

博弈树搜索

1.适用于具有**完全信息**的**两人零和有限确定性博弈**。

2.传统博弈树搜索：基于Alpha-beta剪枝的极小极大算法

3.搜索问题：对每一个可能的竞争者要**做出指定的反应；**

不可能找到确切的结果，所以必须求近似解。

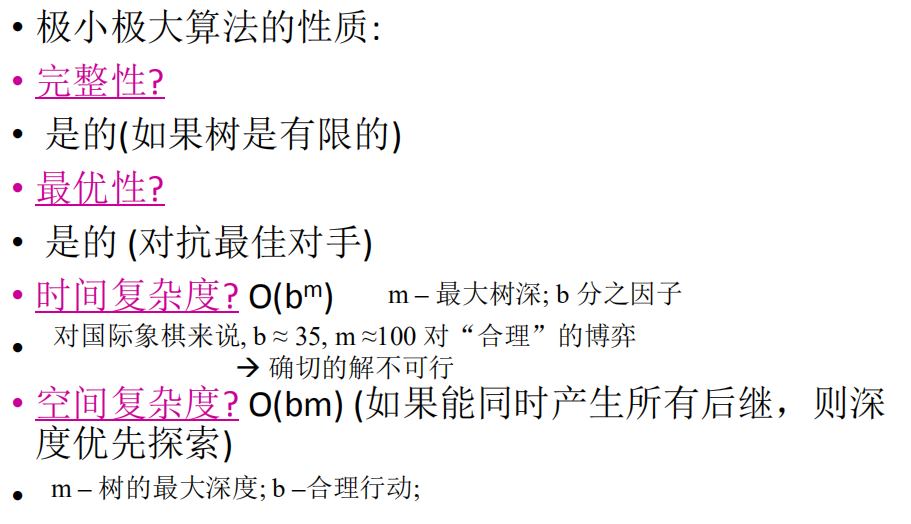
极小极大算法：

1.**二人博弈**游戏

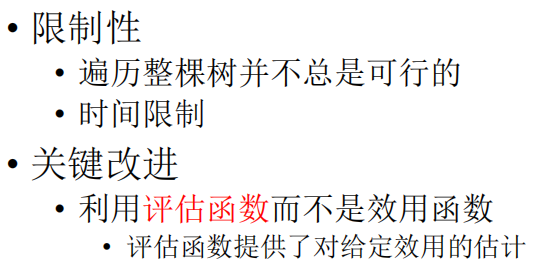
2.Max会最大化其得分、Min会最小化Max的得分

3.目标是达到最优的**极小极大值**，根据**最佳策略确定最佳可实现收益**。

4.性质：



5.局限性：



理解alpha-beta剪枝操作中的判断条件：alpha>=beta即不考虑下一次过程，这可以理解为双方已经达成了妥协，有以下两种情况：（1）min方给了自己的一个评判标准，现在我掌握到的信息

理解alpha-beta剪枝操作中的深度优先遍历：就是从根节点->左子树->右子树依次添加有效信息的过程。扫描左子树后回馈到根节点，得到的是依据左子树得到的全部有效信息作出的最有效选择（即一条路径以及这条路径的最终奖励）；扫描右子树时，我们就会根据根节点已经获取的（左子树在这时是部分的有效信息下的）部分最有效选择，与右子树逐步显现的有效信息进行判断，这时以alpha和beta作为依据进行比较，判断是否alpha>=beta，如果成立，则说明根节点根据之前的部分最有效选择比当前给定的新有效选择更好，且以后即使再新增新有效选择也不会改变双方之后的选择，则说明可以不考虑这棵子树下的任何可选项。

