final_report_stats

December 13, 2019

1 ai-chess-agent statistics

[1]:
```
import numpy as np
import pandas as pd
import csv
import os
import matplotlib.pyplot as plt
```

[2]:
```
def get_csv_paths(dir):
    files = dict()
    for (dirpath, dirnames, filenames) in os.walk(dir):
        for file in filenames:
            if file.endswith(".csv"):
                files[file] = os.path.join(dirpath, file)
    file_list = []
    for item in sorted(files.keys()):
        file_list.append([files[item], item.split(".")[0]])

    return file_list
```

[3]:
```
f_list = get_csv_paths("./../src/driver_notebooks/results/")
COLUMNS = ['round_num','iterations', 'depth', 'white_agent','black_agent',
           'white_victory','winner','moves_played','remaining_w_pieces',
           'remaining_b_pieces', 'remaining_tot_pieces']
```

[4]:
```
df_from_each_file = (pd.read_csv(f[0], names=COLUMNS, header=0) for f in f_list)
```

[5]:
```
df = pd.concat(df_from_each_file, ignore_index=True)
```

1.0.1 Total number of games played

[6]:
```
games = df['round_num'].count()
games
```

[6]: 340

[7]:
```
white_checkmate_df = df.apply(lambda x: True if x['winner'] == 'checkmate:␣
↪White wins!' else False , axis=1)
white_checkmate_num = len(white_checkmate_df[white_checkmate_df == True].index)
```

[8]:
```
black_checkmate_df = df.apply(lambda x: True if x['winner'] == 'checkmate:␣
↪Black wins!' else False , axis=1)
black_checkmate_num = len(black_checkmate_df[black_checkmate_df == True].index)
```

[9]:
```
draw_stalemate_df = df.apply(lambda x: True if x['winner'] == "draw: stalemate"␣
↪else False , axis=1)
draw_stalemate_num = len(draw_stalemate_df[draw_stalemate_df == True].index)
```

[10]:
```
draw_fivefold_df = df.apply(lambda x: True if x['winner'] == "draw: 5-fold␣
↪repetition" else False , axis=1)
draw_fivefold_num = len(draw_fivefold_df[draw_fivefold_df == True].index)
```

[11]:
```
draw_insufficient_material_df = df.apply(lambda x: True if x['winner'] == "draw:
↪ insufficient material" else False , axis=1)
draw_insufficient_material_num =␣
↪len(draw_insufficient_material_df[draw_insufficient_material_df == True].
↪index)
```

[12]:
```
draw_claim_df = df.apply(lambda x: True if x['winner'] == "draw: claim" else␣
↪False , axis=1)
draw_claim_num = len(draw_claim_df[draw_claim_df == True].index)
```

1.0.2 Overall results

[13]:
```
winner_df = pd.DataFrame([(white_checkmate_num, black_checkmate_num,␣
↪draw_stalemate_num, draw_fivefold_num, draw_insufficient_material_num,␣
↪draw_claim_num)],
columns = ['checkmate: White', 'checkmate: Black', 'draw: stalemate', 'draw:␣
↪5-fold repetition', 'draw: insufficient material', 'draw: claim'])
winner_df.style.hide_index()
```

[13]:
```
<pandas.io.formats.style.Styler at 0x7fa39890d2e8>
```

1.0.3 Overall Percentages

[14]:
```
winnings = df['winner']
counts = winnings.value_counts()
percent = winnings.value_counts(normalize=True)
percent100 = winnings.value_counts(normalize=True).mul(100).round(1).
↪astype(str) + '%'
win_df = pd.DataFrame({'counts': counts, 'percent': percent, 'percent 100':␣
↪percent100 })
```

```
win_df
```

[14]:

	counts	percent	percent 100
checkmate: Black wins!	172	0.505882	50.6%
checkmate: White wins!	106	0.311765	31.2%
draw: claim	38	0.111765	11.2%
draw: stalemate	22	0.064706	6.5%
draw: insufficient material	2	0.005882	0.6%

1.0.4 Top 10 games, ordered by moves played ascending

[15]:
```
df.sort_values(by=['moves_played'], inplace=False, ascending=True).head(10)
```

