Heuristics Analysis

Artificial Intelligent Nanodegree Program

Project 2: Building a Game Playing Agent

Jeanne Sim

17 July 2017

Experience with this project:

In video “16. Solving 5x5 Isolation”, Malcom introduced the methods he used for this project.

1) Symmetry
    a. Using horizontal axis
    b. Using diagonal axis
    c. Center point
2) Partition
    a. Area where the player has more moves indicates that the player is a winner
3) Reflection
    a. Reflect the opponent’s move
        i. Player 1 wins by reflecting Player 2’s move
        ii. Player 2 wins by moving to a spot that Player 1 cannot reflect Player 2’s move

However, I was having a tough time figuring out how to code heuristics after centrality. As such, I decided to revisit all the tutorial videos to familiarise with the concepts taught. Video “9. Evaluating Evaluation Functions” spoke of evaluation functions which I thought will be perfect for the project.

- #my_moves - #opponent_moves

Where the computer player seeks moves with the most options while trying to get in the way of the opponent’s move.

- #my_moves - 2 * #opponent_moves

A more aggressive function than the former. Instead, the computer player chases after the opponent.

With these functions, I proceeded to add them to the custom_score2 and custom_score3 respectively.

The heuristics methods I used for this project consist:

1) custom_score

```
if game.is_loser(player):
      return float("-inf")
if game.is_winner(player):
      return float("inf")
return float(len(game.get_legal_moves(player)))
```

This method gets any legal moves of the player.

2) custom_score2

```
if game.is_loser(player):
      return float("-inf")
if game.is_winner(player):
      return float("inf")
my_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
return float(my_moves - opp_moves)
```

This method is the less aggressive method where the the computer player seeks moves with the most options while trying to get in the way of the opponent's move.

3) custom_score3

```
if game.is_loser(player):
      return float("-inf")
if game.is_winner(player):
      return float("inf")
my_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
return float(my_moves - 2 * opp_moves)
```

Unlike custom_score2, this method is more aggressive where the computer player chases after the player.