$
- $V(S_t)$=average(Returns(S_t))



- 到底用哪个?

$\lambda - return$



$$R_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}R_t^{(n)}.$$

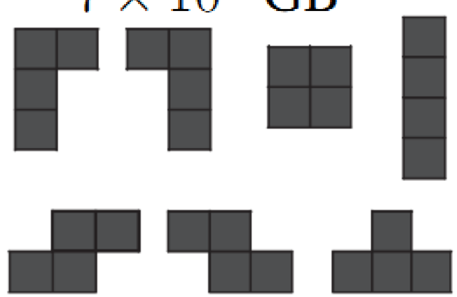
■ MDP综合数据：$< V(s), R_t^{\lambda} >$

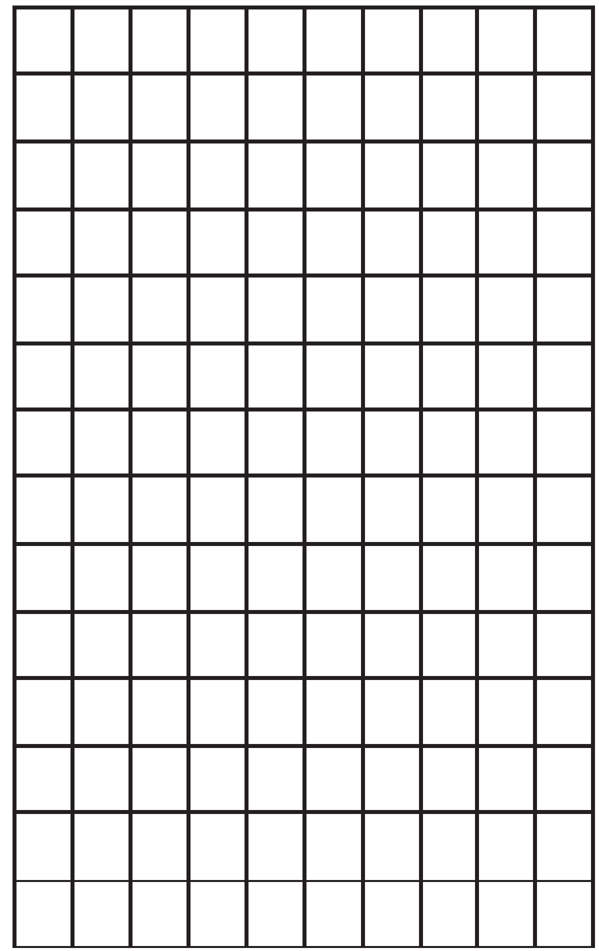
状态空间大小

- **以Tetris游戏为例**
  - 状态空间大小

    $$2^{10\times 20} \times 7 = 7 \times 2^{200}$$

- **值表**
  - 若采用值表对存储，
  - 每个值用一个字节表示，
  - 则需要空间： 7×10^{51}GB





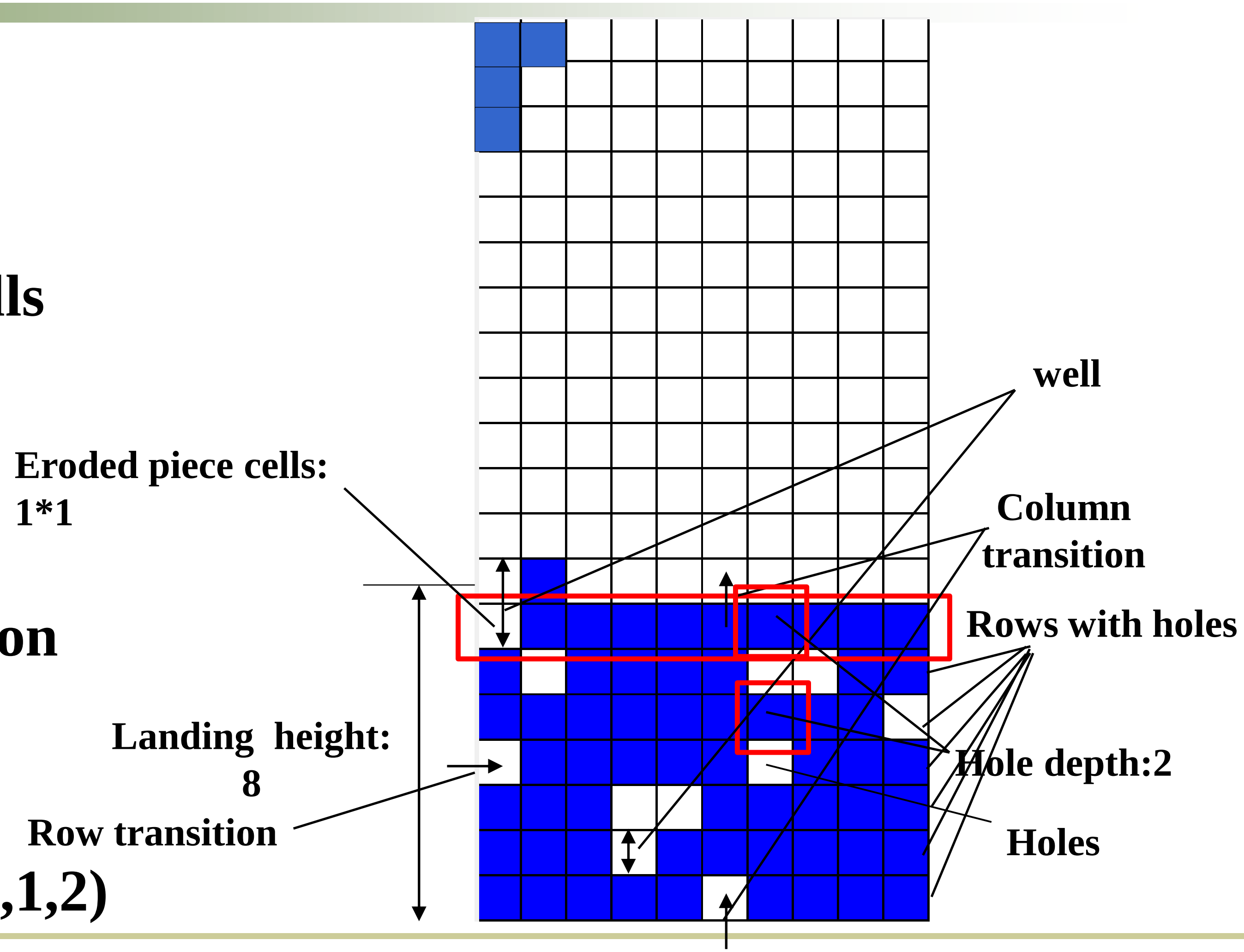
存在的问题

- 强化学习中的维度灾难体现在两个方面
  - 空间复杂度
    - 状态、动作空间的基数随着维度的增加呈指数上升趋势
  - 时间复杂度
    - 强化学习算法优化的过程与状态、动作空间的大小成正比
- 表现形式
  - 大规模或者连续状态空间
  - 连续动作空间
- 解决方法：值函数估计

特征

特征有

- Landing height
- Eroded piece cells
- Holes
- Hole depth
- Rows with holes
- Column transition
- Row transition
- Board wells
- Diversity(-2,-1,0,1,2)



线性值函数

- 线性值函数 $$V_\theta(s) = \theta\phi^\top(s) = \sum_{i=1}^{M} \theta_i \phi_i(s)$$

- 特征 $$\phi(s) = [\phi_1(s), \phi_2(s), \ldots, \phi_M(s)]$$

- 权重 $$\theta = [\theta_1, \theta_2, \ldots, \theta_M]$$

目标函数

$$\mathrm{MSE}(\theta) = ||V_\theta - v||_D^2 = (V_\theta - v)^\top D(V_\theta - v).$$

$$\mathrm{MSPE}(\theta) = ||V_\theta - \Pi v||_D^2.$$

$$\mathrm{MSBE}(\theta) = ||V_\theta - TV_\theta||_D^2.$$

目标函数

$$\mathrm{MSPBE}(\theta) = ||V_\theta - \Pi T V_\theta||_D^2.$$

$$\mathrm{NEU}(\theta) = \mathbb{E}[\delta\phi]^\top \mathbb{E}[\delta\phi].$$

优化

- 基于梯度
  - 值迭代、策略迭代
  - 策略梯度
- 不基于梯度
  - 启发式
    - Cross entropy
    - Genetic algorithm

