应用于黑白棋的几种智能算法

16337341 朱志儒

January 2019

1 摘要

极具趣味性和挑战性的学科非人工智能莫属，许多科研人员都在为之奋斗，将人工智能算法应用于传统的棋类游戏是人们关注和研究人工智能的内容之一。黑白棋 [3] 是 19 世纪末英国人发明，上个世纪 70 年代日本人长谷川五郎将其发展的棋类游戏。与其他棋类游戏（例如，中国象棋、国际象棋和围棋）相比，黑白棋每层搜索的节点数量并不是特别庞大，其游戏规则也较为简单，所以棋局的评估函数可以做的较为细致。

本论文将设计并实现复杂的评估函数，并将强化学习中的 Q-learning 和 Sarsa 算法以及博弈树搜索中的 Minimax 算法和 Alpha-beta 剪枝算法这四种智能算法运用于黑白棋，比较这四种算法的智能程度以及效率。最后，经过测试，Alpha-Beta 剪枝算法比 Minimax 算法在效率上有相当大的提高，且它们在黑白棋上的智能较高，而 Sarsa 与 Q-learning 则在黑白棋上的智能较低。

2 引言

黑白棋 [3] 使用8×8 的棋盘，下棋时将棋下在方格中间。开始时，在棋盘的正中间有两个黑棋和两个白棋交叉放置，黑棋先手，双发轮流落子。下子时必须在一个空位下子并且新落子的棋子必须与己方的棋子在同一直线（横、竖、斜）上夹着对方的棋子，这些被夹的棋子将转变成己方棋子。如果有一方没有合法位置下子，则对手可继续下子直到该方有位置可下。如果轮到一方下子且有位置可下，则其必须下子而不能弃权。黑白双方轮流下

子，直到棋盘下满或双方都没有合法位置可下，则游戏结束，棋盘上棋子多的一方获胜。

在黑白双方对弈的过程中，计算机需要对整个棋局进行分析和评估，将整个棋局为分多个方面的因素，每个因素根据重要性分配相应的权重，最后进行加权求和得到整个棋局的评估值。考虑的因素越多，对整个棋局的分析和评估也就越准确，但效率也就越底下。所以，对整个棋局的分析和评估需要考虑数量和效率，并做好二者的权衡。

本论文在对棋局进行分析和评估时，将棋局抽象成下列因素：基于位置特征的估计值、基于黑白子比例的估计值、基于行动力的估计值、基于近角位的估计值、基于稳定子的估计值和基于角位的估计值，为这些因素分配不同的权重，最后将这些因素加权求和得到棋局的估计值。

本论文的主要研究的内容是为黑白棋的棋局设计复杂的评估函数，将强化学习中的 Q-learning 算法和 Sarsa 算法以及博弈树搜索中的 Minimax 算法和 Alpha-Beta 剪枝算法运用到黑白棋，并比较这四种算法的智能程度以及效率。

3 方法

3.1 评估函数

3.1.1 基于位置特征的估计值

黑白棋和围棋相似，有“金边银角草肚皮”的说法，棋子在四个角的优势巨大，因为在四角的棋子不可能被翻转；然而在近角位的优势小，因为这些位置容易让对方占角或是被对方翻转大量的棋子；四条边上的其他位置的优势也比较大，因为迅速占边可以比较容易地获得边界稳定子的优势；而在棋盘的中心位置优势较低。棋盘的所有位置的权重值如下：

20	-3	11	8	8	11	-3	20
-3	-7	-4	1	1	-4	-7	-3
11	-4	2	2	2	2	-4	11
8	1	2	-3	-3	2	1	8
8	1	2	-3	-3	2	1	8
11	-4	2	2	2	2	-4	11
-3	-7	-4	1	1	-4	-7	-3
20	-3	11	8	8	11	-3	20

计算己方所有棋子权重的和与对方所有棋子权重的和，再相减就可以得到基于位置特征的估计值。

3.1.2 基于黑白子比例的估计值

在黑白棋中，如果己方棋子比对方棋子数量多，则说明己方占优，如果对方棋子比己方棋子数量多，则说明对方占优。在实际对战过程中，这项估计值的参考价值并不是特别大，因为黑白子比例与谁将下子关系很大，比如说，在己方棋子数目远小于对方棋子数目的情况下，己方着子后可能翻转对方大量的棋子，从而会逆转局势。所以黑白子比例的估计值在棋局估计值中占比较低。

