人工智能实验 3 ——四子棋

UESTC · 2021 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

	o	o	x		x	
	x	x	x	o	o	
	x	x	o	x	o	o
x	x	x	o	o	x	o
0	1	2	3	4	5	6

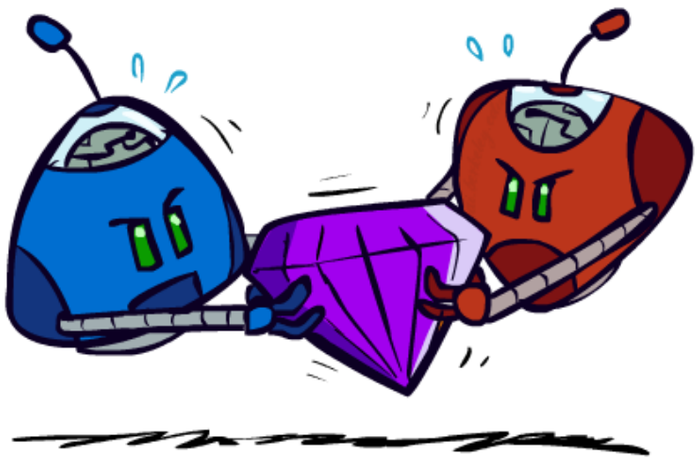
7列: 0 - 6

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?

```
def value(self, state, color):
    if color == self.colors[0]:
        o_color = self.colors[1]
    else:
        o_color = self.colors[0]

    # 统计本方的4连、3连、2连的数量
    my_fours = self.checkForStreak(state, color, 4)
    my_threes = self.checkForStreak(state, color, 3)
    my_twos = self.checkForStreak(state, color, 2)
    # 统计对方的4连的数量
    opp_fours = self.checkForStreak(state, o_color, 4)
    # 如果对方有4连，肯定要不得！！！
    if opp_fours > 0:
        return -100000
    else:
        # 一个简单的h_value计算式
        h_value = my_fours * 100000 + my_threes * 100 + my_twos
        return h_value
```

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

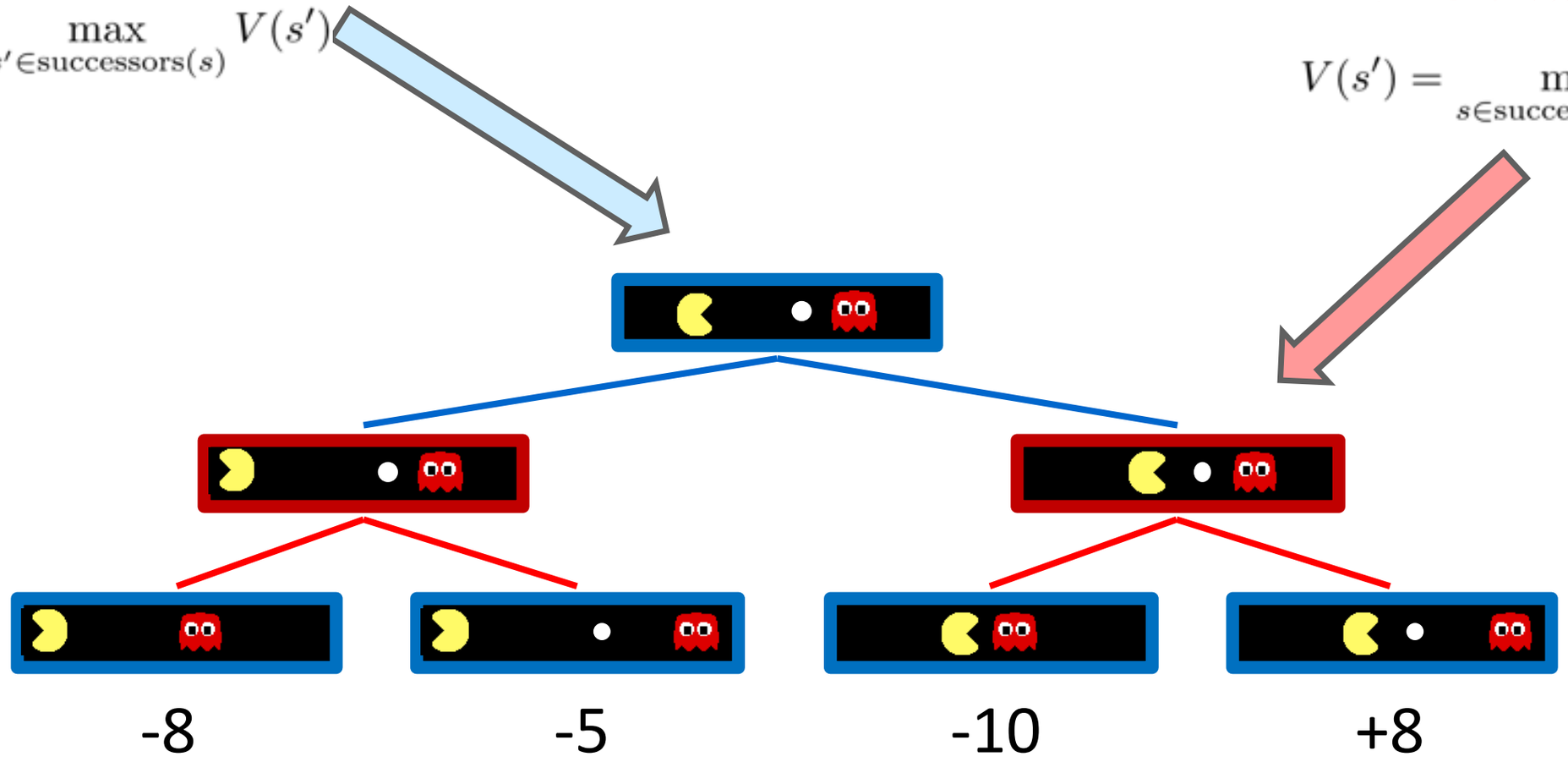
极大极小值(Minimax values)