TD(λ)

传统的线性时序差分学习（Temporal Difference Learning：TD）对于权重向量，更新公式如下：

$$\theta_{t+1} \leftarrow \theta_t + \alpha_t \delta_t \phi_t$$

资格跟踪更新：

$$\phi_{t+1} \leftarrow \gamma \lambda \phi_t + \phi(s_{t+1})$$

时序差分误差（TD error）：

$$\delta_t \leftarrow r_t + \gamma \theta_t \phi^T(s_{t+1}) - \theta_t \phi^T(s_t)$$

GTD、GTD2和TDC

- Sutton2009
- 根据MSPBE提出了GTD，GTD2和TDC。
- GTD：

$$\theta_{t+1} \leftarrow \theta_t + \alpha_t(\phi_t - \gamma\phi_{t+1})(\phi_t^T w_t)$$

$$w_{t+1} \leftarrow w_t + \alpha'_t(\delta_t\phi_t - w_t)$$

- GTD2：

$$\theta_{t+1} \leftarrow \theta_t + \alpha_t(\phi_t - \gamma\phi_{t+1})(\phi_t^T w_t)$$

$$w_{t+1} \leftarrow w_t + \alpha'_t(\delta_t - \phi_t^T w_t)\phi_t$$

- TDC：

$$\theta_{t+1} \leftarrow \theta_t + \alpha_t\delta_t\phi_t - \alpha_t\gamma\phi_{t+1}(\phi_t^T w_t)$$

$$w_{t+1} \leftarrow w_t + \alpha'_t(\delta_t\phi_t - w_t)$$

策略梯度

■ 定义目标函数

- ◆初始状态$S_0, J_0(\theta) = V^{\pi_\theta}(S_0) = \mathbb{E}[V_0]$
- ◆均值, $J_{avg}(\theta) = \sum_s d^{\pi_\theta}(S)V^{\pi_\theta}(S)$

■ 策略梯度

- ◆有限差分策略梯度 $\dfrac{\partial J(\theta)}{\partial \theta_k} \approx \dfrac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$
- ◆蒙特卡罗策略梯度
  - ➢得分函数 $\nabla_\theta \log \pi_\theta(s,a)$

$$\nabla_\theta \pi_\theta(s,a) = \pi_\theta(s,a)\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)}$$
$$= \pi_\theta(s,a)\ \nabla_\theta \log \pi_\theta(s,a)$$

