Udacity AIND Project 3 Game Agent Run Matches Results – July 2018

Terminal: $python run_match.py -r 100

Terminal: $python run_match.py -r 100 -o SELF

Opening book On/Off switch is in code file: my_custom_player.py, USE_OPENING_BOOK

info extract from matches.log

	Self VS AI Minimax								Self VS Self					
	Random Opening Moves				Opening Book				Random Opening Moves			Opening Book		
game	player1	m1	m2	won	player1	m1	m2	won	player1	m1	m2	player1	m1	m2
1	AI	14	16	1	AI	62	58	1	Self	95	23	Self	57	58
2	Self	105	55	0	Self	57	43	1	Self	15	81	Self	57	58
3	AI	9	110	0	AI	45	57	1	Self	104	82	Self	57	58
4	Self	88	97	1	Self	57	23	1	Self	29	31	Self	57	58
5	AI	52	61	0	AI	2	56	1	Self	3	0	Self	57	58
6	Self	93	40	1	Self	57	101	1	Self	33	17	Self	57	58
7	AI	93	105	1	AI	93	57	0	Self	3	46	Self	57	58
8	Self	32	108	1	Self	57	96	1	Self	107	18	Self	57	58
9	AI	23	72	1	AI	48	57	1	Self	78	21	Self	57	58
10	Self	94	66	0	Self	57	100	0	Self	57	23	Self	57	58
11	AI	92	18	0	AI	61	57	0	Self	6	81	Self	57	58
12	Self	14	29	0	Self	57	32	1	Self	5	66	Self	57	58
13	AI	101	91	0	AI	48	57	1	Self	34	108	Self	57	58
14	Self	21	22	0	Self	57	84	1	Self	107	59	Self	57	58
15	AI	104	94	1	AI	3	57	1	Self	10	52	Self	57	58
16	Self	92	83	0	Self	57	4	0	Self	75	72	Self	57	58
17	AI	78	99	1	AI	113	71	1	Self	96	13	Self	57	58
18	Self	42	9	0	Self	57	92	0	Self	99	104	Self	57	58
19	AI	36	9	0	AI	105	69	1	Self	21	85	Self	57	58
20	Self	21	42	1	Self	57	33	1	Self	58	105	Self	57	58
21	AI	20	109	0	AI	78	56	0	Self	18	53	Self	57	58
22	Self	58	16	1	Self	57	30	0	Self	96	61	Self	57	58
23	AI	4	20	0	AI	88	57	0	Self	107	1	Self	57	58
24	Self	23	32	1	Self	57	113	1	Self	99	40	Self	57	58
25	AI	80	1	1	AI	28	58	1	Self	18	14	Self	57	58
26	Self	109	47	1	Self	57	105	0	Self	9	71	Self	57	58
27	AI	95	2	0	AI	45	57	1	Self	43	10	Self	57	58
28	Self	98	68	0	Self	57	110	0	Self	88	100	Self	57	58
29	AI	34	1	1	AI	19	44	1	Self	9	70	Self	57	58
30	Self	44	31	0	Self	57	40	0	Self	106	74	Self	57	58
31	AI	2	110	0	AI	35	56	1	Self	85	97	Self	57	58
32	Self	74	82	0	Self	57	74	0	Self	49	73	Self	57	58
33	AI	10	107	1	AI	41	56	1	Self	104	113	Self	57	58
34	Self	110	61	1	Self	57	61	1	Self	15	46	Self	57	58
35	AI	101	83	0	AI	52	56	0	Self	49	19	Self	57	58
36	Self	10	31	0	Self	57	79	0	Self	73	21	Self	57	58
37	AI	109	41	0	AI	62	58	1	Self	86	75	Self	57	58
38	Self	74	111	1	Self	57	7	0	Self	41	67	Self	57	58
39	AI	18	5	1	AI	65	58	1	Self	95	52	Self	57	58
40	Self	57	23	1	Self	57	68	1	Self	5	75	Self	57	58
41	AI	78	29	1	AI	43	58	1	Self	67	46	Self	57	58
42	Self	35	106	0	Self	57	67	0	Self	60	83	Self	57	58
43	AI	70	26	0	AI	14	56	0	Self	93	87	Self	57	58
44	Self	99	85	0	Self	57	26	0	Self	110	96	Self	57	58