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

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



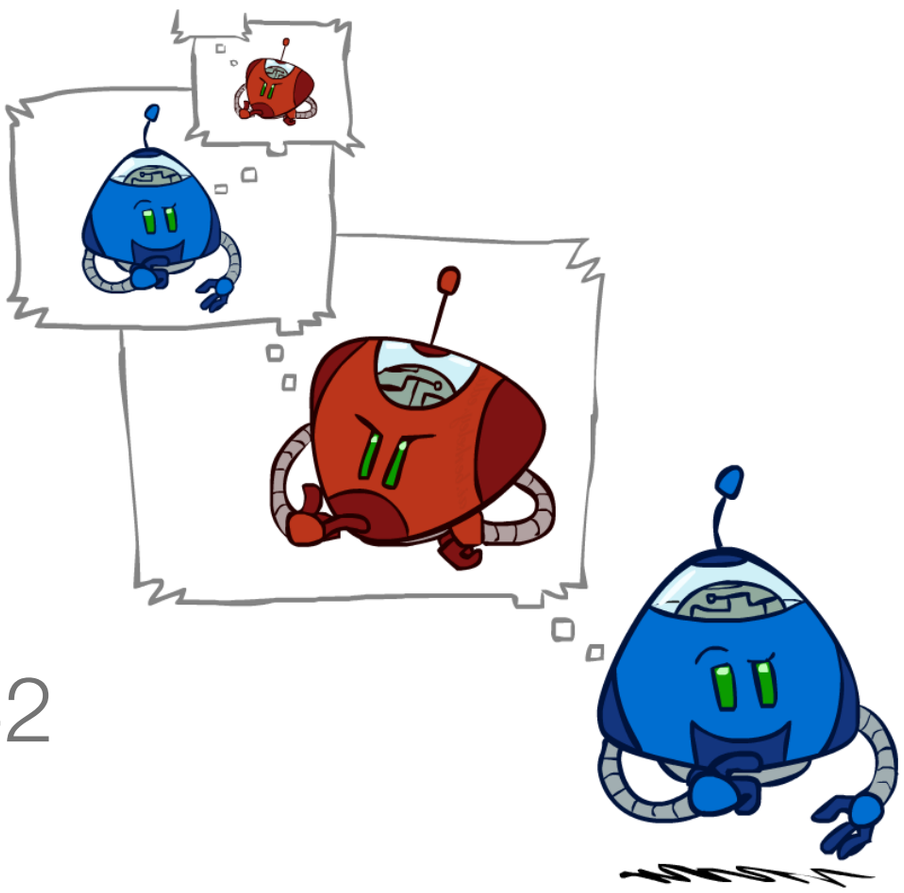
终局状态的值是已知的

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

```
# 搜索深度为depth的子树，返回得分（alpha value）
def search(self, depth, state, curr_player):

    # 遍历所有位置，记录合法的下法
    legal_moves = []
    for i in range(7):
        # 如果下在这里是是合法的
        if self.isLegalMove(i, state):
            # 试探性地下棋
            temp = self.makeMove(state, i, curr_player)
            # 放到legal_moves集合里面去。
            legal_moves.append(temp)

    # 如果此节点（状态）是终端节点或深度==0。。。
    if depth == 0 or len(legal_moves) == 0 or self.gameIsOver(state):
        # 返回节点的启发函数值
        return self.value(state, curr_player)

    # 确定对手的颜色
    if curr_player == self.colors[0]:
        opp_player = self.colors[1]
    else:
        opp_player = self.colors[0]

    alpha = -99999999
    # Max算法
    # 思考：为什么代码中只有max，而没有mini?
    for child in legal_moves:
        if child == None:
            print("child == None (search)")
        alpha = max(alpha, -self.search(depth - 1, child, opp_player))
    return alpha
```

