人工智能实验 3 ——四子棋

UESTC · 2022 fall

四子棋

- 一、由两位棋手，一方持红色棋子，另一方持黄色棋子在棋盘上进行对弈。持红色棋子者为先手方，持黄色棋子者为后手方。

- 二、双方必须轮流把一枚己方棋子投入开口，让棋子因地心引力落在底部或其它棋子上。

- 三、当己方的四枚以上的棋子以纵、横、斜方向连成一线时则获胜，反之则失败。

- 四、棋盘满棋时，无任何同色棋子四子连一线，则平手（或者双方都同意平手，则平手）。

棋子：x 和 o

6行

1	|	|	|	|	|	|	|	|
2	|	|	|	|	|	|	|	|
3	|	|	|	|	|	0 |	|	|
4	|	|	|	x |	0 |	x |	|	|
5	|	|	|	0 |	x |	x |	|	|
6	|	|	0 |	x |	x |	o |	|	o |
		-	-	-	-	-	-	-	
		1	2	3	4	5	6	7	

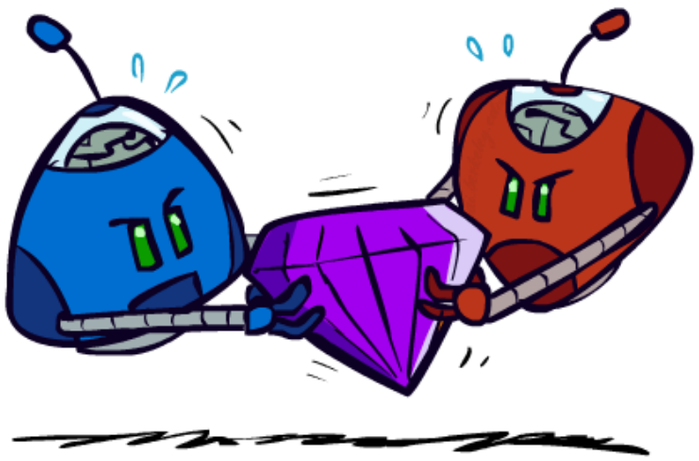
7列: 0 - 6

如何下棋?

- 尽量形成4连
- 不行的话，那就尽量多地3连（从而将来有更多的可能性形成4连）
- 或者，尽量多地2连（从而将来有更多的可能性形成4连）
- 思考：有没有更好的方式?

```
# 一个简单的效用函数，通过计算四子连线、三子连线
# 和二子连线的数量计算效用值
def utility_value(board, player):
    if player == HUMAN_PLAYER:
        opponent = AI_PLAYER
    else:
        opponent = HUMAN_PLAYER

    player_fours = count_sequence(board, player, 4)
    player_threes = count_sequence(board, player, 3)
    player_twos = count_sequence(board, player, 2)
player_score = player_fours * 99999 + player_threes \
           * 999 + player_twos * 99

opponent_fours = count_sequence(board, opponent, 4)
if opponent_fours > 0:
    # 必须避免!
    return float('-inf')

opponent_threes = count_sequence(board, opponent, 3)
opponent_twos = count_sequence(board, opponent, 2)
opponent_score = opponent_threes * 999 \
             + opponent_twos * 99

return player_score - opponent_score
```