启发式优化

- 特点
  - 基于经验规则
  - 黑盒
- 流程
  - 群体采样
  - 评估适应度
  - 根据经验规则更新

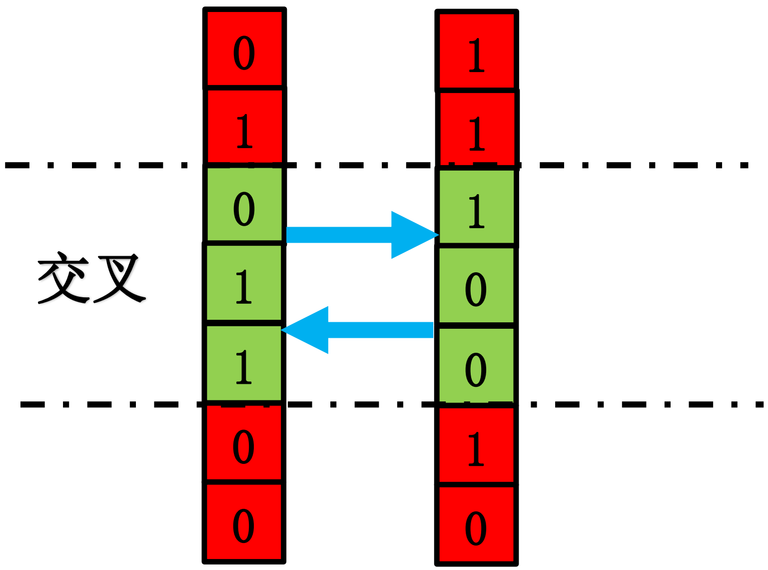
启发式优化

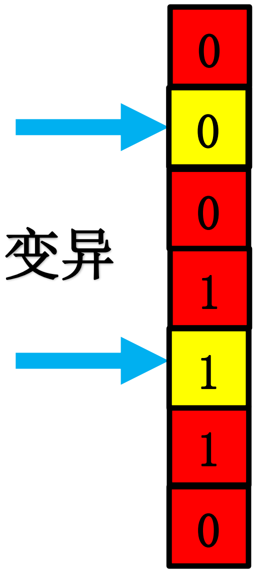
- 遗传算法
- 粒子群算法
- Noisy Cross Entropy
- Covariance Matrix Adaption

遗传算法

- 对问题的解编码（如二进制） 以及解码；
- 循环：
  - 计算当前代每个个体的适应度；
  - 根据先验生成下一代
    - 选择算子，交叉算子，变异算子等；

