Heuristic Analysis

Isolation-Playing Agent through Adversarial Search

Implementation

Here three custom heuristics are implemented.

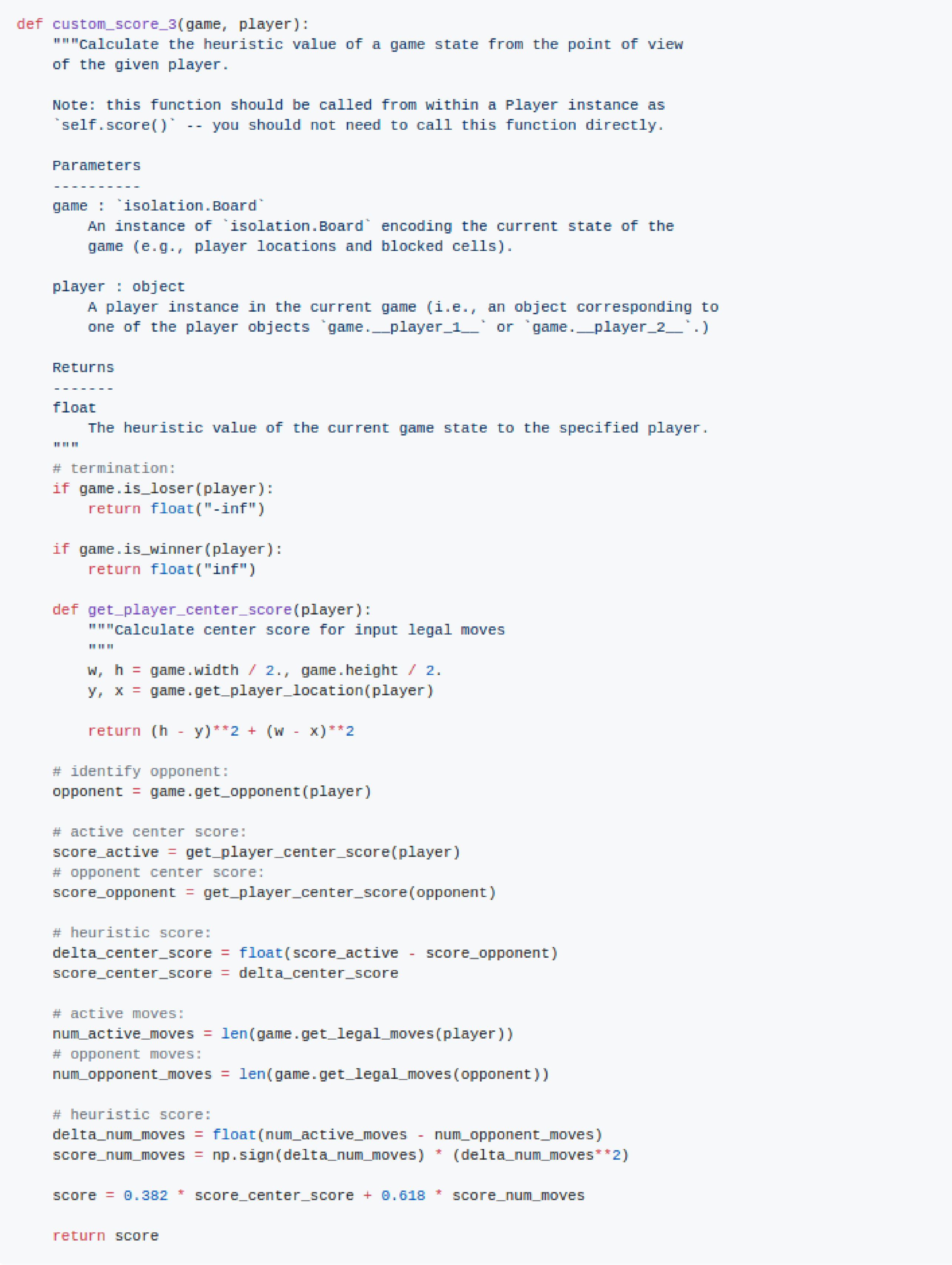
The **first** one is just a simple modification of the improved heuristic. Here L2 norm is used to encourage the agent to pursuit a larger margin of the number of legal moves between active and opponent players.

```
def custom_score(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.

    This should be the best heuristic function for your project submission.

    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    -------
    float
        The heuristic value of the current game state to the specified player.
    """
    # termination:
    if game.is_loser(player):
        return float("-inf")

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

    # identify opponent:
    opponent = game.get_opponent(player)

    # active moves:
    num_active_moves = len(game.get_legal_moves(player))
    # opponent moves:
    num_opponent_moves = len(game.get_legal_moves(opponent))

    # heuristic score:
    delta = float(num_active_moves - num_opponent_moves)

    score = np.sign(delta) * (delta**2)

    return score
```

The **second** one is also a simple modification of the improved heuristic. Here L3 norm is used to encourage the agent to pursuit a larger margin of the number of legal moves between active and opponent players.

```python
def custom_score_2(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.

    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    -------
    float
        The heuristic value of the current game state to the specified player.
    """
    # termination:
    if game.is_loser(player):
        return float("-inf")

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

    # identify opponent:
    opponent = game.get_opponent(player)

    # active moves:
    num_active_moves = len(game.get_legal_moves(player))
    # opponent moves:
    num_opponent_moves = len(game.get_legal_moves(opponent))

    # heuristic score:
    delta = float(num_active_moves - num_opponent_moves)

    score = delta**3

    return score
```

The **third** one combines the ideas from improved and center heuristics. Here a combined score is calculated from both the difference of number of legal moves and center score between active and opponent players.