但是，只考虑自己是不行的！！



下棋是一个零和游戏

- 智能体竞争实现相反的利益
- 一方最大化这个利益,另一方最小化它

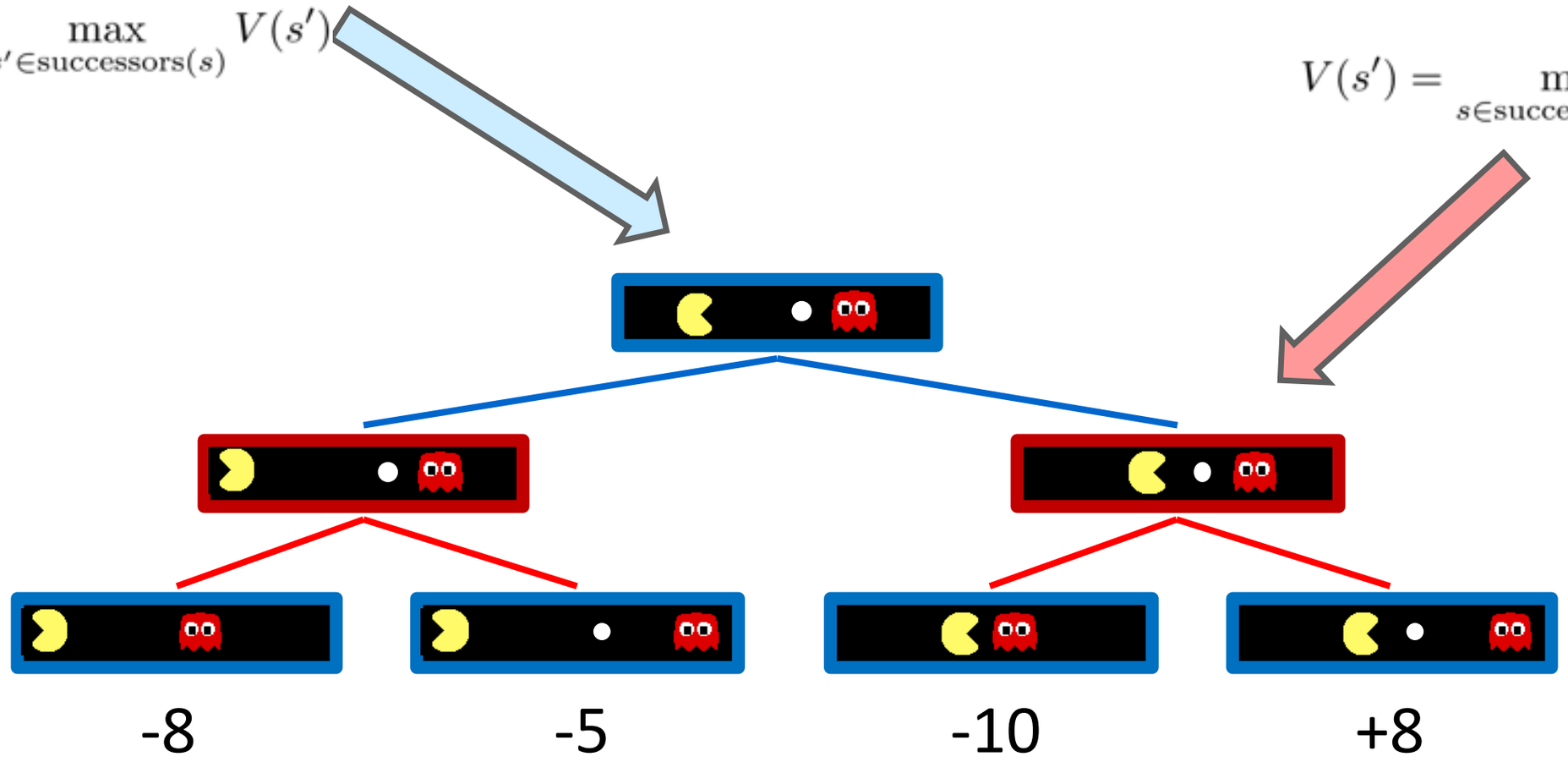
极大极小值(Minimax values)