遗传算法





示例

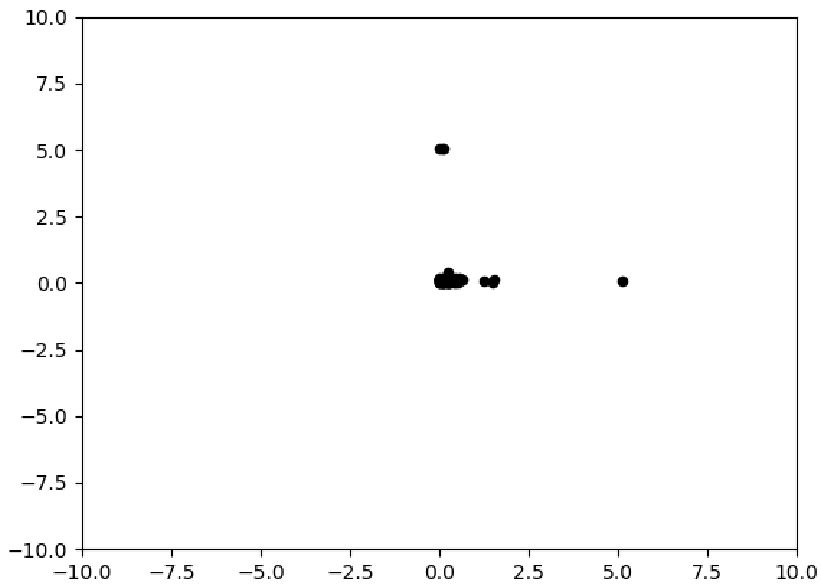
- 变量：x，y
- 函数

$$f(x, y) = x^2 + y^2$$

- 目标

$$x, y = \mathrm{argmin}_{x,y}\, f(x, y)$$

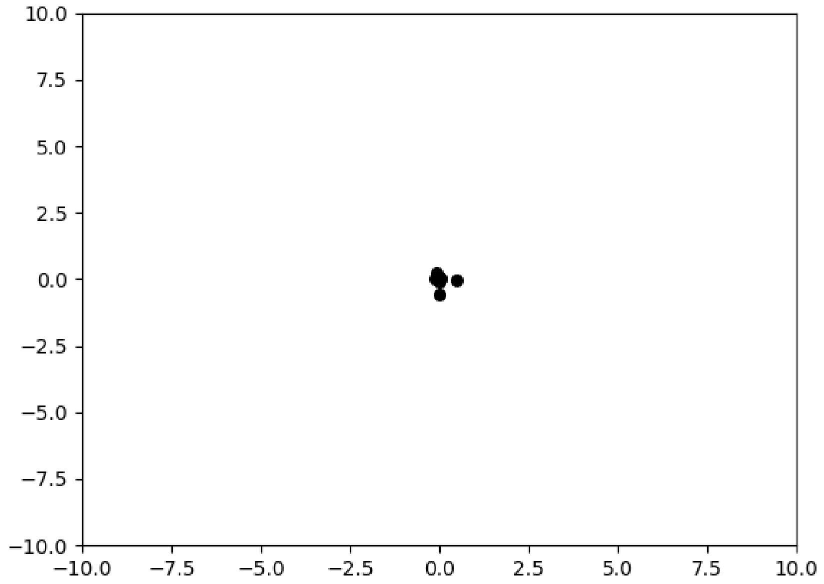
遗传算法



粒子群算法

- 初始化粒子群
- 循环：
  - 计算每个粒子的适应度
  - 根据两个极值来更新自身的速度和位置
    - 种群中的最优解 gBest
    - 历史中粒子的最优解pBest
  - $v = w * v + c_1 * rand() * (pBest - v) + c_2 * rand() * (gBest - v)$
  - $position = position + v$

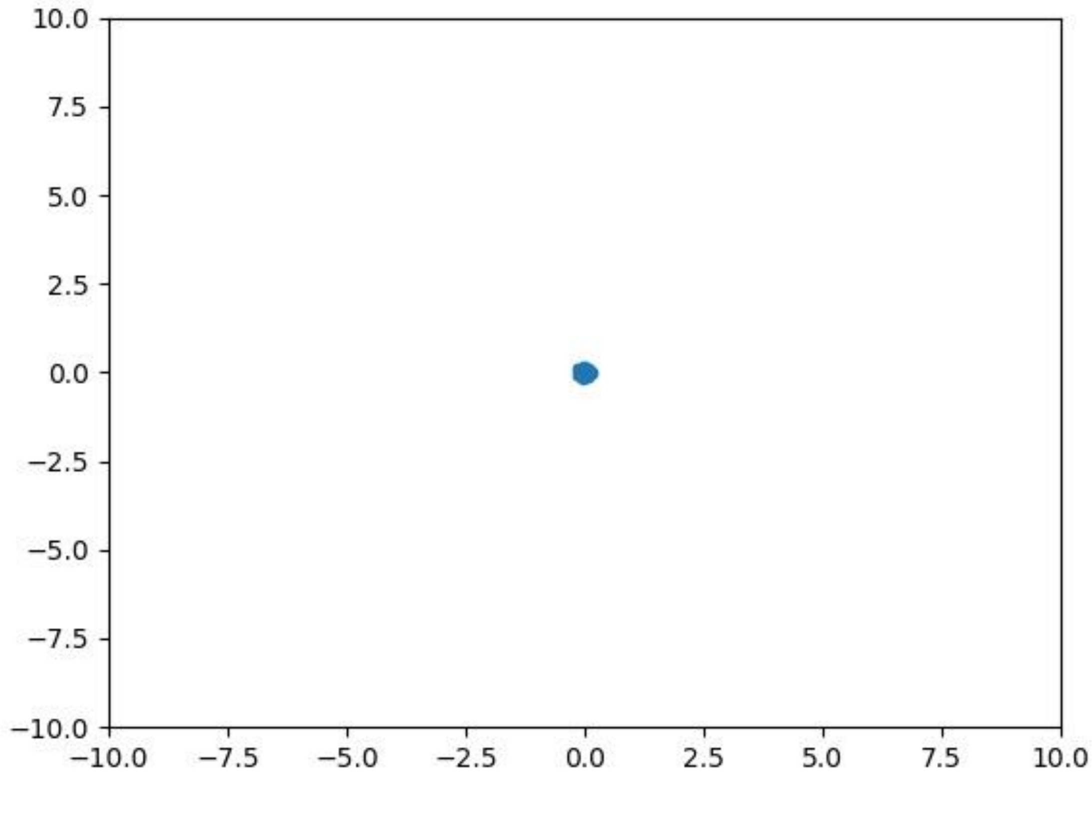
粒子群算法



Cross Entropy

- 初始化：均值、方差，设置种群数量
- 循环：
  - 根据均值方差，生成种群
  - 评估种群，计算适应度，并排序
  - 根据选择前（10%）的种群，计算均值方差