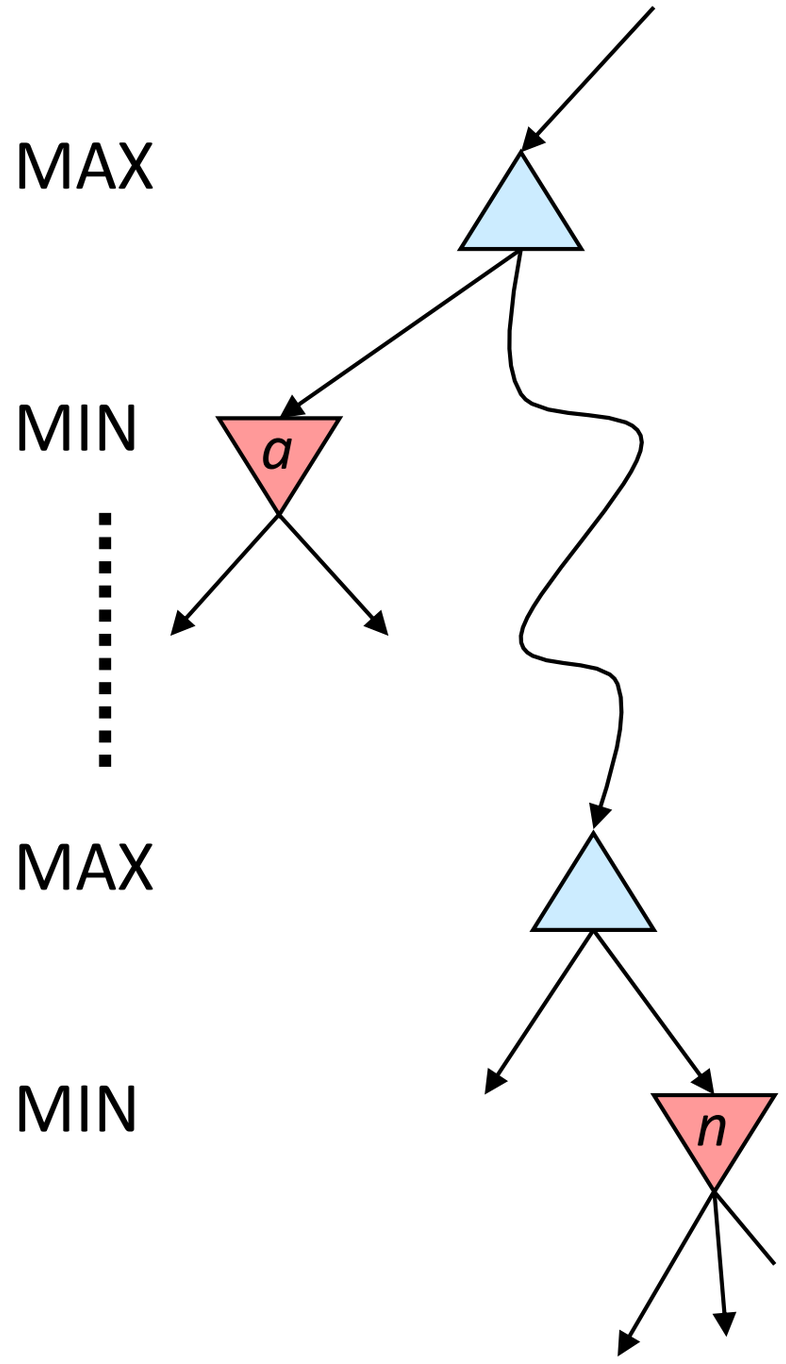
极大极小搜索的效率

- 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?