3.1.3 基于角位和近角位的估计值

在黑白棋中，角位是无法被翻转的，占领角位将带来极大的优势，所以在角位附加一项极大的估计值。而落子于近角位将很可能把角位送给对方，这将会给己方带来极大的劣势，所以在近角位附加一项极小的负估计值。角位和近角位的估计值在棋局估计值占比较高。

3.1.4 基于稳定子的估计值

在黑白棋中，稳定子是棋局估计值的重要指标，拥有更多的稳定子，既能保证己方最少的棋子数，又能帮助己方翻转大量对方的棋子，形成成片稳定子的作用。稳定子可分为两种：内部稳定子和边界稳定子，内部稳定子的作用并没有比边界稳定子大，因为内部稳定子并不一定能帮助己方翻转对方棋子。计算内部稳定子的估计值时，只需为每个内部稳定子赋予一个相同

的权重。对于边界稳定子的估计值的计算，从棋盘的一个角开始，权重分别为：

1	2	3	4	5	6	7	8

这样设计权重将鼓励占边，形成成片的稳定子，加大优势。稳定子的估计值在棋局估计值中占比较大。

3.1.5 基于行动力的估计值

在黑白棋中，行动力是指棋盘上某一方的可下子位置个数，行动力较高时，可保证之后的若干步都会有较好的下法。与黑白子比例的估计值相似，行动力的估计值也采用比例算法，但需要考虑两种特殊情况：

1. 如果己方没有位置下子，那么设置特低的行动力估计值
2. 如果对方没有位置下子，那么设置特高的行动力估计值

这样就可以避免己方处于无子可下的糟糕局面，而倾向于选择使对方无子可下的优势局面。行动力估计值在棋局估计值中占比较高。

3.2 强化学习

3.2.1 Q-learning[1]

Q-learning 在黑白棋的应用中，每个不同的状态 S 表示每个不同的棋局，动作 a 表示下一步下子的位置，动作集 A 表示可下子位置的集合，状态 S 的即时回报 R 表示 S 所对应棋局的估计值。Q-learning 在黑白棋中的步骤为：

1. 初始化棋局 S 和表 Q

2. 进入循环，根据 ϵ-greedy 算法在棋局 S_i 中选择下一步下子位置 A_i，在位置 A_i 下子且对方也下完子后得到棋局 S_{i+1} 和 S_{i+1} 的估计值 R_{i+1}，然后更新表 Q，即

$$\mathrm{Q}(S_i,\, A_i) = \mathrm{Q}(S_i,\, A_i) + \alpha(R_{i+1} + \lambda max\mathrm{Q}(S_{i+1},\, a_{i+1}) \text{ - } \mathrm{Q}(S_i,\, A_i))$$

其中 $max\mathrm{Q}(S_{i+1},\, a_{i+1})$ 表示在棋局 S_{i+1} 中对于所有可下子位置选择对应最大的 Q 值。

3. 不断循环直至游戏结束

3.2.2 Sarsa[1]

Sarsa 在黑白棋的应用中，状态 S、动作 a、动作集 A 和即时回报 R 所表示的意义与 Q-learning 在黑白棋的应用相同，Sarsa 在黑白棋中的步骤为：

1. 初始化棋局 S 和表 Q

2. 进入循环，根据 ϵ-greedy 算法在棋局 S_i 中选择下一步下子位置 A_i，在位置 A_i 下子且对方也下完子后得到棋局 S_{i+1} 和 S_{i+1} 的估计值 R_{i+1}，再根据 ϵ-greedy 算法再棋局 S_{i+1} 中选择下一步下子位置 A_{i+1}，然后更新表 Q，即