MAX节点:自己控制下的节点状态

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN节点:对手控制下的节点状态

$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$

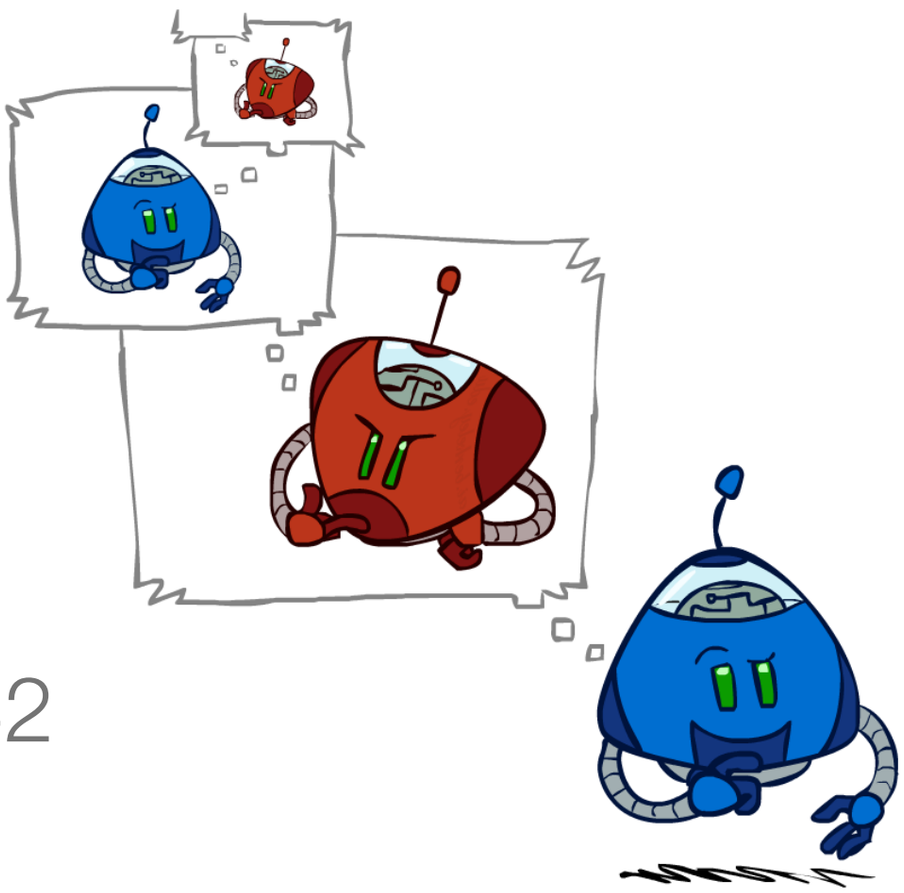


终局状态的值是已知的