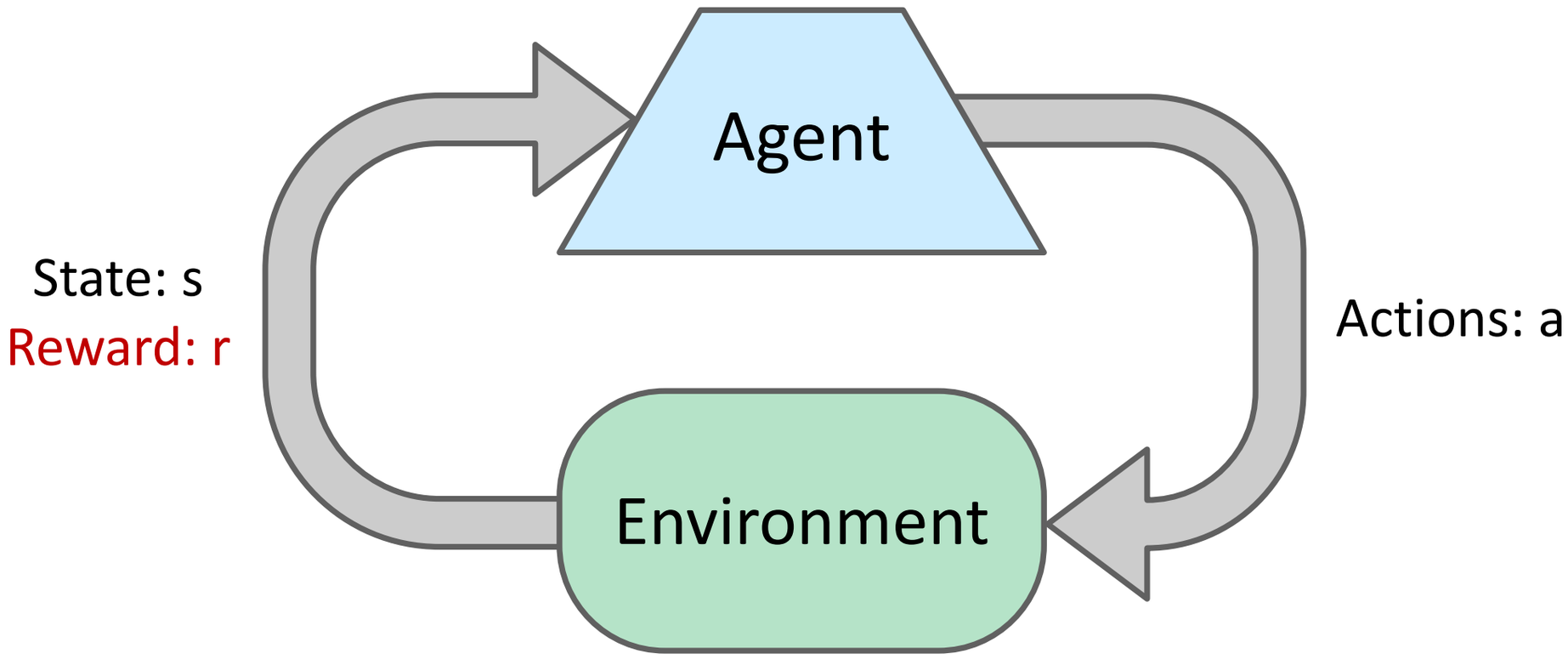
Alpha-Beta 剪枝算法

- 