**蒙特卡洛树搜索——**决策树的启发式搜索算法

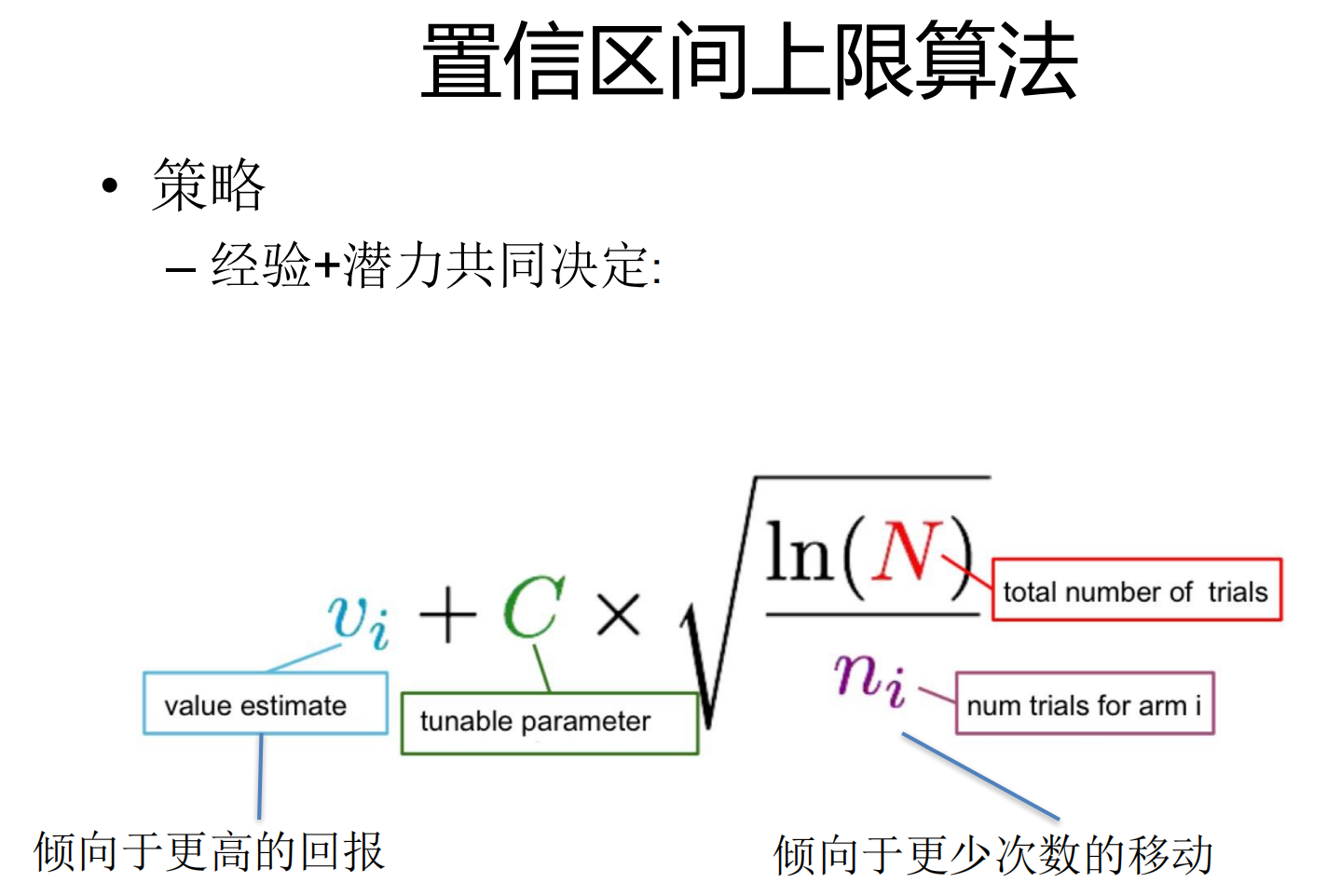
1.Selection

2.Expansion

3.Simulation

4.Backpropagation

5.UCB公式



其中 v\_i 是节点估计的值，n\_i 是节点被访问的次数，而 N 则是其父节点已经被访问的总次数。C 是可调整参数。

**AlphaGo算法区别**

前面我们实现的这种基于UCB的MCTS算法，实际上离训练围棋机器人的算法还是比较远的，其中AlphaGo也是基于MCTS算法，但做了很多优化。

我简单看了下 http://www.algorithmdog.com/alphago-zero-notes 的介绍，AlphaGo不仅替换了经典实现的UCB算法，而是使用policy network的输出替换父节点访问次数，同样使用子节点访问次数作为分母保证exploration，Q值也改为从快速走子网络得到的所有叶子节点的均值，神经网络也是从前代的CNN改为最新的ResNet，这样复杂的神经网络模型比一般的UCB算法效果会好更多。

首先，AlphaGo每个节点可选action太多了，selection阶段不能像前面先遍历完所有子节点再expansion，这里是用改进的UCB算法来找到最优的需要expansion子节点，算法基本类似也是有控制exploration/exploitation的常量C并且与该子节点visit times成反比。

其次，进行expansion时不会像前面这样直接random选择任意的action，而是这里也考虑到exploration/exploitation，一般前30步根据visit times成正比来选择，这样可以尽可能得先探索（根节点加入了狄利克雷分布保证所有点都经过），后面主要是根据visit times来走了。

第三，新版AlphaGo Zero去掉了基于handcraft规则的rollout policy，也就是快速走子网络，以前是必须有快速走子直到完成比赛才能得到反馈，现在是直接基于神经网络计算预估的winer value概率值，然后平均得到每个子节点的state-action value也就是Q值。

第四，AlphaGo在MCTS基础上收集最终的比赛结果作为label，MCTS作为policy evalution和policy iteration来实现增强学习。

当然AlphaGo在MCTS上还可以做更懂优化，只是一旦我们理解了MCTS的核心原理，看这些paper和介绍就更加清晰明了。

蒙特卡洛在游戏中的应用：参考以下几种可能性：

1.五子棋

2.井字棋

困难：

1.如何渲染棋局？

2.如何实现AI和用户之间的交互？

3.AI-MCTS如何嵌入到游戏进程中？

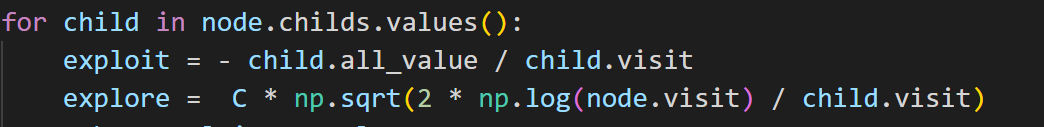
需要进一步修改的地方：

1.如果是五子棋而不是井字棋呢？

重点是如何判断终止条件

2.在Game.py State类的get\_new\_state函数中，如果new\_borad不是深拷贝的话，当对new\_board进行操作的时候会出现问题。

3.在计算UCB公式的时候，注意，对于当前节点选择已经扩展过的所有子节点中的最优的子节点的过程中，左边价值估计的时候，要明白物理意义：左边是子节点的价值估计，子节点是对方的利益估计，对于我当前的结点，我的目的是最大化自己的利益，在零和博弈中，相当于最小化对方的利益，所以要添加负号。



4.后续需要改进的地方：

（1）模拟过程中，改掉随机模拟策略，而是使用如AlphaGo的快速走棋网络

5.算法本身不足的地方：

（1）通过调整游戏棋盘大小本身，即调整状态空间的大小，可以看到为了让MCTS算法找到最优的选择方式需要更多的搜索次数。如果某个游戏设置的MCTS搜索次数上限不够大，则容易出现在有限时间内找不到给定状态下的最佳行动方式。