三、对抗搜索

> 不可能穷举

极小-极大模型

向前看若干步，从最不利的情景中选择最有利的走法

每一层对方走时选最小，每一层我方走时选最大，树根为当前应该选的走法

> 限定深度也不可能穷举

极小-极大模型的问题：

- 类似 BFS，所需存储空间太大 $\Longrightarrow$ 改进为 DFS
- 本质还是穷举，考虑很多步时仍需要极多时间

α - β 剪枝算法

和极小-极大算法得到的结果一样，但计算量小

极大节点的下界为 α，极小节点的上界为 β。

后辈节点的 β 值 $\leq$ 祖先节点的 α 值时，剪枝；

后辈节点的 α 值 $\geq$ 祖先节点的 β 值时，剪枝；

所有祖先节点中的任意一个满足即可剪枝

（整体过程是 DFS，**剪枝即为当前节点提前结束递归**，直接回溯父节点）

情况的估值：根据总结的专家知识

搜索深度越深效果越好

> 注意：α - β 剪枝只是得到一步结果，接下来每一步都需要再用一次算法

α - β 剪枝在围棋上失败，状态多不是本质原因。本质原因在于 α - β 剪枝依赖于局面评估的准确性，而围棋存在知识的统一性问题。

蒙特卡洛树搜索

Monte Carlo Tree Search，一种随机模拟方法

选择 $\Longrightarrow$ 扩展 $\Longrightarrow$ 模拟 $\Longrightarrow$ 回传

选择策略，考虑两方面因素：对尚未充分了解的（之前表现不好的）节点的探索；对当前具有较大希望的（之前表现好的）节点的利用

蒙特卡洛树搜索的问题：要生成所有子节点，且模拟具有盲目性，导致了模拟量太大

信心上限算法（UCB）

信心上限：$I_j = \bar{x}_j + \sqrt{\frac{2\ln(n)}{T_j(n)}}$

$\bar{x}_j$ 是拉杆 j 获得回报的均值；n 是当前访问总次数；$T_j(n)$ 是拉杆 j 访问次数

将 UCB 用于蒙特卡洛树搜索中

信心上限树算法（UCT）：实际计算时，$I_j = \bar{x}_j + c \times \sqrt{\frac{2\ln(n)}{T_j(n)}}$ 使用参数 c 进行调节

选下棋落子时，$\bar{x}_j =$ 获胜次数 / 模拟总次数。（获胜次数是从节点对应方角度说的）

每次选择时，双方都选 I_j 最大的点（因为认为对方也选对他最有利的落子）

（随机）模拟要一直模拟到决出胜负，然后向上回传 1 或 -1

AlphaGo

AlphaGo 将神经网络与蒙特卡洛树搜索结合，缩小了搜索范围，同时提高了模拟水平

AlphaGo 用的两类网络：

策略网络（policy network）

输入：当前棋局（48 个通道，每个通道 19×19）

输出：每个点的落子概率

种类：

- SL，有监督
- RL，强化学习
- Rollout Policy，推演策略网络（特点：浅，快速，用于随机模拟）

结构：输入 $\Longrightarrow$ 卷积 + ReLu $\Longrightarrow$ 卷积 + ReLu $\Longrightarrow$ ... $\Longrightarrow$ 卷积 + softmax $\Longrightarrow$ 输出（共 13 层）

等效为分类问题：棋盘有 $19 \times 19 = 361$ 个点，故有 361 个类别

损失函数：交叉熵 $L(w) = -t_a \times \log(P_a)$

t_a：当前棋局的实际落子在 a 处则为 1 ，否则为 0

P_a：策略网络输出的 a 处落子概率

估值网络（value network）

输入：49 个通道

输出：当前棋局收益 $[-1, 1]$

等效为回归问题：获胜时收益为 1，失败为 -1

误差平方和损失函数：$L(w)=(R-V(s))^2$

R：标签（棋局胜负）

$V(s)$：预测结果

节点 s 第 i 次模拟的收益：$v_i(s)=\lambda \text{ value }(s)+(1-\lambda) \text{ rollout }(s).$

AlphaGo 与 MCTS 不同之处：选择直到叶节点为止，生成其所有子节点，但只对被选中的叶节点用 Rollout Policy 进行模拟（模拟要直到决出胜负）。没有被模拟过的节点 $\text{rollout }(s)=0$ 。

每个节点记录的信息：总收益；行棋到该节点的概率；被选择的次数

根节点的子节点中被选择次数最多的节点作为最终走步。（而不是收益均值最大的，因为均值最大的可能模拟次数少，不够可靠，被选次数最多的更可靠）

模拟后将模拟结果 $\text{rollout }(s)$ 与 $\text{value }(s)$ 加权平均得到收益 $v(s)$ /再将 $v(s)$ 向上回传（$v(s)\in[-1,1]$，对方应 $-v(s)$ ，本方 $+v(s)$ ）。更新各节点的总收益。

