Contents:

Overview 概述

- The minimax algorithm
  generates the entire game search space.
  minimax算法生成整个博弈搜索空间。
- The alpha–beta pruning algorithm
  allows us to prune large parts of it.
  alpha–beta剪枝算法允许我们将其剪去大部分。
- However, alpha–beta still has to search all the way to terminal states for at least a portion of the search space.
  然而，alpha–beta仍然需要搜索抵达终端状态的所有途径、至少是搜索空间的一部分。
- This depth is usually not practical, because moves must be made in a reasonable amount of time.
  这个深度通常是不实际的，因为移动必须在合理的时间内完成。

Apply a Heuristic Evaluation Function 应用启发式评价函数

- Claude Shannon proposed instead that programs should cut off the search earlier and apply a heuristic evaluation function to states in the search, effectively turning nonterminal nodes into terminal leaves. The suggestion is in two ways:

  克劳德·香农提出：程序应该早一些剪断搜索，并在搜索中对状态应用启发式评价函数，有效地将非终端节点转换为终端叶节点。该建议是用如下两种方法：

  - Use EVAL instead of UTILITY

    用EVAL来代替UTILITY

    EVAL: a heuristic evaluation function, which estimates the position's utility.

    EVAL：一个启发式评价函数，用于估计位置的效用。

  - Use CUTOFF-TEST instead of TERMINAL-TEST

    用CUTOFF-TEST来代替TERMINAL-TEST

    CUTOFF-TEST: decides when to apply EVAL.

    CUTOFF-TEST：确定何时应用EVAL函数。

Heuristic H-Minimax vs. Minimax 启发式H-Minimax与Minimax

function H-MINIMAX(s, d) **returns** an action
  if CUTOFF-TEST(s, d) **then return** EVAL(s)
  if PLAYER(s) = MAX **then return** $\max_{a \in \text{ACTIONS}(s)}$ H-MINIMAX(RESULT(s, a), d +1)
  if PLAYER(s) = MIN **then return** $\min_{a \in \text{ACTIONS}(s)}$ H-MINIMAX(RESULT(s, a), d +1)

function MINIMAX(s) **returns** an action
  if TERMINAL-TEST(s) **then return** UTILITY(s)
  if PLAYER(s) = MAX **then return** $\max_{a \in \text{ACTIONS}(s)}$ MINIMAX(RESULT(s, a))
  if PLAYER(s) = MIN **then return** $\min_{a \in \text{ACTIONS}(s)}$ MINIMAX(RESULT(s, a))

Evaluation Functions 评价函数

- An evaluation function returns an *estimate* of the expected utility of the game from a given position. How to design good evaluation functions?

  一个评价函数返回从一个给定位置该博弈的期望效用估计。如何设计好的评价函数?

  - It should order the *terminal* states in the same way as the true utility function:
    a) the states that are wins must evaluate better than draws,
    b) the states that are draws must be better than losses.

    它应该与真实效用函数相同的方式对终端状态进行整理:
    a) 获胜状态必须评价为优于平局,
    b) 平局状态必须评价为优于失败。

  - The computation must not take too long.

    计算的时间一定不能太长。

  - Nonterminal states should be strongly correlated with actual chances of winning.

    非终端状态应该与实际获胜的机会密切相关。

Weighted Linear Function 加权线性函数

-- A kind of evaluation function 一种评价函数

$$\text{EVAL}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$
$$= \sum_{i=1}^{n} w_i f_i(s)$$

where 其中

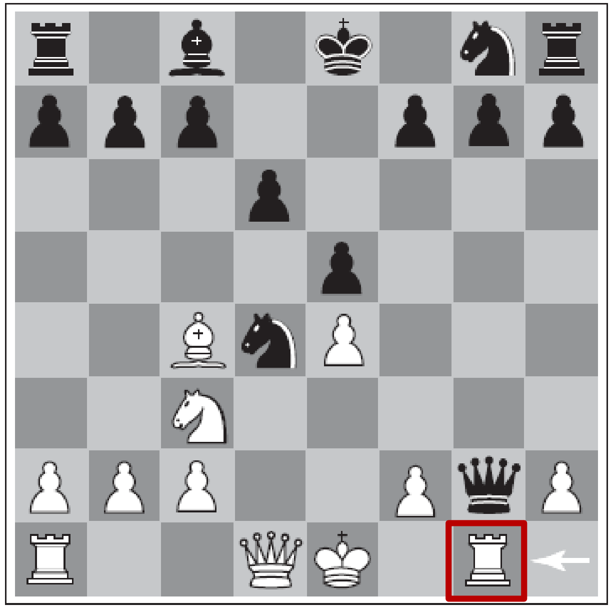
- w_i -- a weight 权重
- f_i -- a feature of the position 位置的特征

For Chess 对于国际象棋

- f_i could be the numbers of each kind of piece on the board
  应该为棋盘上每种棋子的数量
- w_i could be the values of the pieces: 1 for pawn, 3 for bishop, etc.
  应该为棋子的数值：1表示兵、3表示象、等等。

Chess 国际象棋

(a) White to move 白棋要走子



(b) White moved 白棋走子后

Two chess positions that differ only in the position of the rook at lower right:
(a) Black has an advantage of a knight and two pawns, which should be enough to win the game.
(b) White will capture the queen, giving it an advantage that should be strong enough to win.

两盘国际象棋布局仅在右下角车的位置不同：
(a) 黑棋多一个马和两个兵，应该赢得这盘棋。(b) 白棋将捕获皇后，使得它处于足以获胜的态势。