45	AI	75	52	0	AI	27	57	0	Self	65	110	Self	57	58
46	Self	109	60	0	Self	57	45	0	Self	46	70	Self	57	58
47	AI	100	2	0	AI	10	56	0	Self	42	80	Self	57	58
48	Self	74	31	0	Self	57	68	1	Self	55	105	Self	57	58
49	AI	46	23	0	AI	61	57	0	Self	112	52	Self	57	58
50	Self	109	58	0	Self	57	52	0	Self	105	112	Self	57	58
51	AI	9	58	1	AI	41	56	1	Self	68	40	Self	57	58
52	Self	5	94	0	Self	57	21	1	Self	60	30	Self	57	58
53	AI	73	58	1	AI	67	56	0	Self	87	55	Self	57	58
54	Self	23	60	0	Self	57	31	1	Self	96	43	Self	57	58
55	AI	78	2	0	AI	80	57	0	Self	74	1	Self	57	58
56	Self	104	78	1	Self	57	46	0	Self	98	31	Self	57	58
57	AI	114	66	1	AI	82	57	1	Self	94	7	Self	57	58
58	Self	17	26	1	Self	57	43	1	Self	65	109	Self	57	58
59	AI	27	98	0	AI	80	57	0	Self	14	81	Self	57	58
60	Self	49	87	1	Self	57	73	0	Self	41	46	Self	57	58
61	AI	85	19	0	AI	21	56	1	Self	31	18	Self	57	58
62	Self	49	2	1	Self	57	35	1	Self	53	93	Self	57	58
63	AI	86	54	1	AI	56	58	1	Self	114	47	Self	57	58
64	Self	80	101	0	Self	57	108	0	Self	85	55	Self	57	58
65	AI	18	0	0	AI	85	58	1	Self	33	45	Self	57	58
66	Self	4	81	0	Self	57	6	0	Self	82	0	Self	57	58
67	AI	84	22	0	AI	49	58	0	Self	18	32	Self	57	58
68	Self	72	106	1	Self	57	43	1	Self	9	59	Self	57	58
69	AI	100	5	1	AI	81	56	1	Self	82	19	Self	57	58
70	Self	26	27	1	Self	57	15	0	Self	82	0	Self	57	58
71	AI	41	29	1	AI	46	57	1	Self	58	84	Self	57	58
72	Self	34	13	0	Self	57	82	0	Self	44	55	Self	57	58
73	AI	83	4	0	AI	54	56	0	Self	30	21	Self	57	58
74	Self	109	101	0	Self	57	60	1	Self	35	21	Self	57	58
75	AI	21	66	1	AI	36	58	1	Self	57	72	Self	57	58
76	Self	2	5	0	Self	57	23	1	Self	62	93	Self	57	58
77	AI	72	10	1	AI	40	55	1	Self	40	29	Self	57	58
78	Self	97	104	1	Self	57	0	0	Self	8	82	Self	57	58
79	AI	5	42	0	AI	18	56	1	Self	114	28	Self	57	58
80	Self	23	18	1	Self	57	66	1	Self	105	34	Self	57	58
81	AI	10	17	1	AI	54	56	0	Self	61	23	Self	57	58
82	Self	23	10	1	Self	57	23	1	Self	14	46	Self	57	58
83	AI	9	56	1	AI	36	58	1	Self	29	20	Self	57	58
84	Self	111	106	1	Self	57	105	0	Self	111	45	Self	57	58
85	AI	73	4	1	AI	20	56	1	Self	68	3	Self	57	58
86	Self	9	112	0	Self	57	73	0	Self	23	7	Self	57	58
87	AI	97	109	0	AI	91	56	0	Self	68	80	Self	57	58
88	Self	99	109	0	Self	57	32	1	Self	19	18	Self	57	58
89	AI	4	57	1	AI	62	58	1	Self	3	62	Self	57	58
90	Self	21	95	1	Self	57	104	1	Self	40	2	Self	57	58
91	AI	62	33	1	AI	104	58	1	Self	93	101	Self	57	58
92	Self	13	5	1	Self	57	27	1	Self	95	82	Self	57	58
93	AI	18	100	0	AI	60	57	1	Self	33	85	Self	57	58
94	Self	95	84	0	Self	57	85	0	Self	45	17	Self	57	58
95	AI	18	7	1	AI	98	56	0	Self	4	75	Self	57	58
96	Self	32	26	0	Self	57	41	1	Self	0	81	Self	57	58
97	AI	31	52	1	AI	68	57	1	Self	43	32	Self	57	58

