Heuristic Analysis

In this project, we are developing an adversarial search agent to play the game "Isolation". We have used minimax method, alpha-beta method and iterative deepening method to build the AI agent. Besides these methods, we also need to develop customized heuristic evaluation methods to help the AI agent decide what is the best move.

How to design a powerful evaluation function is a challenge. In this project, I have tried 6 different evaluation functions. They are as bellows:

1. The first one is similar to improved evaluation function and also mentioned in lecture:

```
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
return float(own_moves - 2.*opp_moves)
```

2. The second one is little bit different. The function will calculate all the available moves which is only maximum two grid away from the opponent player, such that the agent will try to block the move of the opponent.

```
y, x = game.get_player_location(game.get_opponent(player))
count = 0
for move in game.get_legal_moves(player):
    if max(abs(move[0]-y), abs(move[1]-x)) < 2.0001:
        count += 1
return count
```

3. The 3rd function is a combination of 1st and 2nd with weights, shown as below:

   return float(8*(own_moves - 2*opp_moves) + 2*count)

4. The 4th function is a combination of improved function, 2nd function and center_score function from the sample_palyers, shown as below:

   return float(8*(own_moves - opp_moves) + 2*count) + float((h - y)**2 + (w - x)**2)

5. The 5th function is a combination of improved function and center_score functions, shown as below:

   return float(8*(own_moves - opp_moves)) + float((h - y)**2 + (w - x)**2)

6. The 6th function is a combination of 1st function, 2nd function and center_score function:

   return float(10*(own_moves - 2*opp_moves) + 2*count) + float((h - y)**2 + (w - x)**2)

The win rate is shown below:

Win Rate:	AB_Improved	AB_Custom	AB_Custom_2	AB_Custom_3	AB_Custom_4	AB_Custom_5	AB_Custom_6
	62.9%	60.0%	57.1%	58.6%	71.4%	58.6%	67.1%

As we can find, for all the functions we have here, the 4th evaluation has achieved the best win rate, 3rd and 5th functions have the worst win rate. The idea of using combination of functions with weights is coming from the section 5.4.1 of the artificial intelligence textbook. By using a linear function, it should have the potential to improve the win rate as it will count all kinds of situations might happen in the game. The weights I used for all the functions are just rough estimations. Ideally I should have a more fundamental analysis on how to derive the weights or come up a better idea to calculate the weights but I don’t have any clue how should I do that. Besides, in every test, there is a large bias between the win rate for the exact same evaluation functions. I still need to dig deeper to figure out what’s the cause and find a better solution to improve my win rate.

This is a very good project for starting learning AI and has brought me a lot of thinking. The win rate I have achieved is below my expectation. In the future with more study on AI, I believe I can come up a better idea for this project.