围棋中的深度强化学习方法

强化学习

学习者不会被告和如何去做，必须自己通过尝试发现哪些动作会产生最大的收益

两个特征：试错和延迟收益（进行一系列动作后才给反馈）

深度强化学习（关键是如何获得指示信号）

用深度学习（神经网络）方法实现的强化学习

- 将收益转化为“标注”（不能获得所有情况下既正确又有代表性的示例）
- 将深度强化学习问题转化为神经网络训练问题。不同转换方法构成了不同的深度强化学习方法，关键是损失函数的定义

通过自己博弈训练策略的网络

三种实现方法：

基于策略梯度的强化学习

数据：自我博弈产生。(s,a,P_a,t_a)

P_a：s 棋局在 a 处走子的获胜概率（网络输出）

t_a：胜负值，胜为 1 ，负为 -1

损失函数： $L(w)=-t_a\times\log(P_a)$

- 假设胜者的行为都正确，负者的行为都不正确
- 假设获负时对权重的修改量大小与获胜时一样，方向相反

正常情况，交叉熵损失函数的 t_a ，P_a 都 $\in[-1,1]$ 。此处 t_a 可能为 -1 ，故需额外假设 $t_a=1$ ，使 P_a 上升；$t_a=-1$ ，使 P_a 下降；

> 强化学习流程，见 PPT 第 66 页

注意点：

1. **强化学习流程中，每个样本只使用一次。原因：样本噪声大，不一定可靠；样本由自身博弈产生，数量无上限**
2. 基于策略梯度的强化学习学到的是每个可落子点走子的获胜概率。不同于监督学习策略网络（目的是模仿人）学到的是某个可落子点走子的概率

基于价值评估的强化学习

学习到的是每个可落子点获取最大收益的概率。

（tanh）价值评估网络：对一个行棋点的价值（收益）进行评估，输出为 $[-1,1]$ 间的估值。

数据：自我博弈产生。$(s,a,V(s,a),R)$

$V(s,a)$：s 棋局在 a 处走子时，评估网络的输出

R：胜负值，胜为 1 ，负为 -1

损失函数；$L(w)=[R-V(s,a)]^2$

基于演员-评价方法的强化学习

收益增量：评价一步棋的好坏

$A=Q(s,a)-V(s)$（ $V(s)$ 是棋局 s 的预期收益； $A\in[-2,2]$ ）

$Q(s,a)$ 为在 a 处行棋后的收益，不好计算，延迟为最终的胜负值 R $\Longrightarrow A=R-V(s)$

损失函数

- 评价部分：$L_2(w)=[R-V(s)]^2$
- 演员部分：$L_1(w)=-A\log(P_a)$，A 为收益增量 $A=R-V(s)$
- 综合损失函数：$L(w)=L_1(w)+\lambda\times L_2(w)$ ，λ 是调节系数

> 网络结构，见 PPT 第 74 页

本方法强调的是重要行棋点的学习，通过收益增量对走法的重要性进行评价，学习到的是每个落子点获得最大收益增量的概率

AlphaGo Zero

- 不再使用人类棋手的数据
- 不再使用人工特征作为输入
- 利用强化学习从零学习

将策略网络和估值网络合并为一个“双输出”网络（多了一个“放弃”行为，变成 $19 \times 19 + 1$ 分类，其 MCTS 与 AlphaGo 的不同：节点 s 第 i 次模拟的收益 $v_i(s) = \text{value}(s)$，不再要随机模拟的结果

将 MCTS 结合到深度强化学习中

损失函数：

- 估值网络的不变：$[R - V(s)]^2$
- 策略网络：$L_2 = -\pi_1 \log(P_1) - \ ... \ - \pi_{362} \log(P_{362})$，（$P_i$ 为策略网络输出的概率）
- 综合损失函数：$L = L_1 + L_2 + \|\theta\|_2^2$，最后一项是正则化项，用于减少过拟合

> π_i 是 MCTS 中由选中次数转化而来的概率（含“放弃”）

引入多样性：

防止走向错误方向，人为引入噪声。对策略网络的输出增加噪声。噪声不会引起不良反应，MCTS 有纠错能力

落子概率：$\lambda \times P_a + (1 - \lambda) \times P_d$

P_a 是策略网络输出，P_d 是引入的狄利克雷分布采样

AlphaGo Zero 强化学习的对弈中，落子由策略-估值网络 + MCTS 决定