$$V(s) = \text{known}$$

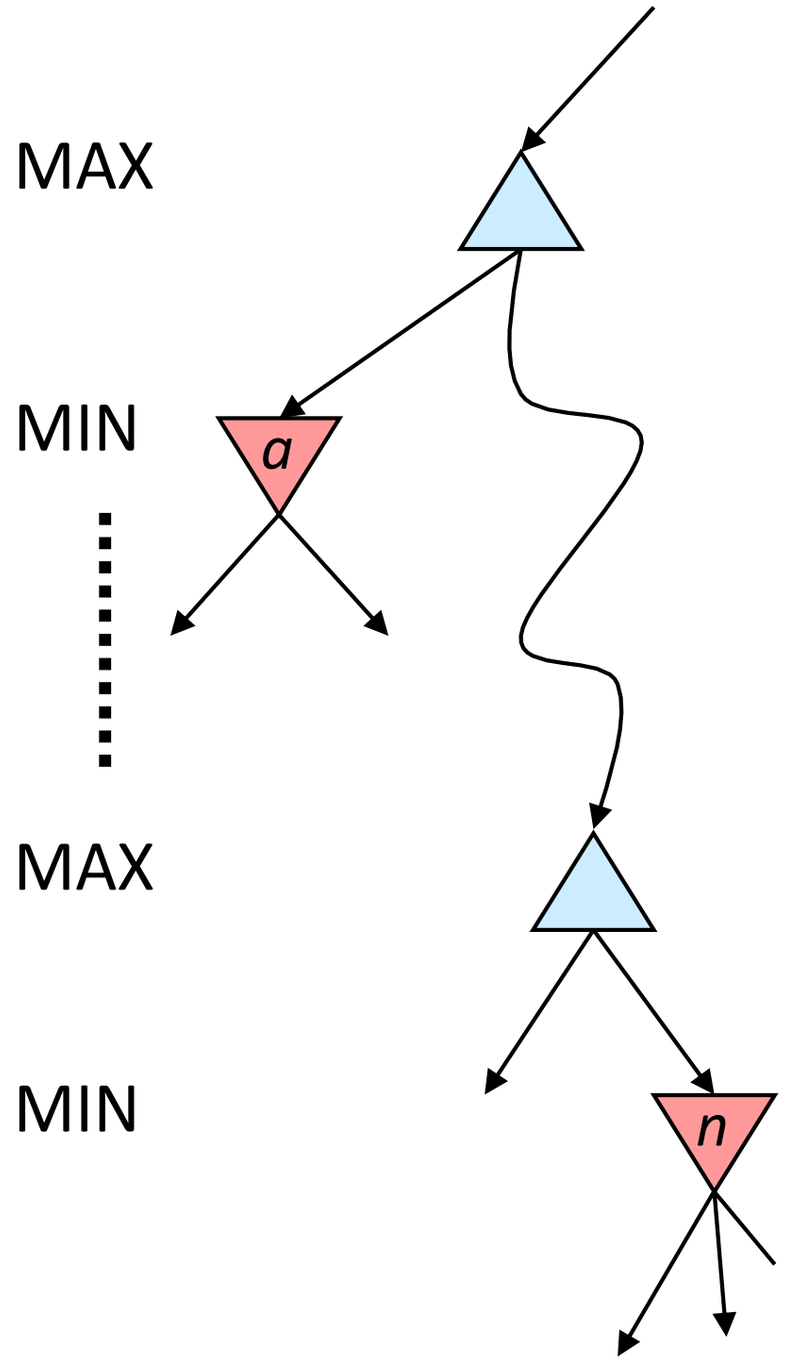
极大极小搜索的效率

- Minimax的效率?
  - 深度优先穷尽搜索
  - 时间复杂度:$O(b^m)$
  - 空间复杂度:O(bm)
- 举例：Connect4，b≈7, m≈42
  - 找到准确解是完全不可行的
  - 但是，有必要探索整棵树吗?人是如何下象棋的?



