Heuristic Analysis

Jason McDonald

For this project, I tried a variety of heuristics and combinations thereof. In the end, I was only able to attain mild improvements over the simple metrics provided in sample players with no standouts. All custom score functions played slightly different and performed similarly with 70% for the test agents I choose which is slightly better than 'AB_Improved' which average 67% for the same tests.

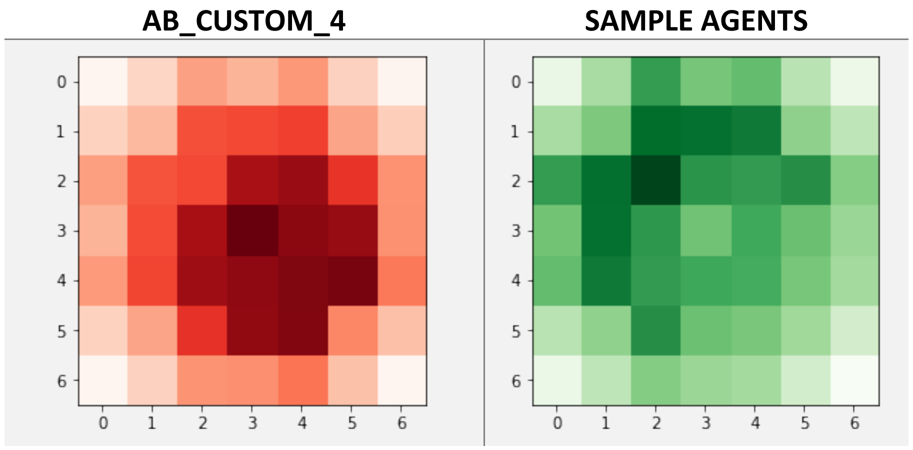
```
                                                 Tournament results

   Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3  AB_Custom_4  AB_Custom_5  AB_Custom_6

                            Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost

       1       Random      10  |   0     9  |   1    10  |   0    10  |   0    10  |   0    10  |   0    10  |   0

       2       MM_Open      7  |   3     8  |   2     9  |   1     9  |   1     8  |   2     9  |   1     9  |   1

       3      MM_Center    10  |   0    10  |   0     8  |   2    10  |   0    10  |   0     8  |   2    10  |   0

       4     MM_Improved    8  |   2     9  |   1     8  |   2     7  |   3     6  |   4     8  |   2     7  |   3

       5       AB_Open      6  |   4     5  |   5     7  |   3     5  |   5     3  |   7     5  |   5     3  |   7

       6      AB_Center     5  |   5     3  |   7     7  |   3     4  |   6     7  |   3     5  |   5     6  |   4

       7     AB_Improved    6  |   4     2  |   8     5  |   5     7  |   3     4  |   6     4  |   6     5  |   5
    --------------------------------------------------------------------------------------------------------------

              Win Rate:      74.3%        65.7%        77.1%        74.3%        68.6%        70.0%        71.4%
```

Win Rate after playing 100 games against each opponent except 'Random' for a total of 600 games per custom score						
AB_Custom	AB_Custom_2	AB_Custom_3	AB_Custom_4	AB_Custom_5	AB_Custom_6	AB_Improved
68%	70%	71%	69%	69%	69%	67%

```python
"""
Move Options:

'Move Options' attempts to return a more robust version of "Number of My Moves"
by returning the total number of spaces accessible up to 'depth' moves in the future.
this increased the search time per call to the heuristic and over all only provides
minor improvements with depth beyond 1 and diminishing returns after 2 levels
diminishing returns after 2 levels is due to compute time and the possibility of
poor board states in-between the current ply and forecasted ply. This heuristic is
most useful near end game when the game tree thins out and restricted movement
quickly leads to a loss.
"""
# Move Options
def heuristic_1(game, player, depth):
    # Moves avalible to Knight
    directions = [(-2, -1), (-2, 1), (-1, -2), (-1, 2),
                  (1, -2), (1, 2), (2, -1), (2, 1)]
    blanks = game.get_blank_spaces() # blank space on the board
    own_moves = set(game.get_legal_moves(player))
    own_moves -= set(game.get_legal_moves(game.get_opponent(player))) # My moves - Opp moves

    # Move Knight up to 'depth' moves away from initial position
    for _ in range(depth - 1):
        new_moves = []
        # Find all spaces avalible from current depth
        for r, c in own_moves:
            new_moves.extend([(r + dr, c + dc) for dr, dc in directions
                if (r + dr, c + dc) in blanks])
        #if not len(new_moves):
        #    break
        own_moves = set(new_moves)
    # Return a score that is total number of spaces that can be reached in 'depth' moves
    return float(len(own_moves))
```

```
Center Score:

Returns the vertical + horizontal distance from the board center for the given player.
This is faster to compute than the square distance while still preserving the ordering
of scores computed. Since the Minimax algorithm is only concerned with Min or Max
the ordering of scores is all that matters.

Generally this is used to preferentially keep the player near the center as trapping
often occurs at the corners especially late in the game
"""
def heuristic_2(game, player):
    w, h = (game.width-1) / 2., (game.height-1) / 2.
    y, x = game.get_player_location(player)
    return float(abs(h - y) + abs(w - x))
```

