

hw4

1. 本题以井字棋（圈与十字游戏）为例练习博弈中的基本概念。定义 X_n 为恰好有 n 个 X 而没有 O 的行、列或者对角线的数目。同样 O_n 为正好有 n 个 O 的行、列或者对角线的数目。效用函数给 $X_3 = 1$ 的棋局 +1，给 O_3 的棋局 -1。所有其他终止状态效用值为 0。对于非终止状态，使用线性的评估函数定义为

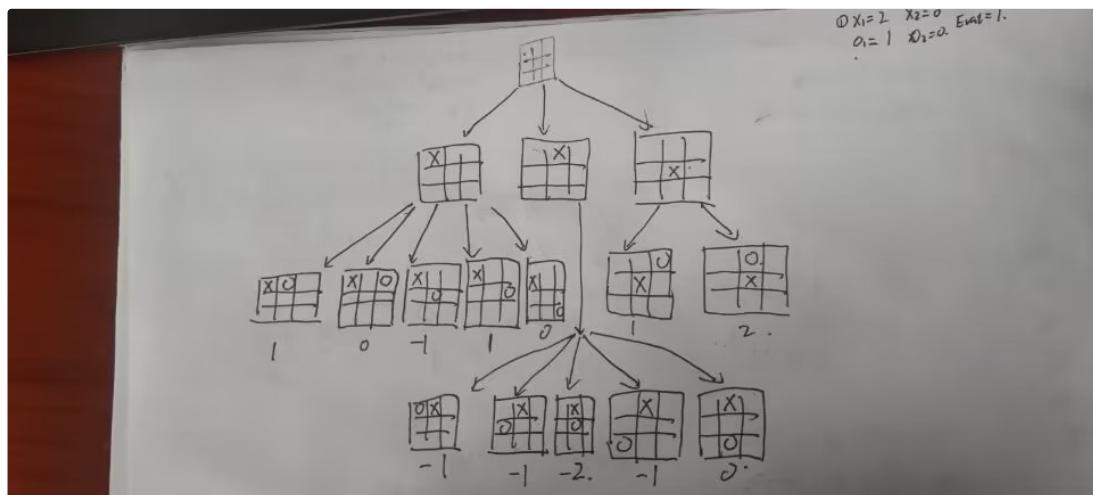
$$Eval(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$$

1. 估算可能的井字棋局数。

每个人轮流，故有 $9!$ 种棋局

2. 考虑对称性，给出从空棋盘开始的深度为 2 的完整博弈树(即，在棋盘上一个 X 一个 O 的棋局)

假设 X 先动，状态空间也是对称的



3. 标出深度为 2 的棋局的评估函数值

如图，已经标出

4. 使用极小极大算法标出深度为 1 和 0 的棋局的倒推值，并根据这些值选出最佳的起始行棋

深度为 1 的 min 值分别是： -1, -2, 1

深度为 0 的 max 值是： 1

因此初始行棋选中间

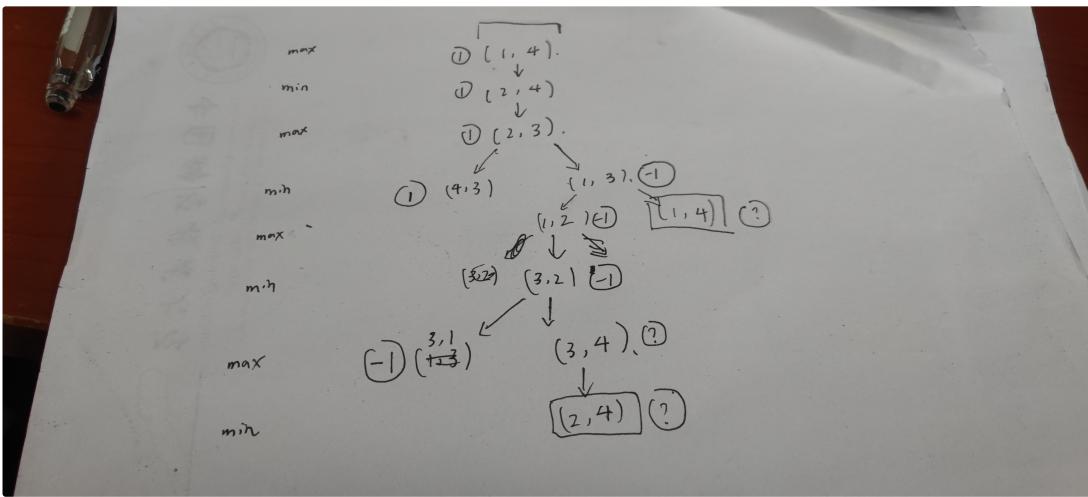
5. 假设结点按对 $\alpha - \beta$ 剪枝的最优顺序生成，圈出使用 $\alpha - \beta$ 剪枝将被剪掉的深度为 2 的结点。

如果先处理左 1，再处理左 3，中间整棵树都会被裁剪

2. 考虑图 5.17 中描述的两人游戏

1. 根据如下约定画出完整博弈树：

- 每个状态用 (s_a, s_b) 表示，其中 s_a 和 s_b 表示棋子的位置。
- 每个终止状态用方框画出，用圆圈写出它的博弈值。
- 把循环状态（在到根结点的路径上已经出现过的状态）画上双层方框。由于不清楚他们的值，在圆圈里标记一个“?”。