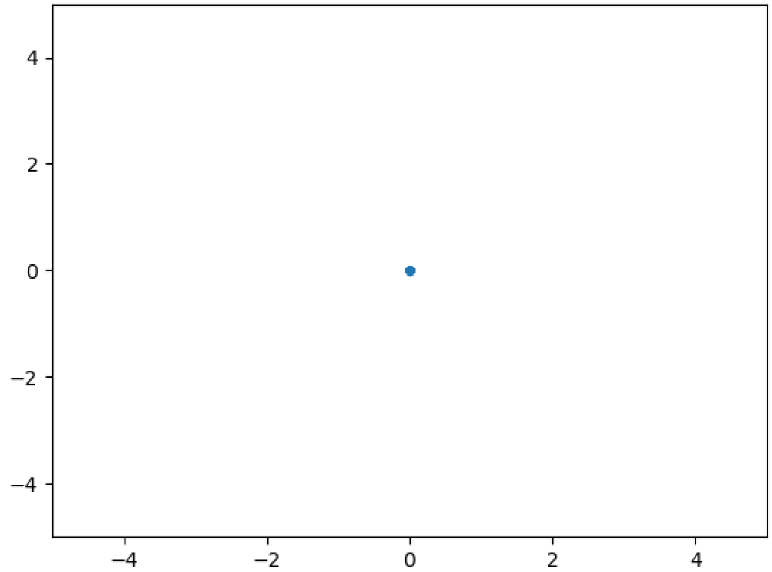
Cross Entropy



CMA-ES

- 初始化种群数量及其分布$N{\sim}(m,\ \sigma^2 C)$；
- 循环直到达到目标精度或者最大迭代次数:
  - 根据当前分布采样;
  - 计算目标函数值，评估排序;
  - 取前一半，基于递减权重更新均值;
  - 更新进化路径、协方差、步长

CMAES



优化

- 值迭代、策略迭代
  - 值函数不稳定
- 策略梯度
  - 局部最优
- 启发式
  - 从头到尾模拟

基于分类的改进策略迭代算CBMPI

- **策略评估（回归）**
  - 状态值的近似结构：$\widehat{v}_k(s^{(i)}) = \phi(s^{(i)})w$
  - 策略$\pi_k(s_t^{(i)})$下的m步马尔科夫序列：

$$\left(s^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, s_m^{(i)}\right)$$

  - 当前$\hat{v}_k(s_t^{(i)})$的无偏估计为：

$$\widehat{v}_k(s^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_{k-1}(s_m^{(i)})$$

  - 生成训练集$\{(s^{(i)}, \hat{v}_k(s_t^{(i)}))\}_{i=1}^{N}$的损失函数(最小二乘)：

$$\widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; v) = \frac{1}{N}\sum_{i=1}^{N}\left(\widehat{v}_k(s^{(i)}) - v(s^{(i)})\right)^2.$$

基于分类的改进策略迭代算CBMPI

- **策略改进（分类）：**
  - 策略值的近似结构：$\pi_u(s) = \operatorname{argmax}_a \psi(s,a)u$
  - 策略$\pi_k(s_t^{(i)})$下M局游戏的m步马尔科夫序列：

$$\left(s^{(i)}, a, r_0^{(i,j)}, s_1^{(i,j)}, a_1^{(i,j)}, \ldots, a_m^{(i,j)}, r_m^{(i,j)}, s_{m+1}^{(i,j)}\right)_{j=1}^{M}$$

  - 当前$\widehat{Q}_k(s_t^{(i)})$的无偏估计为：$\frac{1}{M}\sum_{j=1}^{M} R_k^j(s^{(i)}, a)$

$$R_k^j(s^{(i)}, a) = \sum_{t=0}^{m} \gamma^t r_t^{(i,j)} + \gamma^{m+1} v_{k-1}(s_{m+1}^{(i,j)})$$

  - 构造代价敏感的损失函数(cma-es)：

$$\widehat{\mathcal{L}}_k^{\Pi}(\widehat{\mu}; \pi) = \frac{1}{N'} \sum_{i=1}^{N'} \left[\max_{a \in \mathcal{A}} \widehat{Q}_k(s^{(i)}, a) - \widehat{Q}_k\left(s^{(i)}, \pi(s^{(i)})\right) \right]$$

基于分类的策略迭代算CBMPI

- 2013 NIPS
  - Approximate Dynamic Programming Finally Performs Well in the Game of Tetris
- 2015 JMLR
  - Approximate Modified Policy Iteration and its Application to the Game of Tetris

Boards \ Policies	**DU**	**BDU**	**DT-10**	**DT-20**
Small (10×10) board	3800	4200	5000	4300
Large (10×20) board	31,000,000	36,000,000	29,000,000	51,000,000

研究结束了吗?