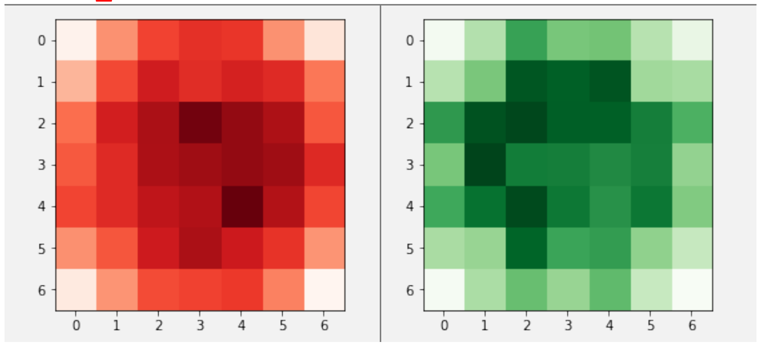
Originally this heuristic was calculated using 'w', 'h' found in the 'center_score()' function of 'sample_players.py'. However after performing an average over ***30,000*** *games between 'custom_score_4' and numerous opponents an asymmetry was found (bellow) even though all score functions were expected to be topologically symmetric. The noticeable offset in this heuristic and 'sample_players.py' were being caused by a bias. (3.5, 3.5) is the lower right hand corner of (3,3) not the center. As can be seen, AB_Custom_4 which relies on favoring the center positon has an off center distribution. As Sample Agents all include a center score player from 'sample_players.py' all distribution are asymmetrically biases for each score evaluation.*



```
Improved Score:

This simply returns the number of player's moves
minus the number of player's opponent moves. Generally
this is a good metric to use as a score as it encourages
move options for the player while minimizing those for
the opponent
"""
def heuristic_3(game, player):
    own_moves = game.get_legal_moves(player)
    own_len = float(len(own_moves))
    opp_moves = game.get_legal_moves(game.get_opponent(player))
    opp_len = float(len(opp_moves))
    return (own_len - opp_len)



"""
Coverage:

This is a simple metric that determines how much of
the board is occupied  by unusable space and is generally
useful for scaling up or down score contributions based
on the how far the game has progressed.
"""
def heuristic_4(game):
    return 1. - len(game.get_blank_spaces()) / (game.width * game.height)
```

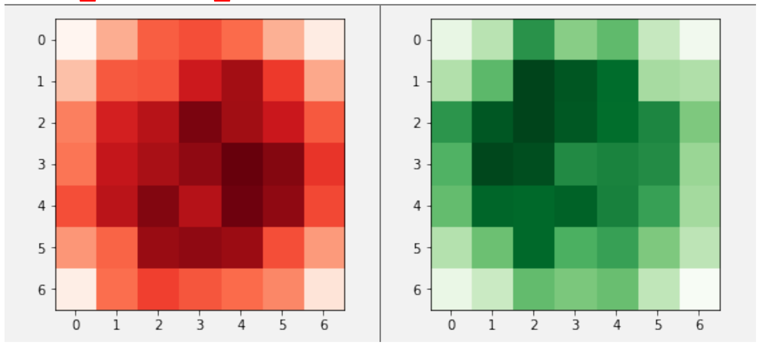
```
My moves - Opp moves - Distance from center:

This custom score balances heuristic of maximizing the players moves and
minimizing the opponents moves with the heuristic of avoiding the corners
of the board.
"""
def custom_score(game, player):
    if game.is_loser(player):
        return float("-inf")
    # Best outcome
    if game.is_winner(player):
        return float("inf")

    # Tuning parameter
    p0 = 0.5 # p0 = parameters['p0_1']
    # Improved Score
    score = heuristic_3(game, player) # My moves - Opp moves
    # Center Score
    own_dist = heuristic_2(game, player) # My distance from board center
    score -= p0*(own_dist)
    return score
```