2. 给出每个结点倒推的极小极大值(也标记在圆圈里)。解释怎样处理“?”值和为什么这么处理。

如上图,也给出了所有的minmax值。

如果当前是min,出现两个 ? 则返回 ?, 出现一个-1,一个 ? 返回-1(因为已经不可能有比-1还差的情况了),出现一个1,一个 ? 返回 ? (没有比1更好的情况了)

同理,如果当前是max,出现两个 ? 则返回 ?, 出现一个-1,一个 ? 返回?, 出现一个1,一个 ? 返回1

3. 解释标准的极小极大算法为什么在这棵博弈树中会失败,简要说明你将如何修正它,在(b)的图上画出你的答案。你修正后的算法对于所有包含循环的游戏都能给出最优决策吗?

因为会遇到循环的状态而陷入无穷无尽的搜索中。

修正方法:记录所有已经出现过的状态,一旦出现循环则标记'?',然后用(2)中的方式,不再对循环状态进行搜索。

不一定,因为这是零和博弈问题,且评价函数只有两个取值,所以可以直接估算出取值

4. 这个4-方格游戏可以推广到n个方格,其中n>2。证明如果n是偶数A一定能赢,而n是奇数则A一定会输



Figure 5.17 The starting position of a simple game. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is +1; if player B reaches space 1 first, then the value of the game to A is -1.

采用归纳法。

奇数:

n=3时, (1,3)->(2,3)->(2,1),仅有一种情况, b必然获胜

假设n = 2k - 1时, b必然获胜

当n = 2k + 1时, 仅有一种情况(1, 2k + 1) → (2, 2k), 相当于变成了n = 2k - 1的情况, 故b获胜

偶数:

n=4时, (1, 4) → (2, 4) → (2, 3) → (4, 3), a可以以这种策略获胜

假设n = 2k时, a必然获胜。

当n = 2k + 2时, 初始状态必然转换为(2, 2k + 1), 变成了n=2k的情况, 故a一定赢

3. 请给出 $\alpha - \beta$ 剪枝正确性的形式化证明。要做到这一点需考虑**图5.18**。问题为是否要剪掉结点 n_j , 它是一个**MAX**结点, 是 n_1 的一个后代。基本的思路是当且仅当 n_1 的极小极大值可以被证明独立于 n_j 的值时, 会发生剪枝。

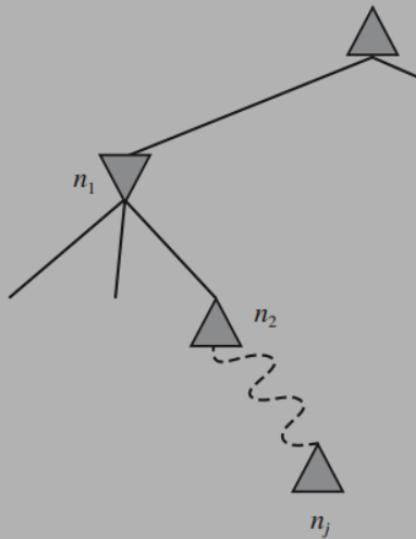


Figure 5.18 Situation when considering whether to prune node n_j .

1. n_1 的值是所有后代结点的最小值: $n_1 = \min(n_2, n_{21}, \dots, n_{2b_2})$ 请为 n_2 找到类似的表达式, 以得到用 n_j 表示的 n_1 的表达式。

同理可得 $n_2 = \max(n_3, n_{31}, \dots, n_{3b_3})$

同理 n_3, n_4, \dots, n_{j-1} , 都可以被这样替换。

而可以逐级回代得到 $n_1 = \min(\max(f(n_j), n_{31}, \dots, n_{3b_3}), n_{21}, \dots, n_{2b_2})$

2. 深度为*i*的结点 n_i 的极小极大值已知, l_i 是在结点 n_i 左侧结点的极小值 (或者极大值)。同样, r_i 是在 n_i 右侧的未探索过的结点的极小值(或者极大值)。用 l_i 和 r_i 的值重写 n_1 的表达式

n_1 可以表示为 n_2, l_2, r_2 , 的最小值, 故 $n_1 = \min(n_2, l_2, r_2)$

同理 l_2 可以继续被向下扩展 $n_2 = \max(n_3, l_3, r_3)$

最后可以表示为 $n_1 = \min(\max(f(n_j), l_3, r_3), l_2, r_2)$

3. 现在重新形式化表达式, 来说明为了向 n_1 施加影响, n_j 不能超出由 l_i 值得到的某特定界限。

如果 n_j 大于 l_i , 在上一层min结点就不会选择 n_j , 也就不能对 n_1 产生影响。因此 n_j 不能超过 $\min(l_2, l_4, \dots, l_j)$

因此可以用这个值来对整个n做剪枝

4. 假设 n_j 是MIN结点的情况, 请重复上面的过程

如果 n_j 小于 l_j , 就不会对上面的有影响, 故 n_j 要大于 $\max(l_3, l_5, \dots, l_j)$