- AlphaGo →AlphaGo Zero
- Tetris?
  - 从一个特别好的策略的采样开始
  - 期待Tetris Zero

回顾

- 游戏的复杂度
  - 算法复杂度p, np, np-complete, np-hard
  - 游戏复杂度
- 玩游戏的人工智能
  - 模型
  - 常见思路
    - 搜索
    - 评估：启发式、值迭代、策略迭代、策略梯度

进度

- 游戏的复杂度
- 玩游戏的人工智能
- 展望

展望

- 更高效，更优的解?
- 如何游戏开发中提高效率?
- 智能算法与传统文化?

中国象棋基本杀招

- 马后炮、双车挫、对面笑、铁门栓、卧槽马、挂角马、八角马、钓鱼马、高钓马、拔簧马、天地炮、空头炮、侧面虎、三进兵、重炮、夹车炮、闷宫杀、双马饮泉、海底捞月、白马现蹄、炮辗丹沙、送佛归殿、二鬼拍门、双车协士、大刀腕心、双照将、大胆穿心、三子归边、借炮使马、借车使炮、借车使马、车炮抽闪、车马炮兵连杀定式、各种杀法的组合

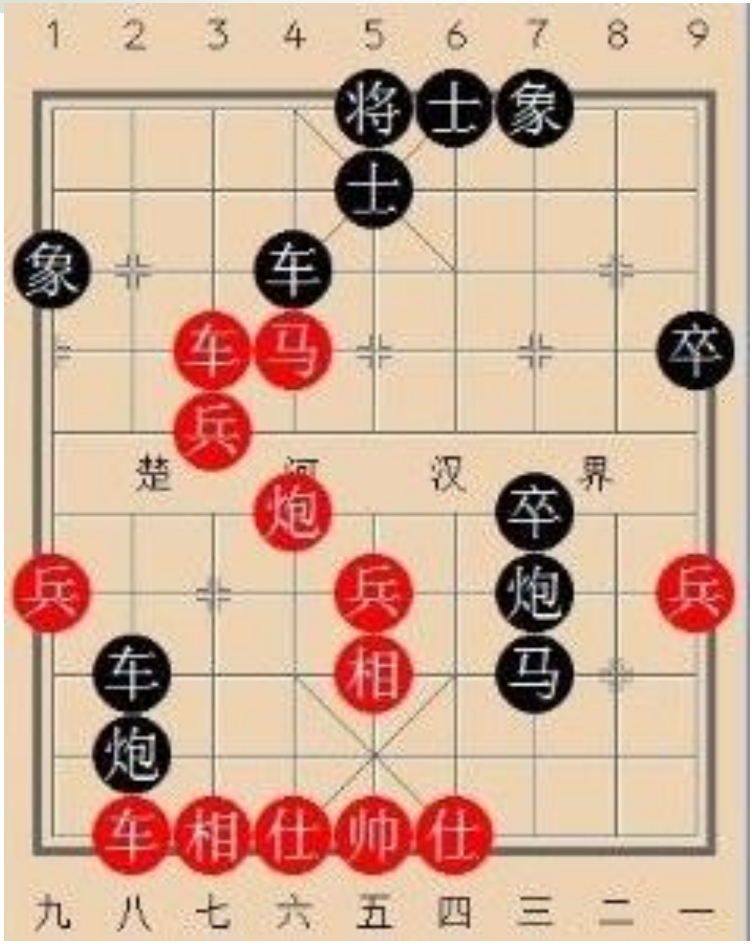
三子归边

- 1974年成都全国象棋个人赛

- 杨官璘（黑方）

  V.S.

- 徐乃基（红方）



谢谢！