AB_CUSTOM vs SAMPLE AGENTS for 600 GAMES



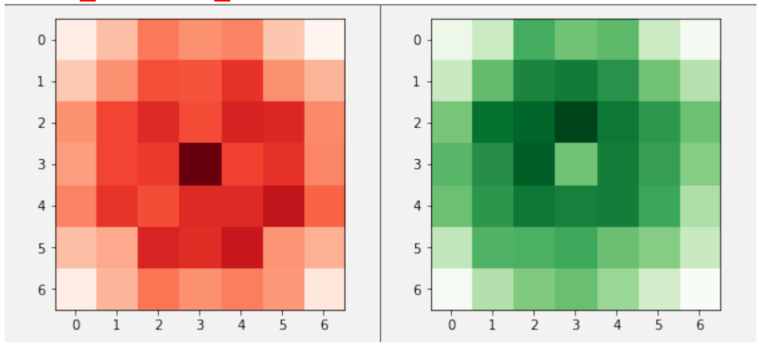
```python
Avoid the Center and Keep my Options Open:

This custom score avoids the corners of the board until move options become limited (< 3)
at which point it switches over to a heuristic that favors game states with most moves two
moves out from the configuration being evaluated
"""
def custom_score_2(game, player):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")
    own_moves = game.get_legal_moves(player)
    opp_moves = game.get_legal_moves(game.get_opponent(player))
    own_moves = [move for move in own_moves if move not in opp_moves] # My moves excluding opp moves
    if len(own_moves) >= 3:
        return  - heuristic_2(game, player) # Distance from center
    else:
        score = heuristic_1(game, player, 2) # Number of moves available two moves out
        if score >= 4:
            return score - 100 # Make sure this is considered a bad score if any board
                               # state is found with more than 2 moves
        else:
            return float("-inf") # assist pruning
```

AB_CUSTOM_2 vs **SAMPLE AGENTS** for **600 GAMES**



```
Opp Distance - Own Distance:

This custom score encourages moves that keep the player near the center
and the opponent player near the corners where they are more likely to
be trapped.
"""
def custom_score_3(game, player):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    own_moves = game.get_legal_moves(player)
    opp_moves = game.get_legal_moves(game.get_opponent(player))
    # Center Scores
    own_dist = heuristic_2(game, player)
    opp_dist = heuristic_2(game, game.get_opponent(player))
    len_moves = len([move for move in own_moves if move not in opp_moves])
    if len_moves >= 3:
        return opp_dist - own_dist # Opponent board center distance minus Own board center distance
    elif len_moves == 2:
        return heuristic_1(game, player, 2) - 100 # if less than two moves available start looking
                                                  # for ways out
    else:
        return float("-inf") # assist pruning
```

AB_CUSTOM_3 vs **SAMPLE AGENTS** for **600 GAMES**

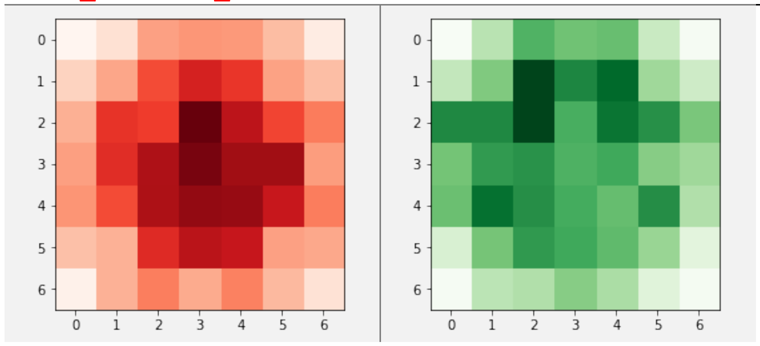


Occupy the center:

This custom board does something like minus board center distance for both player and opponent. It favors occupying the middle of the board by favoring game states with the blank spaces near the edges