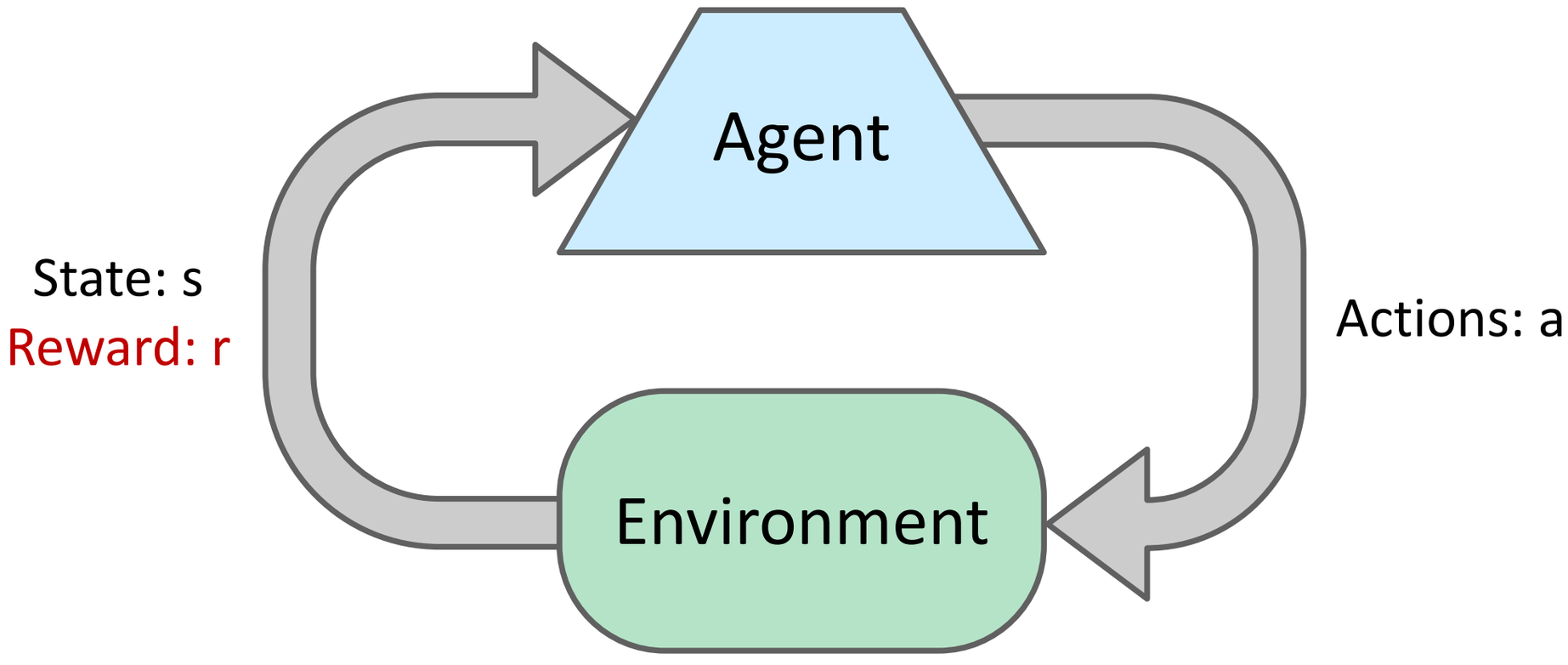
Alpha-Beta 剪枝算法

- 假定修剪MIN节点的子节点
  - 假设正在计算节点n的最小值
  - n节点的值在检查其子节点的过程中逐渐减小
  - 令α是从根节点到当前MIN节点路径上的(任何一个)MAX节点所能取到的最大值
  - 如果n的当前值比α的小，那么路径上的MAX分支节点将会避开这条路径，所以我们可以剪掉(不去检查)n的其他子节点
- 对MAX节点的子节点剪枝操作是对称的
  - 令β是从根到当前MAX节点路径中的MIN节点MIN所能达到的最小值