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



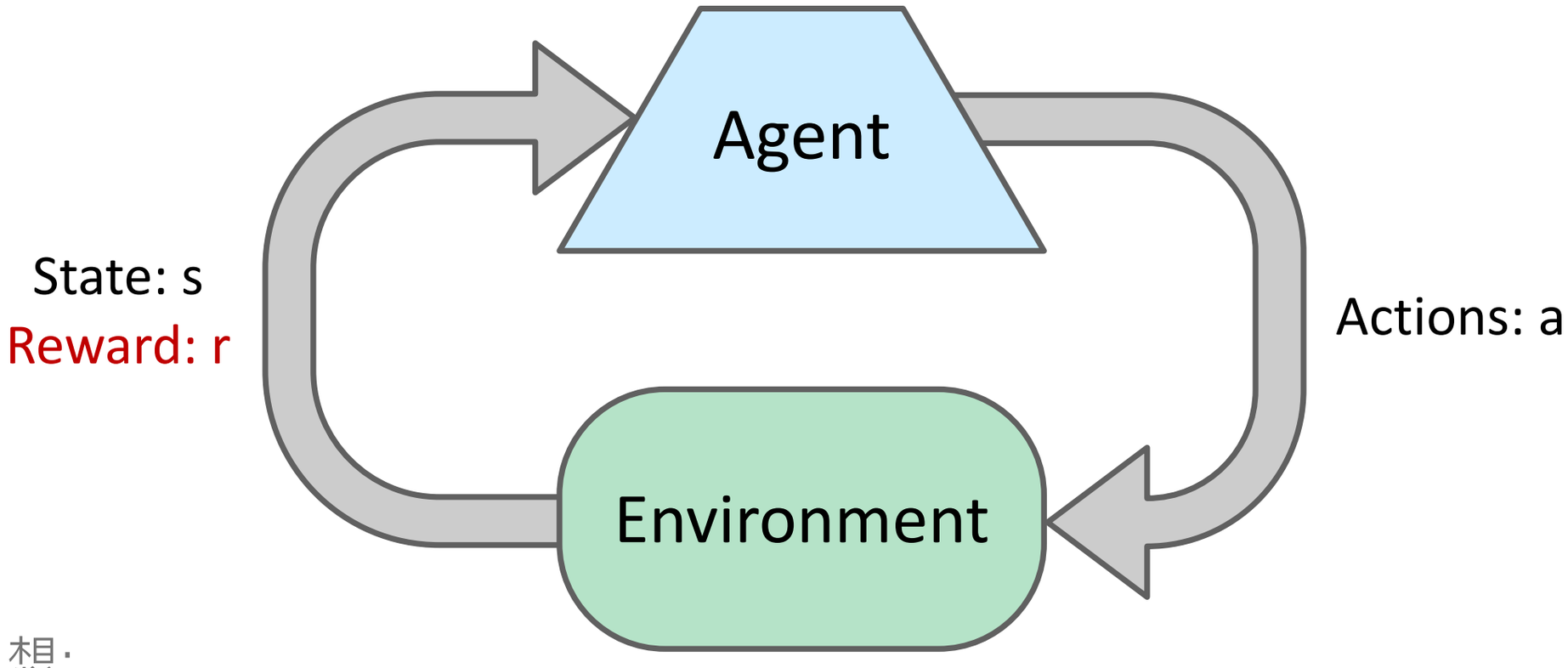
有没有更好的启发式函数?