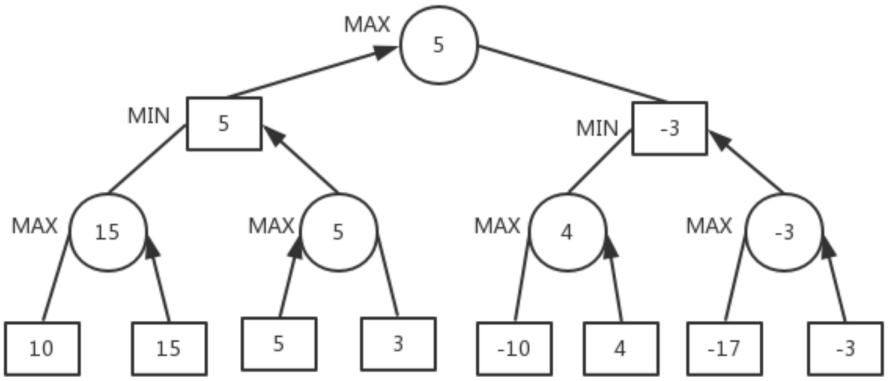
$$\mathrm{Q}(S_i,\, A_i) = \mathrm{Q}(S_i,\, A_i) + \alpha(R_{i+1} + \lambda \mathrm{Q}(S_{i+1},\, A_{i+1}) \text{ - } \mathrm{Q}(S_i,\, A_i))$$

3. 不断循环直至游戏结束

3.3 博弈树搜索

3.3.1 Minimax 算法 [2]

对博弈树进行深度优先搜索获得当前棋局之后的所有可能结果，玩家双方均会选择对自己最有利的走法，也就是说，对于己方而言，会选择最大化己方优势的走法，然而，对方会选择使己方优势最小化的走法。从博弈树来看，每一层轮流从子节点中选取 MAX-MIN-MAX-MIN...



深度优先搜索的深度限制为 4 层时，图中第 4 层为叶子节点；第 3 层为己方根据第 4 层的棋局估计值推出该层节点的棋局估计值，选取的是子节点中的最大值，即最大化自己的优势；第 2 层为对方根据第 3 层的棋局

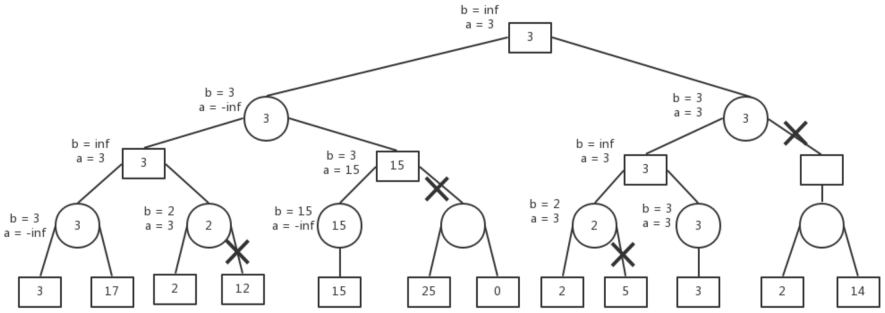
估计值推出该层节点的棋局估计值，选取的是子节点中的最小值，即最小化己方的优势；第 1 层为己方根据第 2 层棋局估计值推出该层节点的棋局估计值，选取的是子节点中的最大值，即最大化己方的优势，从而己方就可以得到下一步最大化优势的走法。

3.3.2 Alpha-Beta 剪枝算法 [2]

Alpha-Beta 剪枝建立在 Minimax 算法的基础上，但它减少了 Minimax 算法搜索树的节点数。

Alpha 剪枝过程：对于 MIN 层的节点，如果估计出其倒推值的上确界 beta 小于或等于其 MAX 层父节点的估计倒推值的下确界 alpha，即 beta≤alpha，则不必再扩展该 MIN 层的其余节点，因为其余节点的估计值如何已经不能改变父节点的 MAX 值了。

Beta 剪枝过程：对于 MAX 层的节点，如果估计出其倒推值的下确界 alpha 大于或等于其 MIN 层父节点的估计倒推值的上确界 beta，即 beta≤alpha，则不必再扩展该 MAX 层的其余节点，因为其余节点的估计值如何已经不能改变父节点的 MIN 值了。



如上图显示的 Alpha-Beta 剪枝过程，方框为 MAX，圆圈为 MIN。初始设置 a = -∞，b = +∞，深度优先搜索到达左下角的节点，第四层 MIN 层取子节点中的最小值 3，3 < b，则 b = 3；回溯到第三层 MAX 层，3 > a，则 a = 3，b 不变还是为正无穷；深度优先搜索右子树到第五层，返回值为 2；回溯到第四层 MIN 层节点，2 < b，则 b = 2，而此时 a = 3，就有 b < a，所以进行剪枝，不用搜索第四层 MIN 节点的右子节点，第四层 MIN 层节点返回 2；回溯到第三层 MAX 层节点，2 < 3，a 不变，返回值为 3；回溯到第二层 MIN 层节点，3 < b，则 b = 3；深度优先搜索到第五层节点，返回值为 15；回溯到第四层 MIN 层节点，没有其他的子节点，所