98	Self	41	75	0	Self	57	62	1	Self	49	58	Self	57	58
99	AI	105	111	1	AI	79	58	0	Self	31	13	Self	57	58
100	Self	58	27	0	Self	57	54	1	Self	81	42	Self	57	58
101	AI	23	33	0	AI	54	56	0	Self	84	48	Self	57	58
102	Self	55	100	0	Self	57	3	1	Self	6	53	Self	57	58
103	AI	91	69	1	AI	53	57	0	Self	78	60	Self	57	58
104	Self	84	100	1	Self	57	18	1	Self	52	19	Self	57	58
105	AI	97	56	0	AI	100	58	0	Self	9	8	Self	57	58
106	Self	52	8	0	Self	57	32	1	Self	112	73	Self	57	58
107	AI	59	48	0	AI	33	58	1	Self	75	73	Self	57	58
108	Self	67	13	1	Self	57	20	0	Self	66	53	Self	57	58
109	AI	42	78	1	AI	28	58	1	Self	67	114	Self	57	58
110	Self	113	14	1	Self	57	67	0	Self	8	56	Self	57	58
111	AI	10	0	0	AI	65	58	1	Self	92	28	Self	57	58
112	Self	106	4	1	Self	57	54	1	Self	83	78	Self	57	58
113	AI	65	108	1	AI	58	56	0	Self	97	68	Self	57	58
114	Self	92	34	1	Self	57	41	1	Self	43	58	Self	57	58
115	AI	5	32	1	AI	83	56	0	Self	88	59	Self	57	58
116	Self	97	71	0	Self	57	101	1	Self	32	113	Self	57	58
117	AI	17	91	1	AI	52	56	0	Self	6	69	Self	57	58
118	Self	19	65	0	Self	57	41	1	Self	99	96	Self	57	58
119	AI	5	104	1	AI	55	56	1	Self	105	83	Self	57	58
120	Self	46	52	0	Self	57	66	1	Self	20	56	Self	57	58
121	AI	28	59	0	AI	10	56	0	Self	55	94	Self	57	58
122	Self	73	65	1	Self	57	15	0	Self	34	17	Self	57	58
123	AI	74	30	1	AI	108	56	1	Self	2	4	Self	57	58
124	Self	16	55	0	Self	57	5	1	Self	30	21	Self	57	58
125	AI	108	96	0	AI	54	56	0	Self	17	58	Self	57	58
126	Self	43	31	0	Self	57	42	0	Self	88	85	Self	57	58
127	AI	104	46	0	AI	33	58	1	Self	43	84	Self	57	58
128	Self	13	95	1	Self	57	70	1	Self	96	52	Self	57	58
129	AI	27	26	0	AI	114	57	1	Self	5	47	Self	57	58
130	Self	114	71	0	Self	57	113	1	Self	82	88	Self	57	58
131	AI	10	28	0	AI	79	58	0	Self	82	27	Self	57	58
132	Self	112	7	1	Self	57	3	1	Self	112	78	Self	57	58
133	AI	22	70	0	AI	72	57	1	Self	104	33	Self	57	58
134	Self	67	84	1	Self	57	56	1	Self	114	43	Self	57	58
135	AI	60	9	0	AI	80	57	0	Self	105	34	Self	57	58
136	Self	114	65	0	Self	57	75	0	Self	113	5	Self	57	58
137	AI	6	52	0	AI	109	57	0	Self	44	9	Self	57	58
138	Self	33	105	0	Self	57	58	0	Self	83	29	Self	57	58
139	AI	58	55	1	AI	67	56	0	Self	56	108	Self	57	58
140	Self	72	9	1	Self	57	94	1	Self	48	44	Self	57	58
141	AI	20	5	0	AI	79	58	0	Self	14	44	Self	57	58
142	Self	82	4	1	Self	57	43	1	Self	84	79	Self	57	58
143	AI	41	82	0	AI	39	56	1	Self	54	58	Self	57	58
144	Self	82	44	1	Self	57	33	1	Self	14	87	Self	57	58
145	AI	96	75	1	AI	26	58	0	Self	32	72	Self	57	58
146	Self	48	31	1	Self	57	4	0	Self	43	56	Self	57	58
147	AI	104	53	0	AI	62	58	1	Self	55	79	Self	57	58
148	Self	22	85	0	Self	57	26	0	Self	46	85	Self	57	58
149	AI	67	16	1	AI	4	56	0	Self	47	4	Self	57	58
150	Self	73	3	1	Self	57	16	1	Self	104	62	Self	57	58