[15]:

	round_num	iterations	depth	white_agent	black_agent	white_victory	winner	moves_played	remaining_w_pieces	remaining_b_pieces	remaining_tot_pieces
323	4	10	NaN	random_agent	stockfish	False	checkmate: Black wins!	6	16	16	32
146	7	10	1.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	7	16	16	32
334	5	10	NaN	improved_random_agent	stockfish	False	checkmate: Black wins!	8	15	14	29
12	3	10	2.0	advanced_minimax_agent	random_agent	True	checkmate: White wins!	11	16	16	32
144	5	10	1.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	11	16	13	29
198	9	10	2.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	13	16	15	31
158	9	10	2.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	13	16	15	31
191	2	10	2.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	13	16	15	31
320	1	10	NaN	random_agent	stockfish	False	checkmate: Black wins!	14	14	16	30
14	5	10	2.0	advanced_minimax_agent	random_agent	True	checkmate: White wins!	15	16	16	32

```
14                  12                  28
```

1.0.5 Top 10 games where the white agent wins, ordered by moves played ascending

[16]:
```
df.loc[df['winner'] == 'checkmate: White wins!'].
↪sort_values(by=['moves_played'], inplace=False, ascending=True).head(10)
```

[16]:

	round_num	iterations	depth	white_agent	black_agent	white_victory	winner	moves_played	remaining_w_pieces	remaining_b_pieces	remaining_tot_pieces
146	7	10	1.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	7	16	16	32
144	5	10	1.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	11	16	13	29
12	3	10	2.0	advanced_minimax_agent	random_agent	True	checkmate: White wins!	11	16	16	32
158	9	10	2.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	13	16	15	31
191	2	10	2.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	13	16	15	31
198	9	10	2.0	improved_minimax_agent	random_agent	True	checkmate: White wins!	13	16	15	31
14	5	10	2.0	advanced_minimax_agent	random_agent	True	checkmate: White wins!	15	16	12	28
107	8	10	NaN	advanced_agent	random_agent	True	checkmate: White wins!	17	15	12	27
1	2	10	1.0	advanced_minimax_agent	random_agent	True	checkmate: White wins!	19	16	15	31
3	4	10	1.0	advanced_minimax_agent	random_agent	True	checkmate: White wins!	19	16	12	28

1.0.6 Games where the white agent wins and Oppnent Agent is Stockfish Engine, ordered by moves played ascending

[17]:
```
df[(df['winner'] == 'checkmate: White wins!') & (df['black_agent'] ==␣
↪'stockfish')]
```

[17]:
```
Empty DataFrame
Columns: [round_num, iterations, depth, white_agent, black_agent, white_victory,
winner, moves_played, remaining_w_pieces, remaining_b_pieces,
remaining_tot_pieces]
Index: []
```

1.0.7 Top 10 games with the fewest remaining black pieces, ordered by pieces remaining ascending

[18]:
```
df.sort_values(by=['remaining_b_pieces'], inplace=False, ascending=True).
↪head(10)
```

[18]:

	round_num	iterations	depth	white_agent	\
109	10	10	NaN	advanced_agent	
315	6	10	NaN	improved_random_agent	
312	3	10	NaN	improved_random_agent	
86	7	10	NaN	naive_agent	
87	8	10	NaN	naive_agent	
268	9	10	1.0	naive_alpha-beta_minimax_agent	
56	7	10	2.0	advanced_alpha-beta_minimax_agent	
89	10	10	NaN	naive_agent	
54	5	10	2.0	advanced_alpha-beta_minimax_agent	
47	8	10	1.0	advanced_alpha-beta_minimax_agent	

	black_agent	white_victory	winner	moves_played	\
109	random_agent	True	checkmate: White wins!	69	
315	random_agent	False	draw: stalemate	101	
312	random_agent	False	draw: stalemate	149	
86	random_agent	False	draw: claim	121	
87	random_agent	False	draw: claim	57	
268	random_agent	False	draw: stalemate	129	
56	random_agent	True	checkmate: White wins!	77	
89	random_agent	False	draw: claim	115	
54	random_agent	True	checkmate: White wins!	61	
47	random_agent	False	draw: stalemate	59	

	remaining_w_pieces	remaining_b_pieces	remaining_tot_pieces
109	13	1	14
315	13	1	14
312	11	1	12
86	12	1	13

87	13	1	14
268	15	1	16
56	11	1	12
89	11	1	12
54	13	1	14
47	15	1	16

1.0.8 Top 10 games with fewest remaining black pieces where white wins, ordered by black pieces remaining ascending

[19]:
```
df[(df['winner'] == 'checkmate: White wins!')].
↪sort_values(by=['remaining_b_pieces'], inplace=False, ascending=True).
↪head(10)
```

