Heuristic Analysis

- author: Xin Chen
- email: Bismarrck@me.com

1. Overview

In this project three different parameterized heuristics are implemented. The parameters for these score functions are learnt with grid search:

```
python3 grid_search.py {fn1|fn2|fn3} --num_matches=10
```

2. Custom Score 3

The heuristic `custom_score_3` is an improved version of the score function `improved_score`:

```
float(num_own_moves * a - num_opp_moves * b)
```

- The best parameter set is `a = 2, b = 3`.
- The grid search suggests that the win rate is improved from **`104/140`** to **`108/140`**.

3. Custom Score 2

The heuristic `custom_score_2` not only considers the current available legal moves (`custom_score_3`) but also tries to include the possible moves for next turn.

```
player_score = num_next_own * b + num_own_moves * a
opp_score = num_next_opp * c
return float(player_score - opp_score)
```

- `num_opp_moves` is ignored because it is included in `num_next_opp`.
- The best parameter set is `a = 1, b = 1, c = 1`
- The grid search suggests the win rate is around **`80%`**.

4. Custom Score 1

The heuristic `custom_score` is further improved from `custom_score_2` as only unique future moves are included:

```
num_own_controlled = len(set(own_controlled))
num_opp_controlled = len(set(opp_controlled))
```

The final score has four learnable parameters:

```
own_score = num_own_moves * a + num_own_controlled * c
opp_score = num_opp_moves * b + num_opp_controlled * d
return float(own_score - opp_score)
```

- The best parameter set is `a = 3, b = 2, c = 1, d = 1`
- The grid search suggests that the win rate is around **`85%`**.