151	AI	94	13	0	AI	58	56	0	Self	28	106	Self	57	58
152	Self	58	36	0	Self	57	7	0	Self	27	78	Self	57	58
153	AI	56	82	0	AI	94	56	0	Self	54	26	Self	57	58
154	Self	108	30	0	Self	57	49	1	Self	5	82	Self	57	58
155	AI	15	35	0	AI	94	56	0	Self	93	7	Self	57	58
156	Self	59	48	0	Self	57	44	0	Self	67	6	Self	57	58
157	AI	27	80	1	AI	112	58	1	Self	13	82	Self	57	58
158	Self	73	8	0	Self	57	21	1	Self	39	95	Self	57	58
159	AI	17	108	0	AI	43	58	1	Self	33	20	Self	57	58
160	Self	88	93	0	Self	57	106	1	Self	84	88	Self	57	58
161	AI	83	54	1	AI	52	56	0	Self	71	75	Self	57	58
162	Self	108	30	0	Self	57	4	0	Self	0	72	Self	57	58
163	AI	75	36	1	AI	55	56	1	Self	111	4	Self	57	58
164	Self	17	33	0	Self	57	101	1	Self	107	9	Self	57	58
165	AI	17	3	1	AI	109	57	0	Self	19	6	Self	57	58
166	Self	108	40	1	Self	57	114	0	Self	66	30	Self	57	58
167	AI	3	68	0	AI	60	57	1	Self	109	107	Self	57	58
168	Self	74	98	0	Self	57	27	1	Self	80	57	Self	57	58
169	AI	71	74	0	AI	95	57	0	Self	81	72	Self	57	58
170	Self	15	60	1	Self	57	86	1	Self	112	6	Self	57	58
171	AI	99	72	1	AI	29	56	0	Self	54	31	Self	57	58
172	Self	42	44	1	Self	57	85	0	Self	32	68	Self	57	58
173	AI	48	13	0	AI	91	56	0	Self	72	73	Self	57	58
174	Self	23	62	1	Self	57	9	1	Self	47	99	Self	57	58
175	AI	99	48	1	AI	113	71	1	Self	14	100	Self	57	58
176	Self	96	41	0	Self	57	88	1	Self	28	114	Self	57	58
177	AI	17	40	0	AI	61	57	0	Self	8	14	Self	57	58
178	Self	44	47	0	Self	57	100	0	Self	58	13	Self	57	58
179	AI	41	42	1	AI	99	57	1	Self	52	101	Self	57	58
180	Self	93	65	1	Self	57	53	0	Self	112	79	Self	57	58
181	AI	53	31	0	AI	39	56	1	Self	7	97	Self	57	58
182	Self	113	87	0	Self	57	10	1	Self	5	104	Self	57	58
183	AI	46	36	1	AI	101	59	0	Self	4	16	Self	57	58
184	Self	6	111	0	Self	57	47	1	Self	66	65	Self	57	58
185	AI	72	109	1	AI	47	58	1	Self	66	41	Self	57	58
186	Self	112	114	0	Self	57	29	1	Self	32	14	Self	57	58
187	AI	32	68	1	AI	105	69	1	Self	8	55	Self	57	58
188	Self	54	9	1	Self	57	95	1	Self	96	73	Self	57	58
189	AI	40	53	1	AI	100	58	0	Self	80	40	Self	57	58
190	Self	94	29	0	Self	57	105	0	Self	75	9	Self	57	58
191	AI	99	97	1	AI	69	58	0	Self	99	16	Self	57	58
192	Self	13	66	0	Self	57	55	1	Self	6	112	Self	57	58
193	AI	29	80	1	AI	60	57	1	Self	97	100	Self	57	58
194	Self	43	0	0	Self	57	87	1	Self	41	83	Self	57	58
195	AI	18	6	1	AI	113	71	1	Self	16	21	Self	57	58
196	Self	69	113	1	Self	57	80	1	Self	59	29	Self	57	58
197	AI	109	87	1	AI	68	57	1	Self	97	61	Self	57	58
198	Self	27	81	1	Self	57	100	0	Self	86	97	Self	57	58
199	AI	71	47	0	AI	109	57	0	Self	98	13	Self	57	58
200	Self	29	6	0	Self	57	93	1	Self	87	108	Self	57	58

Winning Rate %: 47.5 Winning Rate %: 56.5

The winning rate has increased using opening book.

Report Question and Answers

Q1A. Describe your process for collecting statistics to build your opening book.

The opening book follow the same logic as the lessons, by building a table that maps [gamestate] = best action
Each action is selected randomly. There is no evaluation to determine which state action is better than another, except when the game ends with a winner.
There can be more than one action per state, the action that accumulates the most wins for the active player will be saved into the opening book.
This uses raw win counts, which are a poor statistic to estimate the value of an action, i have not used a better statistics method.
The large number of rounds i used, about 6 million, is the key of building the opening book, by simulate enough games.