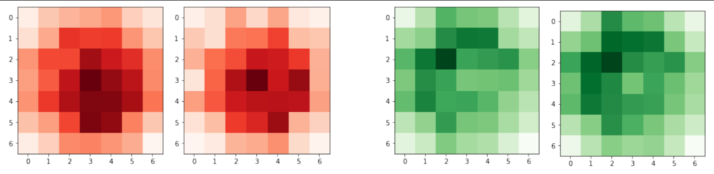
```
def custom_score_4(game, player):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")
    own_moves = game.get_legal_moves(player)
    opp_moves = game.get_legal_moves(game.get_opponent(player))
    len_moves = len([move for move in own_moves if move not in opp_moves])
    if len_moves >= 3: # if more than 2 moves available
        score = 0.
        w, h = (game.width-1) / 2., (game.height-1) / 2.
        for y, x in game.get_blank_spaces(): # for each blank space add its vertical and horizontal
                                             # distance from the center to the score
            score += (abs(h - y)  + abs(w - x))
        return  float(score)
    elif len_moves == 2:
        return heuristic_1(game, player, 3) - 200 # search for escapes near end game
    else:
        return float("-inf") # assist pruning
```

AB_CUSTOM_4 vs SAMPLE AGENTS for 600 GAMES



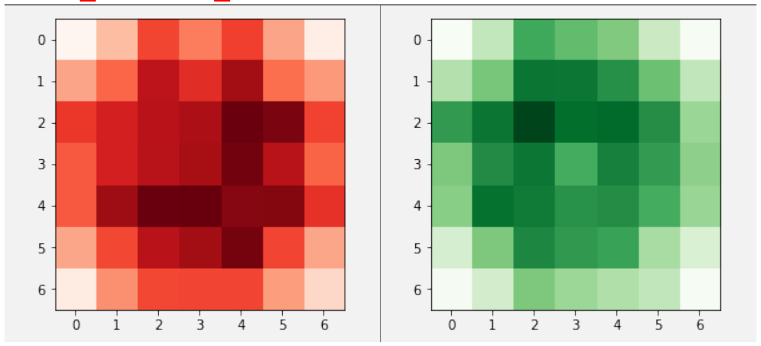
AB_Custom_4

(AB_Custom_4 was run twice for 600 games to show variation)



```
Find a way out:

This custom score simply favors more move options and near the end
of the game, when the game tree thins out, it looks further and
further out to find paths with the most available outs. Compute
time per call to the score function increases near end game but not
exponentially as less moves become available near end game.

def custom_score_5(game, player):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    if len(game.get_legal_moves(player)) >= 2:
        coverage = heuristic_4(game)
        # look further ahead for moves as game play progresses
        return heuristic_1(game, player, int((3*coverage)**2 + 1))
    else:
        return float("-inf")
```

AB_CUSTOM_5 vs SAMPLE AGENTS for 600 GAMES

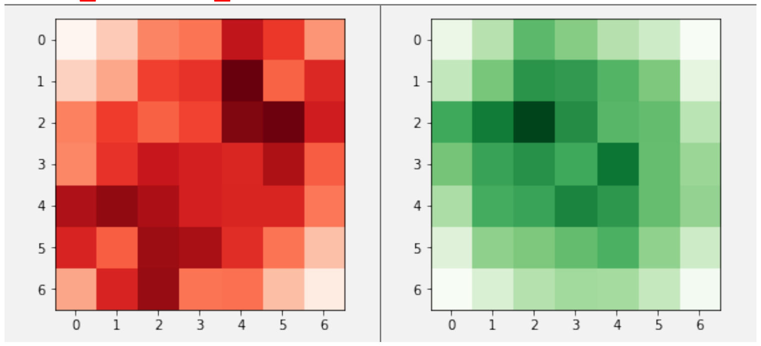


Occupy the Diagonal:

This custom score is similar to 'Occupy the center'. However instead of preferring a board that occupies the center in prefers a board with a filled diagonal from (0,0) to (7,7). This still largely avoids the trapping of corners near end game and has the potential to corral the opponent. The heat map for this function clearly shows the preference for the diagonal.

```
def custom_score_6(game, player):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")
    if len(game.get_legal_moves(player)) >= 2:
        # score increases for each blank space roughly proportional
        # to its distance from the center
        return float(-sum([abs(y - x) for y, x in game.get_blank_spaces()]))
    else:
        return float("-inf") # assist pruning
```

AB_CUSTOM_6 vs SAMPLE AGENTS for 600 GAMES



Example games against “AB_Improved” for each custom score.