以返回值为 15；回溯到第三层 MAX 层节点，15 > a，则 a = 15，此时 b = 3，就有 b < a，所以进行剪枝，不需要探索第三层 MAX 层节点的右子节点，第三层 MAX 层节点返回值为 15；回溯到第二层 MIN 层节点，3 < 15，所以其返回值为 3；回溯到根节点，3 > a，则 a = 3，再深度优先搜索根节点的右子节点，同时剪枝，最后结果如上图。

4 实验

4.1 强化学习

将 ϵ 设置为 0.5，学习速率 a 设置为 0.8，折扣因子 λ 设置为 0.5，使用评估函数就可以得到每个棋局 S 的即时回报 R。在训练 Q-learning 算法的模型时，与随机数对弈 50 万局；在训练 Sarsa 算法的模型时，与随机数对弈 100 万局。

在实际对弈时，对于当前棋局 S 以及可下子位置集 A，在训练后的表 Q 中寻找具有最大 Q 值的（S，a）对，可下子位置 a 就是当前棋局 S 最优的下子位置。如果训练后的表 Q 中没有与（S，A）对应的 Q 值，则随机选取 A 中的一个位置下子。

训练好两个模型后分别与随机数进行博弈，对弈 100 局，下表为统计数据，train_times 表示训练时与随机数对弈的总局数，rate 表示与随机数对弈 100 局的胜率，Q_numbers 表示训练 train_times 局后表 Q 中存储的（S，A）对以及对应 Q 值的数量，avg_hits 表示在每局对弈中（S，A）对出现在训练后的表 Q 的频率。

	train_times	rate	Q_numbers	avg_hits
Q-learning	500000	68%	12559838	27%
Sarsa	1000000	69%	24834550	43%

4.2 博弈树搜索

在实际对弈中，以当前棋局作为根节点，进行深度优先搜索，搜索深度限制为 6，在搜索的同时，以 Minimax 算法或 Alpha-Beta 剪枝算法构建博弈树，博弈树中叶子节点的估计值由评估函数得到，搜索并构建好整个博弈树后就可以得到最优的下子位置。

本论文选择与随机数进行博弈，对弈 100 局，下表为统计数据，其中 depth 表示搜索深度，rate 表示与随机数对弈 100 局的胜率，nodes 表示搜索过的节点总数。

	depth	rate	nodes
Minimax	6	92%	5328243
Alpha-Beta	6	92%	2939082

5 结论

从实验中可以看出，强化学习中的 Q-learning 和 Sarsa 并不适合黑白棋，因为黑白棋具有太多的棋局和对应每个棋局不同的下子位置，这导致实际运用中根本无法存储这么多种情况。在统计数据中，每次与随机数对弈时在表 Q 中找到对应的（S，A）的频率也只有 27% 和 43%。再者它与随机数对弈训练，这导致每次训练的棋局也会有所不同，从而 Q 表中对应的值收敛较慢，即使训练 100 万次，但对于 1 亿多个棋局和下子位置，这些训练次数无法使得它们对应的 Q 值收敛，对弈时并不能找到最优的下子位置，所以可以得出，Sarsa 算法与 Q-learning 算法还存在一定的局限性，因为它们的胜率远小于博弈树搜索算法的胜率。

因此，Minimax 算法以及进行 Alpha-Beta 剪枝优化以提升计算速度是更适合黑白棋的，从实验结果展示可以看出它们的胜率远远高于另外两个。且在相同的搜索深度条件下，Alpha-Beta 剪枝探索的节点数远远小于 Minimax 算法探索的节点数。

参考文献

[1] Sutton R S, Barto A G. Reinforcement learning: An introduction[J]. 2011.

[2] 王永庆. 人工智能原理与方法 [M]. 西安: 西安交通大学出版社, 1998: 288-290.

[3] 黑白棋. 百度百科. (2018-9-16) https://baike.baidu.com/item/