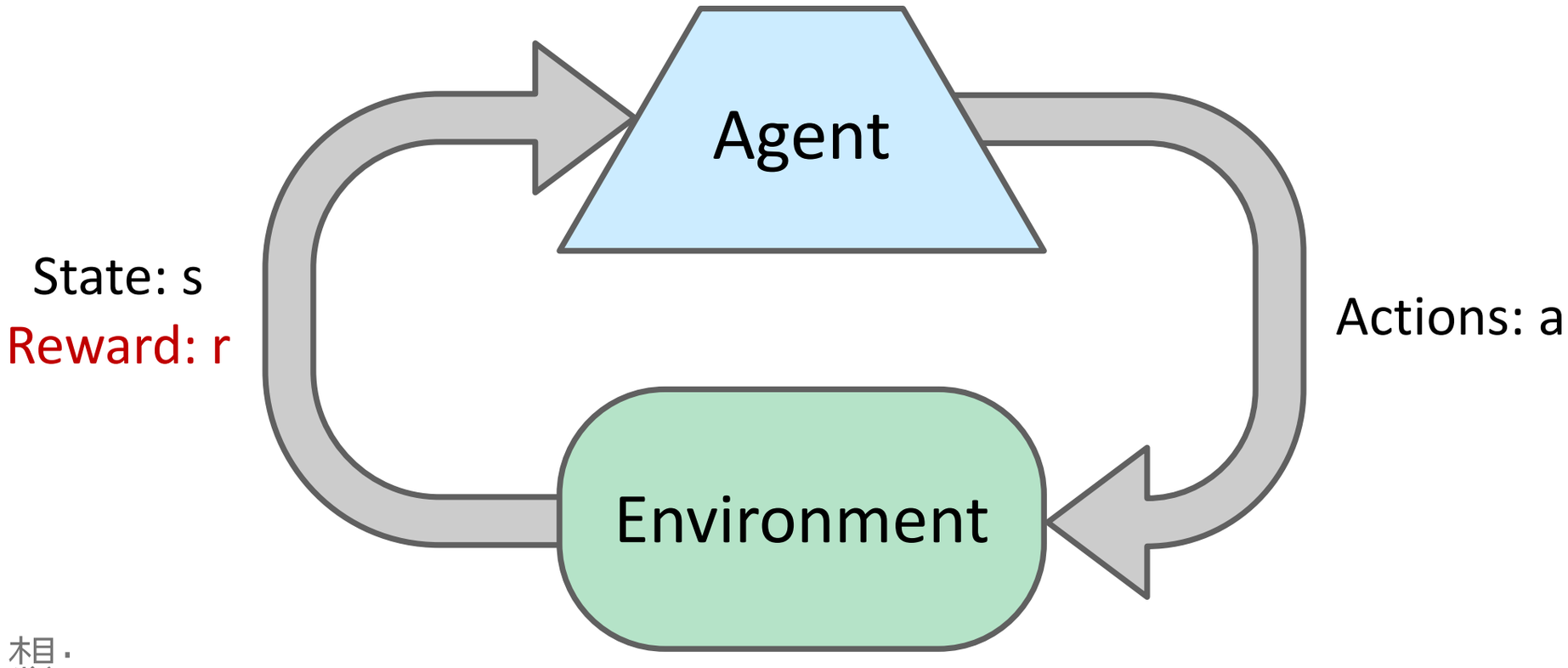
有没有更好的启发式函数?

