

Heuristic analysis

Udacity - Artificial Intelligence Nanodegree

- custom_score 1

In this function, I calculated the distance from center. Because I thought there are a lot of case in the center. So, I thought it could make more efficient move. And then, added basic score: *own_moves_numver* - *opp_moves_number*. And, when a player lost the game, the agent will get negative reward ($-\infty$). Contrary, when a player win the game, the agent will get positive reward(∞).

- custom_score2

This function is very similar with 'AB_improved'. I added *blank_spaces*. The Blank spaces means the locations that are still available on the board. So I thought it can helpful for our score. Except this, all of things are same as AB_improved.

- custom_score 3

In this case, I calculated same spaces in legal moves between own player and opponent player. I thought occupying same legal space first can make reduced opponent's legal moves number. So I thought it can be efficient measure.

- Conclusion

***** Playing Matches *****									
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	15	5	16	4	18	2	16	4
2	MM_Open	12	8	12	9	14	6	10	10
3	MM_Center	15	5	13	7	13	7	15	5
4	MM_Improved	13	7	12	8	11	9	14	6
5	AB_Open	11	9	12	9	12	9	9	11
6	AB_Center	10	10	12	8	11	9	10	10
7	AB_Improved	10	10	12	8	10	10	12	8
Win Rate:		61.4%		62.1%		62.9%		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	14	6	18	2	18	2	16	4
2	MM_Open	11	9	13	7	16	4	13	7
3	MM_Center	16	4	11	9	17	3	14	6
4	MM_Improved	11	9	13	7	14	6	10	10
5	AB_Open	8	12	11	9	11	9	11	9
6	AB_Center	9	11	9	11	12	8	13	7
7	AB_Improved	10	10	10	10	8	12	10	10
Win Rate:		56.4%		60.7%		68.6%		62.1%	
***** Playing Matches *****									
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	19	1	17	3	17	3	17	3
2	MM_Open	14	6	14	6	15	5	13	7
3	MM_Center	14	6	13	7	13	7	16	4
4	MM_Improved	11	9	12	8	14	6	13	7
5	AB_Open	9	11	12	8	13	7	12	8
6	AB_Center	11	9	11	9	12	8	9	11
7	AB_Improved	13	7	9	11	9	11	9	11
Win Rate:		65.0%		62.9%		66.4%		63.6%	

I set number of matches in tournament.py as 10. And the following table is that summarized result.

	AB_Improved	distance from center	blank spaces	similar moves
Run 1	61.4%	62.1%	62.9%	61.4%
Run 2	56.4%	60.7%	68.6%	62.1%
Run 3	65.0%	62.9%	66.4%	63.6%
Average	60.9%	61.9%	66.0%	62.4%

In random algorithm, all of the scores are quite well. But in Minimax or Alpha-beta pruning, little different. Here is the result except random situation.

	AB_Improved	distance from center	blank spaces	similar moves
Run 1	59.2%	59.2%	58.3%	58.3%
Run 2	54.1%	55.8%	65.0%	59.2%
Run 3	60.0%	59.2%	63.3%	60.0%
Average	57.8%	58.1%	62.2%	59.2%

Actually I thought score 3 is better than others. So, I understood there is not perfect solution. It depended on situations. So, I choose custom_score 2: added black spaces. Because

1. Only custom_score 2 has over 60.0% average. Sometimes it had poor score than others. But, generally it was best.
2. This score is easy to understand and implement. Just
3. It just check current state. It is efficient because some algorithm should check previous state. Then it can cause lack of memory or overhead.