Heuristic Analysis

Frederick Chyan

Friday, July 21, 2017

The purpose of this project is to find a heuristic that consistently outperforms AB_Improved, which is player's available moves minus opponent's available moves. The heuristic design iteration begins with selecting three custom heuristic functions. The first heuristic penalizes number of opponent's available moves. The second heuristic, rewards number of player's available moves. The third heuristic adds the center score to first heuristic, which is the Euclidean distance between player's position and the center of the board. Followings are the three preliminary heuristic.

```
h1 = own_moves - 2 * opp_moves
h2 = 2 * own_moves - opp_moves
h3 = own_center_score + (own_moves - 2 * opp_moves)
```

Below are the result.

```
                    *************************
                         Playing Matches
                    *************************

 Match #   Opponent    AB_Improved     AB_h1        AB_h2        AB_h3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       8  |   2     6  |   4     6  |   4    10  |   0
    2       MM_Open      5  |   5     6  |   4     5  |   5     6  |   4
    3      MM_Center     9  |   1     8  |   2     6  |   4     8  |   2
    4     MM_Improved    6  |   4     6  |   4     6  |   4     7  |   3
    5       AB_Open      6  |   4     4  |   6     4  |   6     5  |   5
    6      AB_Center     6  |   4     4  |   6     7  |   3     6  |   4
    7     AB_Improved    4  |   6     4  |   6     4  |   6     4  |   6
--------------------------------------------------------------------------
          Win Rate:      62.9%        54.3%        54.3%        65.7%
```

First, notice that AB_h3 seem to perform better than the other two heuristic. However, it still does not outperform AB_Improved. In fact, all three preliminary heuristics score the same against AB_Imrpoved. This result, however, suggest one should further improve upon AB_h3. Since h1 and h2 performs about the same, one wonder whether lowering the penalty to opponent's moves might make any difference as to not amplify prediction error.

```
h4 = own_center_score + (own_moves - 1.414 * opp_moves)
                        ************************
                             Playing Matches
                        ************************

 Match #   Opponent       AB_h4
                        Won | Lost
    1     AB_Improved   21  |  19
-------------------------------------------------------------------------
           Win Rate:      52.5%
```

This is better, however, it's only slightly above 50%. Looking at center score, it was arbitrarily added to the heuristic. There are some problems with this approach. First, it's a different measurement. Second, it does not even have a negative value like own_moves - opp_moves does. Is there a way to map both of them to a same scale and aggregate them? Well it's too much trouble to linearize both measurements at every search level, so it's better to look for an easier way. Here a new scaling factor called *distance boost* is introduced. It amplifies player's own moves by a factor proportional to the ratio of player's center score against opponent's center score. Intuitively, it *encourages movement that will place player further from the center than the opponent is*.

```
distance_boost = own_center_score / (opp_center_score + own_center_score)
h5 = float((1+distance_boost) * own_moves - 1.414 * opp_moves)

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

 Match #   Opponent    AB_Custom_3
                        Won | Lost
    1     AB_Improved   25  |  15
-------------------------------------------------------------------------
           Win Rate:      62.5%
```

The result indeed looks promising, play the game 800 times to ensure this heuristic indeed outperforms AB_Improved consistently. The trials are parallelized by playing 20*2=40 matches on 4 processor cores 5 times.

```
                    *************************
                         Playing Matches
                    *************************

 Match #   Opponent     AB_Custom    AB_Custom    AB_Custom    AB_Custom
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1     AB_Improved   26  |  14    31  |   9    30  |  10    29  |  11
----------------------------------------------------------------------------
           Win Rate:      65.0%        77.5%        75.0%        72.5%


 Match #   Opponent     AB_Custom    AB_Custom    AB_Custom    AB_Custom
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1     AB_Improved   25  |  15    23  |  17    26  |  14    23  |  17
----------------------------------------------------------------------------
           Win Rate:      62.5%        57.5%        65.0%        57.5%


 Match #   Opponent     AB_Custom    AB_Custom    AB_Custom    AB_Custom
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1     AB_Improved   29  |  11    23  |  17    25  |  15    29  |  11
----------------------------------------------------------------------------
           Win Rate:      72.5%        57.5%        62.5%        72.5%


 Match #   Opponent     AB_Custom    AB_Custom    AB_Custom    AB_Custom
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1     AB_Improved   29  |  11    27  |  13    25  |  15    26  |  14
----------------------------------------------------------------------------
           Win Rate:      72.5%        67.5%        62.5%        65.0%


 Match #   Opponent     AB_Custom    AB_Custom    AB_Custom    AB_Custom
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1     AB_Improved   26  |  14    24  |  16    29  |  11    26  |  14
----------------------------------------------------------------------------
           Win Rate:      65.0%        60.0%        72.5%        65.0%
```

Here almost all matches are above 60%. Since the outcome is either win or loss, the result follows a binomial distribution (win with probability p). To test whether the agent can beat AB_Improved with 60% certainty. Use a one-tailed test with null and alternative hypotheses.

```
H0: p ≤ 0.6
H1: p > 0.6
```

The total number of winning trials is calculated by summing all won counters, which is 531. Then calculate the cumulative probability of winning at most 530 times, testing at threshold α = 0.05. The p-value is 0.000116 which is less than α, the null hypothesis can be rejected.

```
26+31+30+29+25+23+26+23+29+23+25+29+29+27+25+26+26+24+29+26 = 531
p-value = 1—BINOM.DIST(530, 800, 0.6, TRUE) = 0.000116736 < .05 = α
```

so the design iteration concludes with 95% confidence that the new heuristic shows a significant improvement over AB_Improved.

Summary

The best evaulation function is `float((1+distance_boost) * own_moves - 1.414 * opp_moves)`. This is supported by three facts. Firstly, it beats AB_Improve consistenly with at least 60% win rate (p-value < 0.05). Secondly, it is tuned from a prilimiary function that has very high win rate against random moves while the other two heuristics only won about half of the time aginst random, meaning the selected heuristic is guarded against random moves. Lastly, the first two other preliminary heuristics had lower win rates.