- 也许不需要编写（固定的）启发式函数



- 能赢就行：让算法自动地从失败/成功中对棋局进行打分

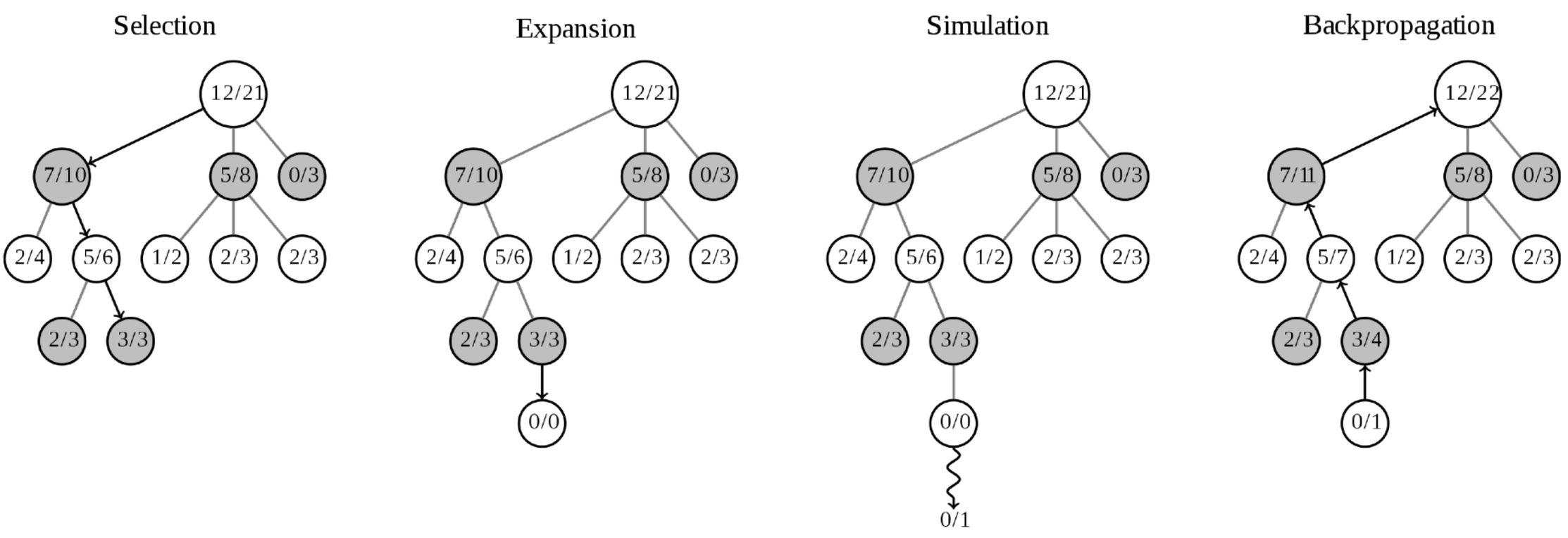
强化学习



- 基本思想:
  - 通过 奖赏值 的形式来获得反馈
  - 智能体的功效值是由 奖赏函数(reward function)来定义的
  - 必须学习如何行动，以获得最大化的期望奖赏值
  - 所有的学习是基于观察到的样本结果!

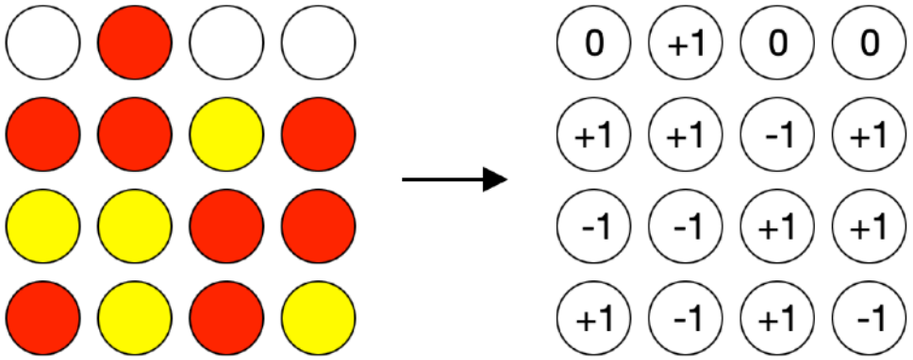
反馈太慢怎么办?