Each round is a new isolation game, a new blank state, the chosen action for each game state is selected totally at random.
At the end, I have not manually used the centre cell 57 of the board as the 'possible' initial advantage, as i think it is not a definite advantage, i wanted completely random choices without human interference.
Also i have not used minimax or alpha beta pruning algorithms when building the opening book, as to my understanding, these algorithms will select its best move all the time.
Given a state, the algorithm will almost certain to select one same move action.
But randomly select an action give equal opportunity of each action to simulate the game and see if it is a good move.
(This is only my intuition, not proven!)

Q1B. How did you choose states to sample?

For the report requirement,
I only need to concentrate on the opening moves, that is the first two states in the game logs.
when player 1 is Self (CustomerPlayer), the self player can select the very first move.
when player 2 is Self (CustomerPlayer), the self player will respond to AI first opening move.

Q2A. And how did you perform rollouts to determine a winner?

Once the run_matches completed,
the winning rate above 50% means Self (CustomerPlayer) is the winner.
the winning rate below 50% means AI (Minimax depth=3) is the winner.
the winning rate of 50% means the current number of rounds is not enough to determine which player is better, consider increase number of rounds.
In most run matches, the winning rate never be right on 50%.

Notice: In my code, i have also used depth=3, as increasing the depth to 4 will naturally be better than depth of 3, i want to make sure the depth are the same for both AI and my code, so the algo winning performance can be measured.

Q2B. What opening moves does your book suggest are most effective on an empty board for player 1 and what is player 2's best reply?

The most effective move on an empty board for player 1 is 57.
Player 2's best reply is 58.

If Self player is player 1, the opening book always select 57, the centre cell, the first move for player 1.
If Self player is player 2, the opening book always select 58 as the reply if the first move is 57.
notice 58 is right next to the centre cell, west.

In Self VS AI minimax, AI player 2's reply to first move 57 is a random choice, because in get_action(), the first two moves are selected randomly.

Opening book Summary

Building the opening book

run_match.py
When running run_match.py, the project already provided code to save all the game play stats in matches.log in the root dir.
The info i need from matches.log for report requirements are the first two opening moves in each game.
but to extract the info, the state action position, if manually done i will need to search, copy and paste text line by line onto a spreadsheet. 100 round = 200 games, running 4 different mode combinations, this adds up to 800 copy and paste.
And if i make a mistake, then its possible redo the task all over again.
To solve this problem, i have written a simple program to extract just the first 2 moves of each game.
Program file: analysis_matches_logs.py
And i run this 4 times, each time after run_match.py completed.

$python run_match.py -r 100 (Self vs AI, not using opening book, flag in code)
$python analysis_matches_logs.py
copy and paste all text into spreadsheet

$python run_match.py -r 100 (Self vs AI, using opening book, flag in code)
$python analysis_matches_logs.py
copy and paste all text into spreadsheet

$python run_match.py -r 100 -o SELF (Self vs Self, not using opening book, flag in code)
$python analysis_matches_logs.py
copy and paste all text into spreadsheet

$python run_match.py -r 100 -o SELF (Self vs Self, using opening book, flag in code)
$python analysis_matches_logs.py
copy and paste all text into spreadsheet

Parameters tweaking - Rounds and Depth

I have run the code to build the opening book about 30 times, each with different combinations of number of rounds + depth.
Also, the code have adjust slightly between the builds, like adding centre cell 57, and taking it out and see the effects.
I have experimented with rounds from 10K to 50 million, and depth from 3 to 50.

Speed Performance

As i increases the number of depth, the time required to build the opening book grow exponentially.
To speed up the process, i have moved the opening book code to a new file, and use pypy3 to build the opening book, it is on average 5 times faster than normal python.
At one time, the opening book is built with depth of 5, with 50 millions round took 7 hours to complete using pypy3.
At the end, i have selected a lighter version of the opening book, built with 6 million rounds and depth of 4.
As the winning rate does not change much with difference in depth by 1, and it only took 332 seconds to built it using pypy3.

Testing

Each time the opening book is completed, I have tested it playing against the AI minimax with default depth of 3.

Winning Stats

During opening book testing, the run result are saved in matches.log, in there i can see what moves are chosen by the opening book for the first few moves, especially opening first move.
I have also used code to visualize the board when executing run_matches.py:
dbstate = DebugState.from_state(state)
print (dbstate)