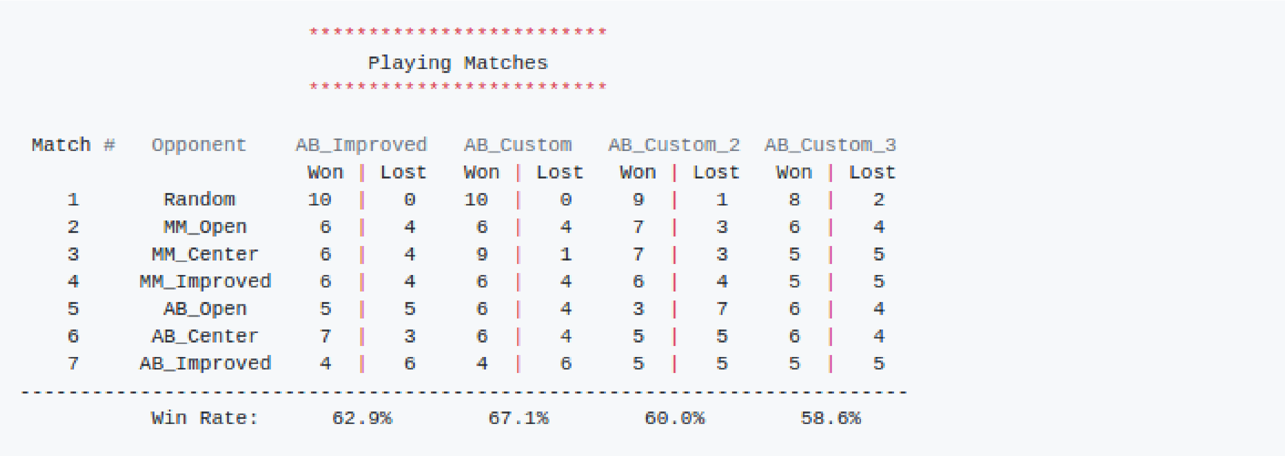


```python
def custom_score_3(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.

    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    -------
    float
        The heuristic value of the current game state to the specified player.
    """
    # termination:
    if game.is_loser(player):
        return float("-inf")

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

    def get_player_center_score(player):
        """Calculate center score for input legal moves
        """
        w, h = game.width / 2., game.height / 2.
        y, x = game.get_player_location(player)

        return (h - y)**2 + (w - x)**2

    # identify opponent:
    opponent = game.get_opponent(player)

    # active center score:
    score_active = get_player_center_score(player)
    # opponent center score:
    score_opponent = get_player_center_score(opponent)

    # heuristic score:
    delta_center_score = float(score_active - score_opponent)
    score_center_score = delta_center_score

    # active moves:
    num_active_moves = len(game.get_legal_moves(player))
    # opponent moves:
    num_opponent_moves = len(game.get_legal_moves(opponent))

    # heuristic score:
    delta_num_moves = float(num_active_moves - num_opponent_moves)
    score_num_moves = np.sign(delta_num_moves) * (delta_num_moves**2)

    score = 0.382 * score_center_score + 0.618 * score_num_moves

    return score
```

Performance Summary

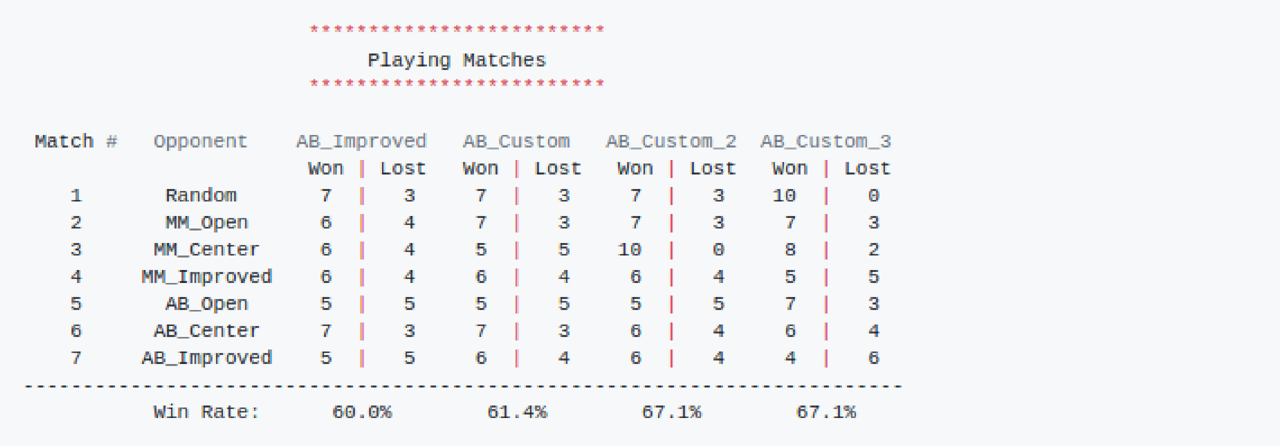
The tournament was carried out three times and the results are as follows:

Tournament 1



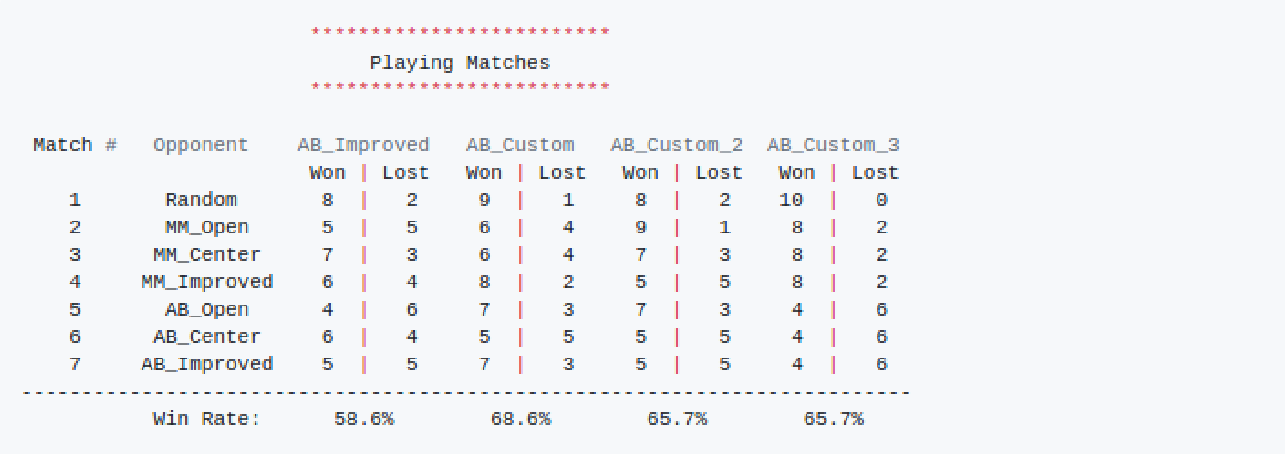
```
                         *************************
                              Playing Matches
                         *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      10  |   0    10  |   0     9  |   1     8  |   2
    2       MM_Open      6  |   4     6  |   4     7  |   3     6  |   4
    3      MM_Center     6  |   4     9  |   1     7  |   3     5  |   5
    4     MM_Improved    6  |   4     6  |   4     6  |   4     5  |   5
    5       AB_Open      5  |   5     6  |   4     3  |   7     6  |   4
    6      AB_Center     7  |   3     6  |   4     5  |   5     6  |   4
    7     AB_Improved    4  |   6     4  |   6     5  |   5     5  |   5
--------------------------------------------------------------------------
           Win Rate:      62.9%        67.1%        60.0%        58.6%
```

Tournament 2



```
                         *************************
                              Playing Matches
                         *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       7  |   3     7  |   3     7  |   3    10  |   0
    2       MM_Open      6  |   4     7  |   3     7  |   3     7  |   3
    3      MM_Center     6  |   4     5  |   5    10  |   0     8  |   2
    4     MM_Improved    6  |   4     6  |   4     6  |   4     5  |   5
    5       AB_Open      5  |   5     5  |   5     5  |   5     7  |   3
    6      AB_Center     7  |   3     7  |   3     6  |   4     6  |   4
    7     AB_Improved    5  |   5     6  |   4     6  |   4     4  |   6
--------------------------------------------------------------------------
           Win Rate:      60.0%        61.4%        67.1%        67.1%
```

Tournament 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       8  |   2     9  |   1     8  |   2    10  |   0
    2       MM_Open      5  |   5     6  |   4     9  |   1     8  |   2
    3      MM_Center     7  |   3     6  |   4     7  |   3     8  |   2
    4     MM_Improved    6  |   4     8  |   2     5  |   5     8  |   2
    5       AB_Open      4  |   6     7  |   3     7  |   3     4  |   6
    6      AB_Center     6  |   4     5  |   5     5  |   5     4  |   6
    7     AB_Improved    5  |   5     7  |   3     5  |   5     4  |   6
--------------------------------------------------------------------------
           Win Rate:      58.6%        68.6%        65.7%        65.7%
```

Heuristic Recommendation

From the above results, the custom heuristic using L2 norm outperforms the baseline one 3 out of 3 times.

So here the first custom heuristic is recommended for the following reasons:

1. Steadily improved performance over the baseline heuristic.
2. No extra computing overhead(compared with heuristics based on complex human knowledge).
3. No hyper-parameter tuning(compared with the third custom heuristic).