
Heuristic Analysis

Chansung, Park - 22 November 2017

Custom Evaluation Function

Former evaluation function measures superiority of a current player in each turn by calculating the difference between the number of moves that a current player could legally can make and the number of moves from the opponent player. It is quiet reasonable since the method represents more opportunities that a current player has.

However, the former method doesn't consider exceptional cases. For instance, when a current player starts placing on the center of the board, it has the maximum number of moves. On the other hand, on the next turn, the opponent wouldn't consider placing on the edge sides because considerable amount of moves couldn't be placeable. The opponent would rather tries to place near the previous player's site.

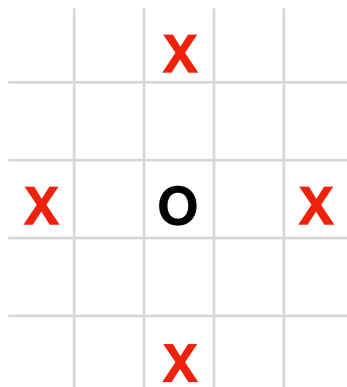


Fig1. Edge Moves

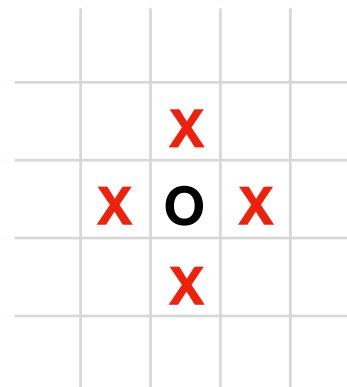


Fig2. Near Moves

Fig1. shows the maximum moves of four can be achieved based on the edges, and Fig2. depicts that there are two more opportunities of move achievable on the near sites at maximum. As a result, the opponent player, depicted as 'X', would choose the near moves over edge moves. However, on the second move, the situation get reversed like players started on the edges will get more chances of moves. It doesn't seem like the measurement of the number of moves has big influence since it can be fluctuated over turns.

	6	6	2	
7		3	5	
7	5	1	1	3
	2	4	4	
	8			

Fig3. Start from near moves

		2	2	5
8	8	4		1
6	3	1	6	2
	7	7	4	
	5			

Fig4. Start from Edge Moves

Fig3. and Fig4. depict situation about the start from the near and from the edge sites respectively. Even though the red color in the Fig3. had more opportunity than other red one in the Fig4., it lost the game. Especially, the red player in the Fig4. tried a special attempt to win the game.

	X		X	
X				X
		O		
X				X
	X		X	

Fig5. Blocking the Opponent's Moves

4		2	2	
		4		1
	3	1		2

Fig6. Exceptional Case

Fig5. Shows a brand new strategy to win the game. A player tries to block the opponent's next moves. This strategy expects to reduce the opponent's next possible moves and optimally maximize its next moves. However, there is an exceptional case as described in Fig6. The red player moved to block its opponent on the fourth move, but it failed because it is silly over the opponent. It needed to think about its next moves as well. In order to do that, the red player would choose the next move as in Fig4.. Like so, the red player could block the opponent along with a guarantee it still has at least an additional move to continue. Fig7. is the logical code which just does what I described above.

```

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

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

    my_moves = game.get_legal_moves(player)
    my_pos = game.get_player_location(player)

    opponent = game.get_opponent(player)
    opponent_moves = game.get_legal_moves(opponent)

    base_score = float(len(my_moves) - len(opponent_moves))

    if my_pos in opponent_moves and len(my_moves) > 1:
        base_score = base_score + 2
        base_score *= len(my_moves) * 1.5

    return base_score

```

Fig7. Evaluation Function Code in Python

***** Playing Matches *****												
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3				
		Won	Lost	Won	Lost	Won	Lost	Won	Lost			
1	Random	8	2	38	8	8	2	8	2			
2	MM_Open	10	0	7	3	9	1	0	0			
3	MM_Center	7	3	38	8	7	3	9	1			
4	MM_Improved	6	4	8	2	4	6	7	3			
5	AB_Open	0	4	4	0	0	0	4	0			
6	AB_Center	6	4	6	4	6	4	6	4			
7	AB_Improved	3	7	6	4	6	4	6	4			
Win Rate:		44.2%		71.4%		54.3%		64.3%				

***** Playing Matches *****												
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3				
		Won	Lost	Won	Lost	Won	Lost	Won	Lost			
1	Random	18	8	10	8	9	1	10	8			
2	MM_Open	8	2	7	3	6	4	6	4			
3	MM_Center	9	1	9	1	8	2	9	1			
4	MM_Improved	9	1	7	3	5	5	4	6			
5	AB_Open	6	4	4	6	4	6	6	4			
6	AB_Center	5	5	7	3	3	7	5	5			
7	AB_Improved	4	4	4	4	4	4	6	6			
Win Rate:		72.9%		71.4%		58.6%		64.3%				

***** Playing Matches *****												
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3				
		Won	Lost	Won	Lost	Won	Lost	Won	Lost			
1	Random	9	1	38	8	8	2	9	1			
2	MM_Open	8	2	8	2	4	6	7	3			
3	MM_Center	10	0	7	3	8	2	7	3			
4	MM_Improved	7	3	8	2	4	6	4	4			
5	AB_Open	5	5	5	5	4	6	5	5			
6	AB_Center	5	5	7	3	6	4	6	7			
7	AB_Improved	5	5	5	5	5	5	6	4			
Win Rate:		74.8%		71.4%		58.6%		61.4%				

***** Playing Matches *****												
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3				
		Won	Lost	Won	Lost	Won	Lost	Won	Lost			
1	Random	9	1	10	8	10	8	10	8			
2	MM_Open	7	3	8	2	7	3	8	2			
3	MM_Center	8	2	9	1	7	3	9	1			
4	MM_Improved	8	2	7	3	5	5	8	2			
5	AB_Open	5	5	6	4	4	6	5	5			
6	AB_Center	6	4	7	3	3	7	7	3			
7	AB_Improved	6	4	7	3	3	7	6	4			
Win Rate:		68.6%		71.1%		55.7%		75.7%				

Fig8. Testments

Fig8. is the result from the testament comparing between the previous evaluation function(AB_Improved) and the new evaluation function proposed in this paper(AB_Custom). It shows a better performance in the three cases and slightly less performance in one case.

In Fig8., the AB_Custom_2 function uses a method of giving weight only when current player has a chance to block the opponent's future move. The AB_Custom_3 function uses the same method to the AB_Custom_2 except using the formula of the number of currently player's future moves minus the number of the opposite player's future moves. The proposed evaluation function in this paper(AB_Custom) adds one more feature to the AB_Custom_3 by checking if current player could have extra moves after blocking the opponent's future moves. As seen in Fig8., AB_Custom performed much better than AB_Custom_2 and AB_Custom_3.

Abbreviations

MM: Mini-Max

AB: Alpha-Beta