Kate Do
4/3/22
CS 6340 Machine Learning

How Much Blackjack Until AI Can Count Cards

For this project we were asked to create a blackjack game with slightly modified rules for ease. We created our own policy before creating a Q-Learning process to optimize these policies and have a deeper understanding of stochastic processes, markov decision problems, and reinforcement learning.

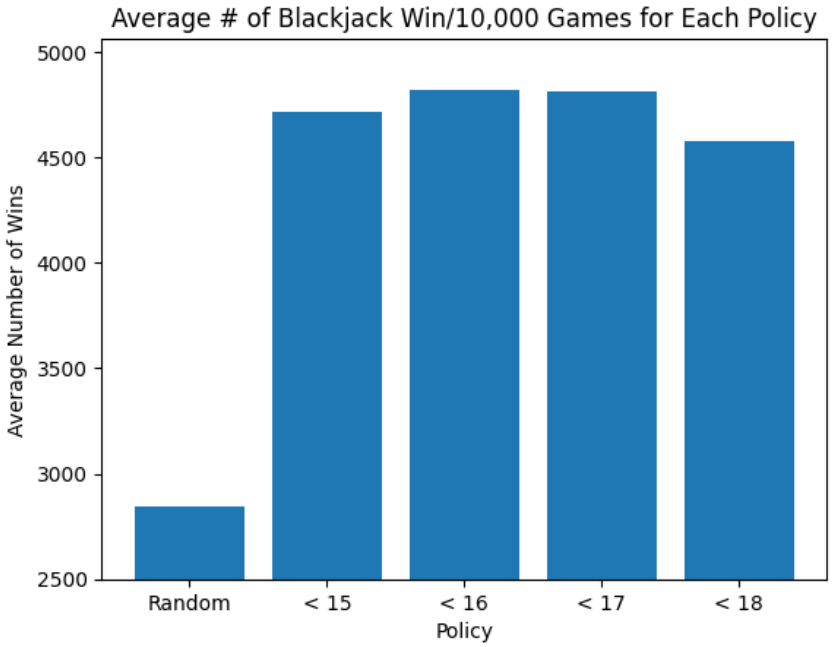
My Policy

Boiled down, my policy was following the state, or current total of all my cards.

→ State <= 15 : Hit | State > 16 : Stand

Which then prompted me to look at other nearby states to see if there was a better number to start “standing” at. And lead to 3 other action cases of 15, 17, and 18. This lead to the graph below. Over 10 loops containing 10 thousand games, I average the number of wins for each policy. As expected the random was very low, but the policies created were still below 50% winrate. In order of most effective was <16, <17, <15, <18. It’s not super surprising considering how risky it is to start hitting as the state goes up, but interesting that policy 15 and 17 weren’t as close as 16 and 17.

Kate Do
4/3/22
CS 6340 Machine Learning



Graph 1: Average # of Blackjack Wins/10,000 Games for Each Policy over 10 Iterations

Q-Learning Algorithm

When it came to the q-learning algorithm, there wasn't a specific iteration of policies that kept showing up. There was definitely a consensus of hitting up until 12, but the stands started to randomly pop up from there on. While it wasn't a significant amount I'd be interested in looking back and the games played to see what happened. The most consistent stands started around state 16 which ties directly back into the policy that I created earlier. Once played, the same winrate did start popping up which was around 50% of the time winning. So odd ones like policy 4 or 9 from the table below had lower win rates overall, yet not as terrible as random.

Kate Do
4/3/22
CS 6340 Machine Learning

Finished Optimizations with Their Policy

Game /State	0	1	2	3	4	8	9	10	11	12	13	14	15	16	17	18	19	# of S/State
Win	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	17
Lose	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	17
1	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
2	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
3	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
4	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
5	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
6	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
7	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
8	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
9	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
10	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
11	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
12	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	0
13	H	H	H	H	S	H	H	H	H	S	H	H	H	H	H	H	H	2
14	H	S	H	H	H	H	H	H	S	H	H	H	H	H	H	H	H	2
15	H	H	S	H	S	H	H	H	H	S	H	S	H	S	H	H	H	5
16	S	S	H	H	S	H	H	H	H	H	H	S	H	H	H	S	H	5
17	H	S	S	S	S	S	H	S	S	H	S	S	H	H	S	S	H	11
18	S	S	S	H	S	S	H	S	H	H	S	S	S	H	S	S	S	12
19	S	S	S	S	S	S	H	S	S	H	S	S	S	H	H	H	S	12
20	S	S	S	S	S	S	S	S	S	S	S	S	S	H	S	S	S	16
21	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	17
# S / Optimization	5	7	6	4	8	5	2	5	5	4	5	7	4	2	4	5	4	XXX

Table 1: Finished Optimizations with Their Policy ***

*** Please note I had to fix the table alignment by deleting a few columns, you'll notice columns 5, 6, and 7 missing from here, but not in the notebook. I also automatically subtracted the Win and Loss **stand** from the bottom most calculation row to see optimizations for the actual decision states.