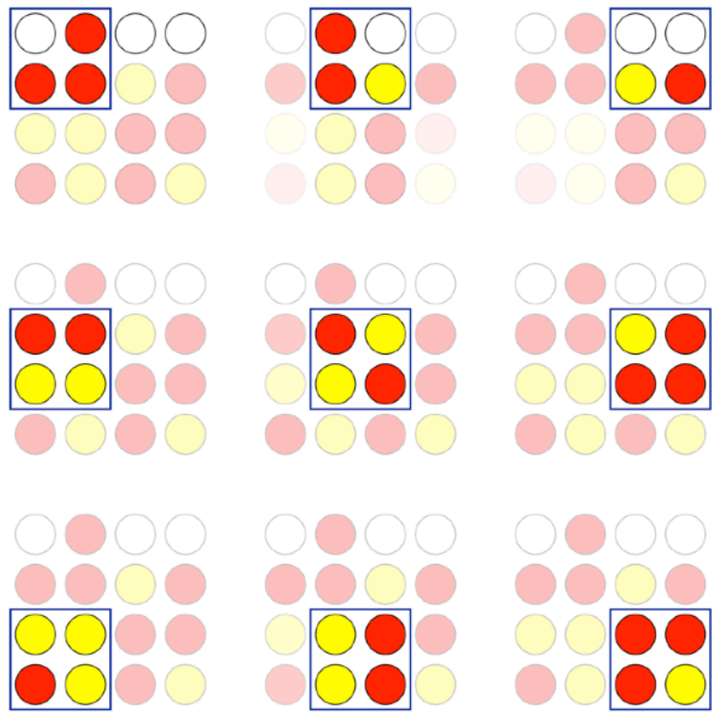
- 蒙特卡洛树搜索

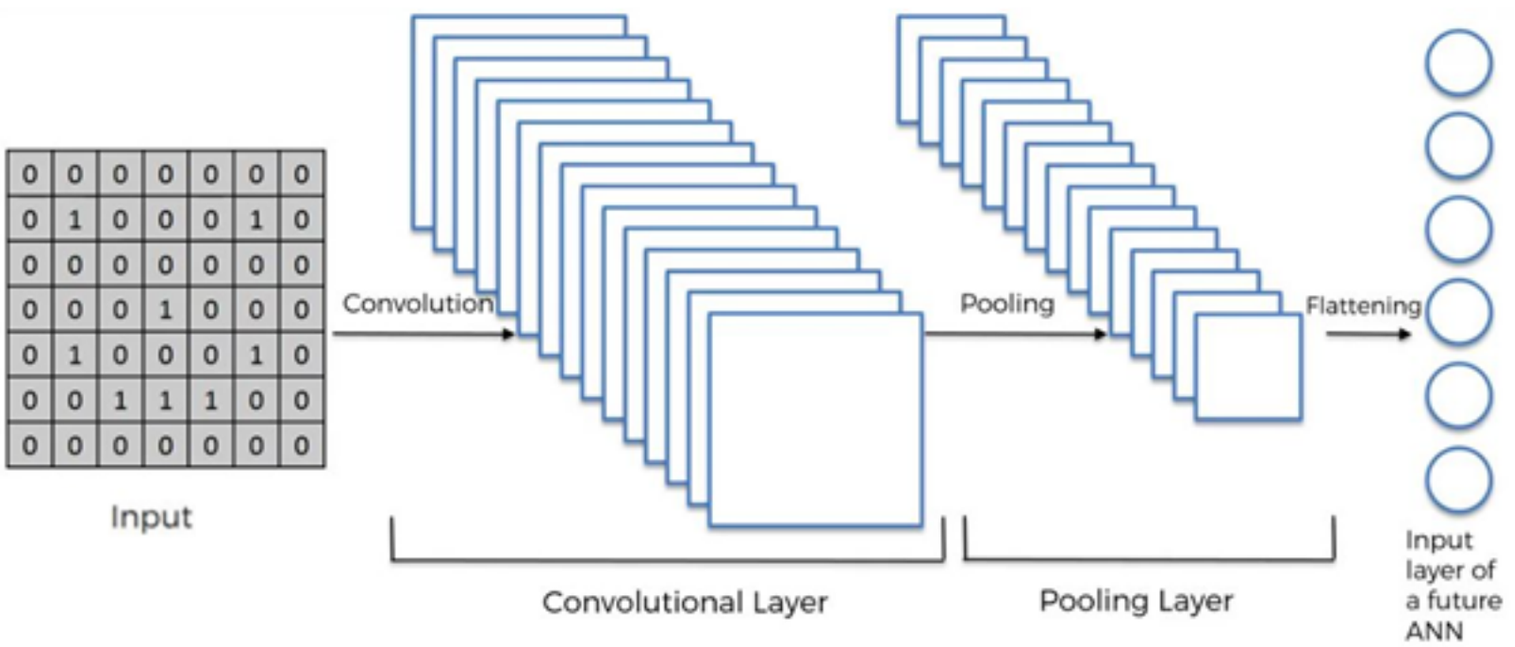


增强评价函数的泛化能力

- 卷积神经网络







实验要求

1. 学习并掌握Connect 4

2. 实现基于MiniMax对抗搜索的下棋算法

3. 实现基于AlphaBeta的剪枝算法

* 思考并探索用更好的效用评估函数改进下棋算法

* 思考并探索利用强化学习改进下棋算法