Heuristic Analysis

Tianpei Xie

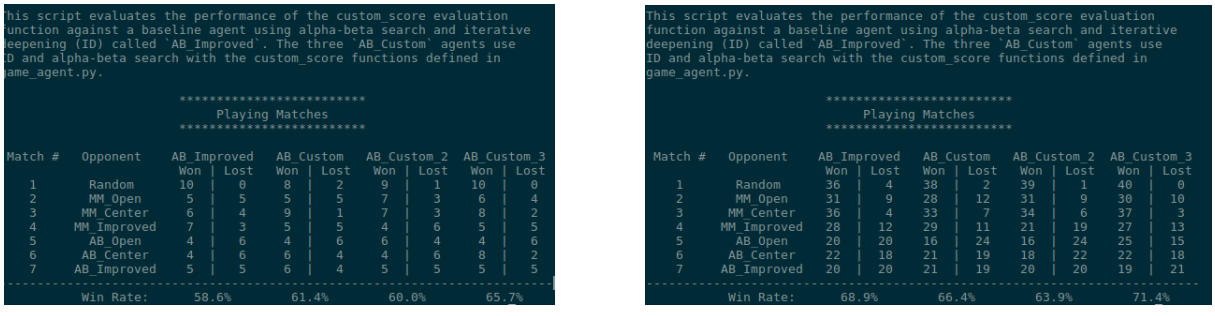
Nov. 25th., 2017

1 Heuristics

In the submitted codes, I includes two heuristics and the final main *custom_score* function is a combination of two of them.

- *custom_score_2* implements the center heuristic. It computes the distance between a move and the center of the board. The intuition is that the possible move at center of the board is largest, thus is most beneficial.
- *custom_score_3* implements a modified version of $open_moves$, which computes the difference between all possible moves for current player and all possible moves of its opponent. In this implementation, we put more weights on the opponents possible moves so that the agent will be more aggressive.
- *custom_score* combines both heuristic to have a better performance than either. It is seen that against *AB_Improved*, the proposed *AB_Custom* win with about $62\% - 66\%$ of chance. Compared with the *AB_improved* which uses a direct difference between the size of possible moves for current player with all possible moves of its opponent, we put more weights on minimizing the possible move of its opponents, which makes the agent more aggressive to block the opponent. This contributes to a slightly better chance of winning for the proposed solution.

  The included is the score from the tournament (with no. of matches = 5)



```
his script evaluates the performance of the custom_score evaluation
unction against a baseline agent using alpha-beta search and iterative
eepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
D and alpha-beta search with the custom_score functions defined in
ame_agent.py.

                        ************************
                             Playing Matches
                        ************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      10  |  0     8  |  2     9  |  1     10 |  0
    2       MM_Open      5  |  5     5  |  5     7  |  3     6  |  4
    3      MM_Center     6  |  4     9  |  1     7  |  3     8  |  2
    4    MM_Improved     7  |  3     5  |  5     4  |  6     5  |  5
    5      AB_Open       4  |  6     4  |  6     6  |  4     4  |  6
    6     AB_Center      4  |  6     6  |  4     4  |  6     8  |  2
    7    AB_Improved     5  |  5     6  |  4     5  |  5     5  |  5
--------------------------------------------------------------------
          Win Rate:      58.6%        61.4%        60.0%        65.7%
```

(a)

```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                        ************************
                             Playing Matches
                        ************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      36  |  4     38 |  2     39 |  1     40 |  0
    2       MM_Open     31  |  9     28 |  12    31 |  9     30 |  10
    3      MM_Center    36  |  4     33 |  7     34 |  6     37 |  3
    4    MM_Improved    28  |  12    29 |  11    21 |  19    27 |  13
    5      AB_Open      20  |  20    16 |  24    16 |  24    25 |  15
    6     AB_Center     22  |  18    21 |  19    18 |  22    22 |  18
    7    AB_Improved    20  |  20    21 |  19    20 |  20    19 |  21
--------------------------------------------------------------------
          Win Rate:      68.9%        66.4%        63.9%        71.4%
```

(b)

Figure 1: (a) Run with 5 matches each (b) Run with 20 matches each.

As shown above, we recommend using a *combination* of modified *open_move* and *center_heuristic* as an evaluation function. It has the following benefits

1. Both *open_move* and *center_heuristic* are easy to compute, with fast implementation avaiable. It allows the algorithm to handle large scale problem effectively.

2. The modified *open_move* emphasize the aggressive strategy to actively block the opponents, we can also add defensive version by adding more weights on the possible self-moves. It effective goes against the simple *open_move* strategy.

3. The final custom function is a linear combination of both strategies. Note that the weight may not need to be fixed. Allowing adjustable weight make it possible to *learn* an ensemble of value functions, similar to what used in the AlphaGo model. It provides a general strategy that not only fit this game but allows for further varities of value functions to be used adaptively.