- 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

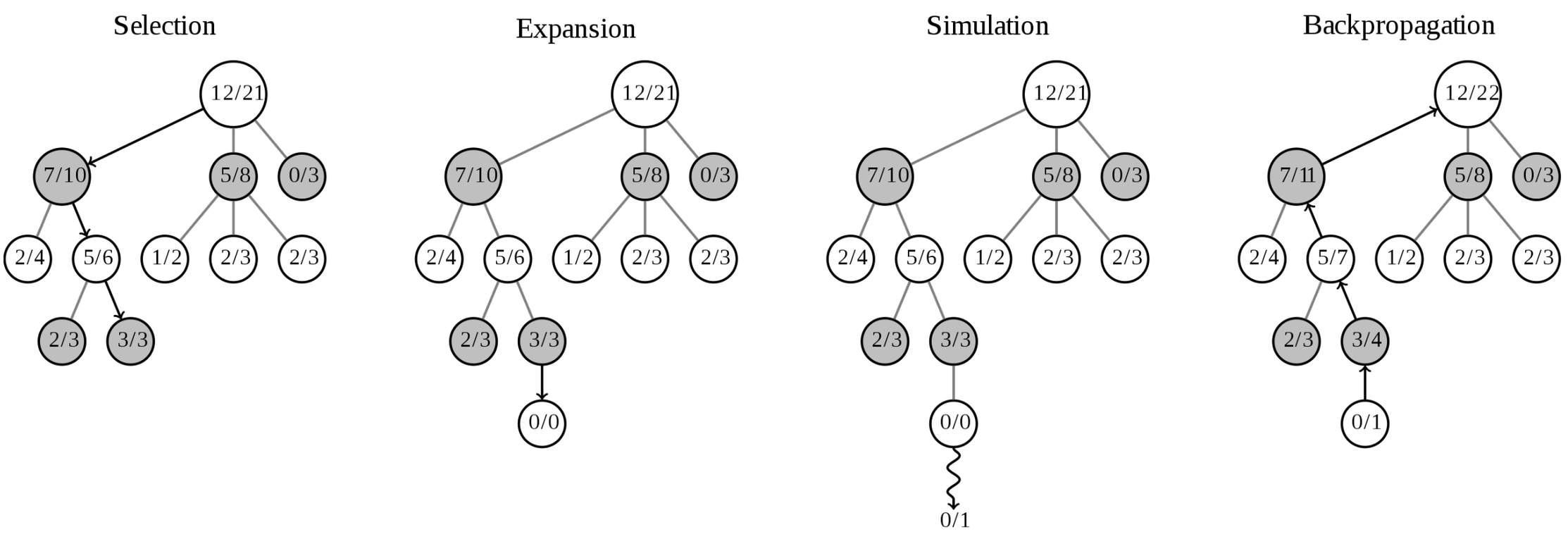
强化学习



- 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!

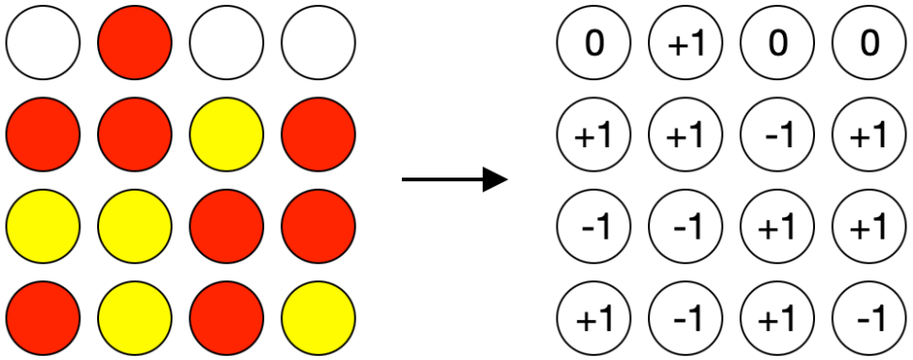
反馈太慢怎么办?