“AB_Improved”: player 2 marker ‘-‘ | “Custom_Score*”: player 1 marker ‘*’

Vs AB_Improved #2	Custom_Score	Custom_Score_2	Custom_Score_3
Win Rate (500 games)	51%	55%	56%

Move #10

Custom_Score

```
Move: 10  to  [4, 2] Score: -1.0
     0   1   2   3   4   5   6
0  |   |   |   |   |   |   |   |
1  |   |   |   |   |   |   |   |
2  |   |   |   |   | 2 |   |   |
3  |   |   | * |   |   |   |   |
4  |   | * | 1 | - | - |   |   |
5  | * | - |   | * |   |   |   |
6  |   |   | * | - |   |   |   |
```

Custom_Score_2

```
Move: 10  to  [5, 6] Score: -94
     0   1   2   3   4   5   6
0  |   |   |   |   |   |   |   |
1  |   |   |   | 2 |   |   |   |
2  |   |   |   |   |   |   | * |
3  |   | - | - |   | * |   |   |
4  |   |   |   | - | * |   |   |
5  |   | - |   |   |   | * | 1 |
6  |   |   |   | * |   |   |   |
```

Custom_Score_3

```
Move: 10  to  [1, 2] Score: 1.0
     0   1   2   3   4   5   6
0  |   | * | 2 |   |   |   |   |
1  |   |   | 1 | * | - |   |   |
2  | * | * |   |   |   |   |   |
3  |   |   |   | - |   |   |   |
4  | * |   |   |   |   |   |   |
5  |   |   |   |   | - |   |   |
6  |   |   |   |   |   |   | - |
```

Move #20

Custom_Score

```
Move: 20  to  [2, 3] Score: 0.0
     0   1   2   3   4   5   6
0  |   |   |   |   |   | - |   |
1  | * |   |   |   | - |   |   |
2  |   |   | * | 1 | - | 2 | - |
3  | * | * | * | - |   |   |   |
4  |   | * | * | - | - |   |   |
5  | * | - |   | * |   |   |   |
6  |   |   | * | - |   |   |   |
```

Custom_Score_2

```
Move: 20  to  [3, 3] Score: -1.0
     0   1   2   3   4   5   6
0  |   | - |   | 2 |   |   |   |
1  |   | - |   | - |   |   |   |
2  |   |   | - |   |   |   | * |
3  | - | - | - | 1 | * | * |   |
4  |   |   |   | - | * | * |   |
5  |   | - |   |   | * | * | * |
6  |   |   |   | * |   |   | * |
```

Custom_Score_3

```
Move: 20  to  [4, 4] Score: 3.0
     0   1   2   3   4   5   6
0  |   | * | - |   |   |   |   |
1  | - |   | * | * | - | * |   |
2  | * | * | - | * |   |   |   |
3  |   | * |   | - | - |   | * |
4  | * |   |   |   | 1 |   | - |
5  |   |   |   |   | - |   |   |
6  |   |   |   |   |   | 2 | - |
```

Move #30

Custom_Score

```
Move: 30  to  [0, 0] Score: -3.5
     0   1   2   3   4   5   6
0  | 1 |   |   |   | * | - |   |
1  | * |   | * | - | - |   | * |
2  |   |   | * | * | - | - | - |
3  | * | * | * | - | - | * |   |
4  |   | * | * | - | - |   | - |
5  | * | - |   | * | - |   |   |
6  |   |   | * | - |   |   | 2 |
```

Custom_Score_2

```
Move: 30  to  [2, 5] Score: -inf
     0   1   2   3   4   5   6
0  |   | - | - | - |   |   |   |
1  |   | - |   | - |   | - |   |
2  |   | - | - | - |   | 1 | * |
3  | - | - | - | * | * | * |   |
4  | 2 | * |   | - | * | * | * |
5  |   | - |   | * | * | * | * |
6  |   |   |   | * |   | * | * |
```

Custom_Score_3

```
Move: 30  to  [3, 0] Score: -inf
     0   1   2   3   4   5   6
0  |   | * | - |   |   |   |   |
1  | - |   | * | * | - | * |   |
2  | * | * | - | * |   |   |   |
3  | 1 | * |   | - | - |   | * |
4  | * | - | 2 | * | * |   | - |
5  | - | * | * | - | - |   |   |
6  |   |   | - |   | * | - | - |
```