[19]:

	round_num	iterations	depth	white_agent	\
310	1	10	NaN	improved_random_agent	
105	6	10	NaN	advanced_agent	
104	5	10	NaN	advanced_agent	
103	4	10	NaN	advanced_agent	
102	3	10	NaN	advanced_agent	
98	9	10	NaN	improved_agent	
56	7	10	2.0	advanced_alpha-beta_minimax_agent	
54	5	10	2.0	advanced_alpha-beta_minimax_agent	
109	10	10	NaN	advanced_agent	
148	9	10	1.0	improved_minimax_agent	

	black_agent	white_victory	winner	moves_played	\
310	random_agent	True	checkmate: White wins!	291	
105	random_agent	True	checkmate: White wins!	61	
104	random_agent	True	checkmate: White wins!	63	
103	random_agent	True	checkmate: White wins!	67	
102	random_agent	True	checkmate: White wins!	61	
98	random_agent	True	checkmate: White wins!	67	
56	random_agent	True	checkmate: White wins!	77	
54	random_agent	True	checkmate: White wins!	61	
109	random_agent	True	checkmate: White wins!	69	
148	random_agent	True	checkmate: White wins!	125	

	remaining_w_pieces	remaining_b_pieces	remaining_tot_pieces
310	9	1	10
105	14	1	15
104	12	1	13
103	15	1	16
102	15	1	16
98	12	1	13
56	11	1	12
54	13	1	14

```
109                 13                  1                  14
148                 12                  1                  13
```

1.0.9 Top 10 games with the fewest remaining white pieces, ordered by pieces remaining ascending

[20]:
```
df.sort_values(by=['remaining_w_pieces'], inplace=False, ascending=True).
↪head(10)
```

[20]:

	round_num	iterations	depth	white_agent	black_agent	white_victory	winner	moves_played	remaining_w_pieces	remaining_b_pieces	remaining_tot_pieces
301	2	10	NaN	random_agent	random_agent	False	draw: stalemate	164	1	7	8
306	7	10	NaN	random_agent	random_agent	False	draw: insufficient material	416	1	2	3
309	10	10	NaN	random_agent	random_agent	False	draw: insufficient material	519	1	2	3
303	4	10	NaN	random_agent	random_agent	False	draw: stalemate	266	1	6	7
76	7	10	2.0	advanced_alpha-beta_minimax_agent	stockfish	False	checkmate: Black wins!	56	2	13	15
135	6	10	NaN	advanced_agent	stockfish	False	checkmate: Black wins!	54	2	11	13
305	6	10	NaN	random_agent	random_agent	False	draw: claim	394	2	2	4
302	3	10	NaN	random_agent	random_agent	False	draw: claim	497	2	1	3
308	9	10	NaN	random_agent	random_agent	False	draw: claim	297	2	3	5
307	8	10	NaN	random_agent	random_agent	False	draw: claim	410	2	3	5

1.0.10 Top 10 games with fewest remaining white pieces where black wins, ordered by white pieces remaining ascending

[21]:
```
df[(df['winner'] == 'checkmate: Black wins!')].
↪sort_values(by=['remaining_w_pieces'], inplace=False, ascending=True).
↪head(10)
```

[21]:

	round_num	iterations	depth	white_agent	black_agent	white_victory	winner	moves_played	remaining_w_pieces	remaining_b_pieces	remaining_tot_pieces
135	6	10	NaN	advanced_agent	stockfish	False	checkmate: Black wins!	54	2	11	13
71	2	10	2.0	advanced_alpha-beta_minimax_agent	stockfish	False	checkmate: Black wins!	54	2	13	15
76	7	10	2.0	advanced_alpha-beta_minimax_agent	stockfish	False	checkmate: Black wins!	56	2	13	15
64	5	10	1.0	advanced_alpha-beta_minimax_agent	stockfish	False	checkmate: Black wins!	68	3	11	14
176	7	10	2.0	improved_minimax_agent	stockfish	False	checkmate: Black wins!	72	3	8	11
25	6	10	1.0	advanced_minimax_agent	stockfish	False	checkmate: Black wins!	56	3	10	13
200	1	10	1.0	improved_minimax_agent	stockfish	False	checkmate: Black wins!	50	3	10	13
206	7	10	1.0	improved_minimax_agent	stockfish	False	checkmate: Black wins!	52	3	10	13
132	3	10	NaN	advanced_agent	stockfish	False	checkmate: Black wins!	50	3	9	12
179	10	10	2.0	improved_minimax_agent	stockfish	False	checkmate: Black wins!	52	4	11	15