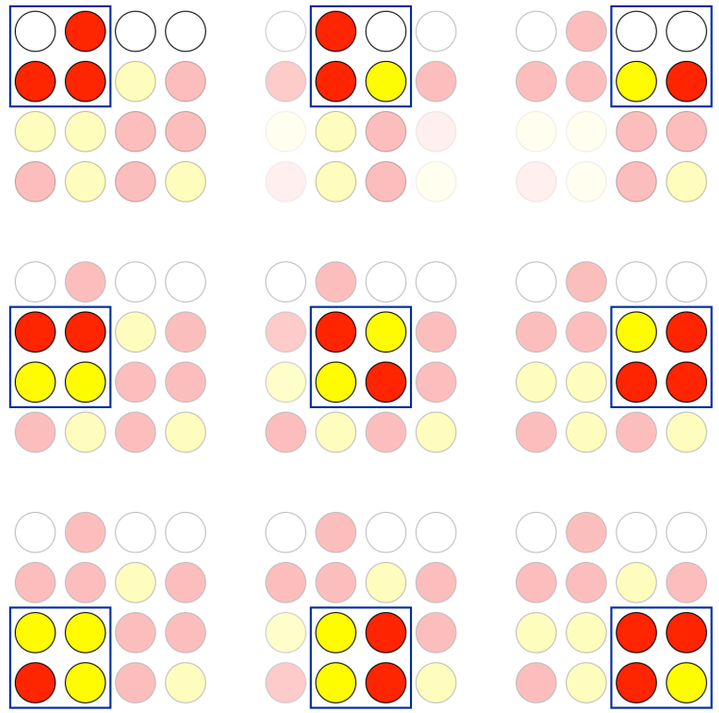
- 蒙特卡洛树搜索

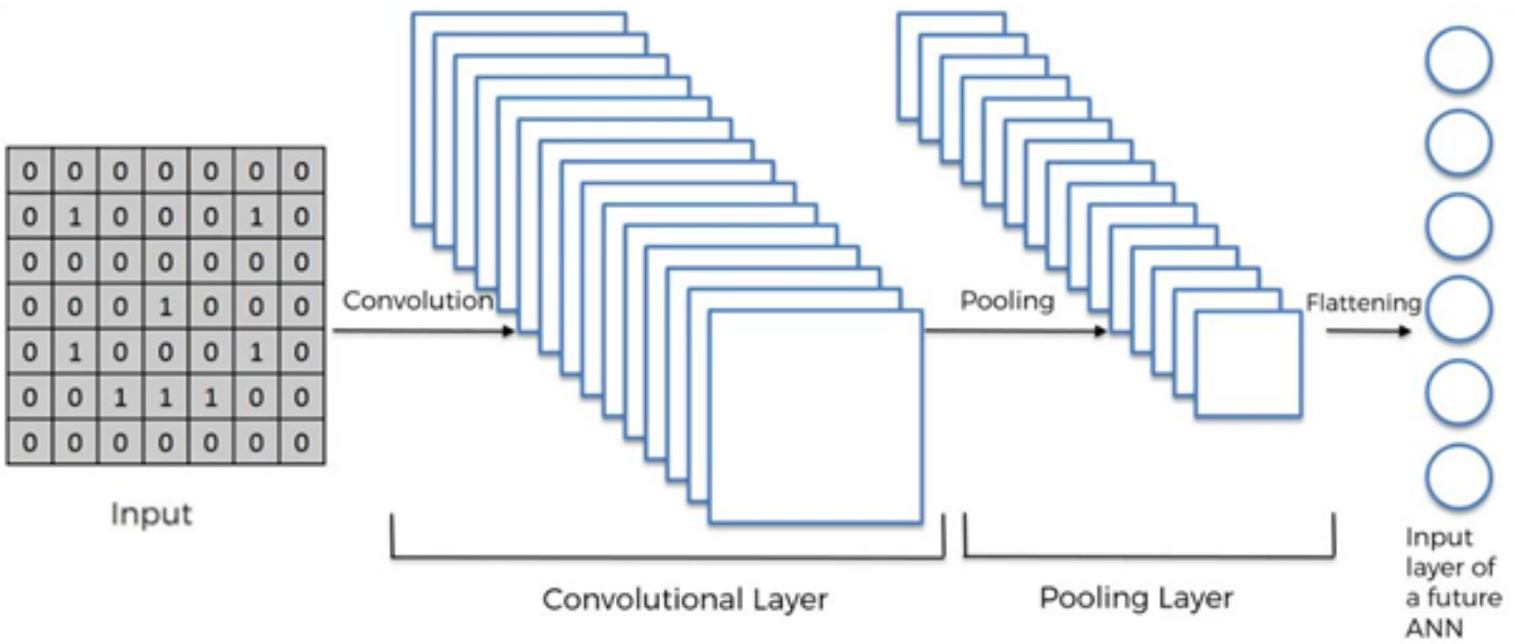


增强评价函数的泛化能力

- 卷积神经网络







实验要求

- 1. 学习并掌握Connect 4
- 2. 实现一个你自己的下棋算法，战胜difficulty=4的基本款Minimax算法
- 3. 与其他同学的算法进行比赛，夺得冠军