End Game

Custom_Score

```
Move: 36  to  [6, 1] Score: inf
     0   1   2   3   4   5   6
0  | * |   |   |   | * | - |   |
1  | * |   | * | - | - |   | * |
2  |   | * | * | * | - | - | - |
3  | * | * | * | - | - | * |   |
4  | * | * | * | - | - | - | - |
5  | * | - |   | * | - |   | 2 |
6  |   | 1 | * | - | - |   | - |
```

Custom_Score_2

```
Move: 38  to  [0, 5] Score: inf
     0   1   2   3   4   5   6
0  |   | - | - | - | * | 1 |   |
1  |   | - |   | - |   | - | * |
2  |   | - | - | - | * | * | * |
3  | - | - | - | * | * | * |   |
4  | - | * | - | - | * | * | * |
5  | - | - |   | * | * | * | * |
6  |   | - | 2 | * |   | * | * |
```

Custom_Score_3

```
Move: 34  to  [0, 3] Score: inf
     0   1   2   3   4   5   6
0  |   | * | - | 1 |   |   |   |
1  | - | * | * | * | - | * |   |
2  | * | * | - | * |   |   |   |
3  | * | * |   | - | - |   | * |
4  | * | - | - | * | * |   | - |
5  | - | * | * | - | - | 2 |   |
6  |   |   | - | - | * | - | - |
```

	Custom_Score_4	Custom_Score_5	Custom_Score_6
Win Rate (500 games)	56%	49%	52%

Move #10

```
Move: 10  to  [5, 6] Score: -inf
      0   1   2   3   4   5   6
0   |   |   |   |   |   |   |   |
1   |   |   |   |   |   |   |   |
2   |   |   |   | * | * |   |   |
3   |   |   |   |   |   | * |   |
4   |   |   | - | * | * |   |   |
5   |   |   |   |   | - | 2 | 1 |
6   |   |   | - | - |   |   |   |
```

```
Move: 10  to  [0, 1] Score: 2
      0   1   2   3   4   5   6
0   |   | 1 | * |   |   |   |   |
1   | * |   |   |   |   | * |   |
2   |   |   | * | * |   |   |   |
3   |   |   |   |   | - |   |   |
4   |   |   | - |   |   |   | - |
5   |   | 2 |   |   |   |   |   |
6   |   |   |   | - |   |   |   |
```

```
Move: 10  to  [4, 5] Score: -87
      0   1   2   3   4   5   6
0   |   |   | - |   | - |   |   |
1   |   |   |   |   |   |   |   |
2   |   | - |   | - |   |   |   |
3   |   |   |   | * |   |   |   |
4   | * | * | 2 |   |   | 1 |   |
5   |   |   |   | * |   |   |   |
6   |   | * |   |   |   |   |   |
```

Move #20

```
Move: 20  to  [0, 2] Score: -189
      0   1   2   3   4   5   6
0   |   |   | 1 |   |   |   |   |
1   |   |   | 2 |   | * |   |   |
2   | - |   |   | * | * |   | * |
3   |   |   | - |   | - | * |   |
4   |   |   | - | * | * | * |   |
5   |   |   |   | - | - | - | * |
6   |   |   | - | - | * |   |   |
```

```
Move: 20  to  [6, 4] Score: 6
      0   1   2   3   4   5   6
0   |   | * | * |   |   | * |   |
1   | * |   | - | * |   | * |   |
2   | - |   | * | * |   |   | * |
3   |   |   | - | - | - |   |   |
4   |   |   | - |   |   | * | - |
5   |   | - | 2 |   |   |   |   |
6   |   |   |   | - | 1 |   |   |
```

```
Move: 20  to  [2, 4] Score: -70
      0   1   2   3   4   5   6
0   |   | 2 | - |   | - |   |   |
1   |   |   | * |   | - |   |   |
2   |   | - | - | - | 1 |   | - |
3   |   | * |   | * | - |   |   |
4   | * | * | - | * |   | * |   |
5   |   |   |   | * |   |   |   |
6   |   | * |   |   | * |   |   |
```

Move #30

```
Move: 30  to  [3, 0] Score: -198
      0   1   2   3   4   5   6
0   |   | * | * |   | - |   |   |
1   |   |   | - | * | * |   |   |
2   | - | * | * | * | * | - | * |
3   | 1 | 2 | - | - | - | * |   |
4   |   |   | - | * | * | * |   |
5   |   |   | - | - | - | - | * |
6   |   |   | - | - | * |   |   |
```

```
Move: 30  to  [2, 5] Score: -inf
      0   1   2   3   4   5   6
0   |   | * | * |   |   | * | * |
1   | * |   | - | * | * | * |   |
2   | - |   | * | * |   | 1 | * |
3   |   |   | - | - | - | * |   |
4   | 2 |   | - | * | - | * | - |
5   |   | - | - | - |   |   |   |
6   |   | - |   | - | * | - |   |
```

```
Move: 30  to  [0, 3] Score: -inf
      0   1   2   3   4   5   6
0   |   | - | - | 1 | - |   |   |
1   |   | * | * | - | - |   |   |
2   |   | - | - | - | * | - | - |
3   | * | * | * | * | - |   |   |
4   | * | * | - | * | - | * | 2 |
5   |   | * |   | * |   |   |   |
6   |   | * |   |   | * | - |   |
```

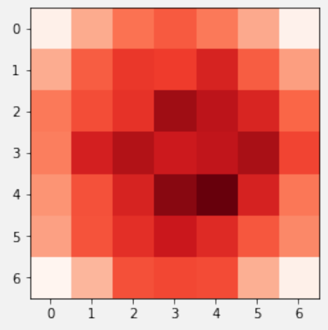
End Game

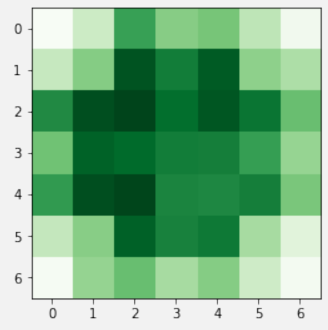
```
Move: 32  to  [5, 1] Score: inf
      0   1   2   3   4   5   6
0   |   | * | * |   | - |   |   |
1   |   |   | - | * | * |   |   |
2   | - | * | * | * | * | - | * |
3   | * | - | - | - | - | * |   |
4   |   |   | - | * | * | * |   |
5   | 2 | 1 | - | - | - | - | * |
6   |   |   | - | - | * |   |   |
```

```
Move: 34  to  [1, 6] Score: inf
      0   1   2   3   4   5   6
0   | 2 | * | * |   | * | * | * |
1   | * |   | - | * | * | * | 1 |
2   | - | - | * | * |   | * | * |
3   |   |   | - | - | - | * |   |
4   | - |   | - | * | - | * | - |
5   |   | - | - | - |   |   |   |
6   |   | - |   | - | * | - |   |
```

```
Move: 36  to  [5, 5] Score: inf
      0   1   2   3   4   5   6
0   |   | - | - | * | - |   |   |
1   |   | * | * | - | - | * |   |
2   |   | - | - | - | * | - | - |
3   | * | * | * | * | - |   | * |
4   | * | * | - | * | - | * | - |
5   | 2 | * |   | * | - | 1 |   |
6   |   | * | - |   | * | - |   |
```

AB_IMPROVED vs SAMPLE AGENTS for 640 GAMES





CUMULATIVE OCCUPATION SCORES

```
[[  89.  170.  223.  243.  216.  171.   86.]
 [ 169.  241.  272.  268.  293.  240.  182.]
 [ 216.  253.  277.  346.  319.  291.  233.]
 [ 211.  297.  330.  304.  315.  338.  261.]
 [ 191.  250.  293.  360.  381.  295.  218.]
 [ 179.  249.  280.  306.  285.  245.  202.]
 [  80.  159.  251.  258.  255.  166.   89.]]
```

AB_Improved

```
[[  78.  140.  258.  199.  211.  154.   89.]
 [ 146.  200.  336.  297.  329.  194.  168.]
 [ 283.  340.  349.  313.  334.  305.  220.]
 [ 216.  323.  316.  297.  295.  261.  188.]
 [ 265.  340.  347.  288.  285.  294.  209.]
 [ 148.  198.  324.  291.  301.  174.  117.]
 [  81.  189.  221.  174.  200.  138.   85.]]
```

Sample Agents