Reinforcement Learning

Andrii Zakharchenko

March 2020

1 Possible space state representations

Note: I don't know if here was expected 3 completely different representations. But these representation describe my thought process when I came up with the first one and then iterated.

1.1 First representation

Intermediate states:

Value of player's hand (values from 4-21, total 18 state) $\times$ Value of dealer's one card (values from 2-11, total 10 sates)

Terminal States:

State that represents a situation when player has value of hand 22+.

Value of player's hand (values from 4-21, total 18 state) $\times$ Value of dealer's hand after he finished (4-22+, where 22+ represent that dealer is busted, 19).

1.2 Second representation

Similar to the first one, but we will also track whether player has ace: Value of player's hand $\times$ Value of dealer's first card $\times$ whether player has ace that he can use (true/false, 2 states)

1.3 Third representation

Similar to the first one, but terminal states may be: player is busted and dealer has finished playing.

2 Number of states in state space representations

2.1 First representation

$|S_1| = 18 \times 10 + 1 + 18 \times 19 = 532$

2.2 Second representation

$|S_2| = 18 \times 10 \times 2 + 1 + 18 \times 19 = 703$

2.3 Third representation

$|S_3| = 18 \times 10 \times 2 + 1 + 1 = 362$

3 Description of the chosen state space

I've chosen the third representation since it requires less states than all other and it also conveys more information that the first one.
Yes, it captures all information, we don't need to know ranks and suits of cards to make decision. Also, we don't need to know in what exactly terminal state we will end up, it doesn't matter whether player busted with 23 or 25.
Under assumption that we now probabilities of transitions we can use exact method to solve this since the state space is relatively small (e.g. value iteration has $O(|S|^2|A|)$, which is feasible in this case).
The goal here is to maximize rewards in long term, so I picked $\gamma = 0.9$ since bigger values of discount factor may force agent to value long-term gains. I chose number of games based on the idea that we need to see all possible states during training multiple times, also by plotting reward during training of SARSA I could see that the curve was flattening, so I determined that 500k iterations should be enough.

4 Agent testing

5 TD

First let's see how utility values of states of TD agent look like:

UTILITY IF AGENT DOESN'T HAVE ACE

Hand value	2	3	4	5	6	7	8	9	10	11
4	0.136	-0.328	-0.332	0.040	-0.131	-0.083	-0.300	-0.274	-0.300	-0.472
5	-0.147	-0.226	-0.185	-0.155	-0.082	-0.098	-0.104	-0.327	-0.329	-0.221
6	-0.077	-0.393	-0.208	-0.275	-0.134	-0.173	-0.316	-0.402	-0.394	-0.309
7	-0.269	-0.141	0.063	0.123	-0.092	-0.077	-0.187	-0.284	-0.278	-0.377
8	-0.193	-0.070	-0.041	-0.031	0.108	0.097	-0.022	-0.165	-0.237	-0.349
9	-0.034	-0.047	0.058	0.208	-0.009	0.359	0.116	-0.071	-0.171	-0.240
10	0.187	0.141	0.130	0.239	0.305	0.211	0.226	0.187	-0.030	-0.187
11	0.101	0.219	0.174	0.159	0.252	0.416	0.212	0.373	0.062	-0.136
12	-0.257	-0.277	-0.243	-0.130	-0.233	-0.183	-0.264	-0.463	-0.364	-0.403
13	-0.307	-0.248	-0.336	-0.357	-0.337	-0.261	-0.225	-0.431	-0.394	-0.578
14	-0.374	-0.407	-0.481	-0.383	-0.360	-0.336	-0.438	-0.446	-0.493	-0.559
15	-0.695	-0.651	-0.388	-0.369	-0.334	-0.397	-0.639	-0.567	-0.477	-0.496
16	-0.573	-0.568	-0.414	-0.474	-0.655	-0.251	-0.439	-0.670	-0.579	-0.578
17	-0.171	-0.008	-0.039	-0.147	-0.053	-0.141	-0.343	-0.352	-0.359	-0.436
18	0.081	0.149	0.145	0.093	0.567	0.585	0.082	-0.102	-0.229	-0.492
19	0.196	0.608	0.607	0.508	0.532	0.651	0.661	0.216	-0.072	-0.080
20	0.645	0.589	0.699	0.686	0.586	0.807	0.829	0.737	0.427	0.030
21	0.808	0.788	0.789	0.907	0.809	0.901	0.965	0.837	0.924	0.656

UTILITY IF AGENT HAS ACE

Hand value	2	3	4	5	6	7	8	9	10	11
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
12	0.030	-0.130	0.394	0.287	0.441	-0.012	0.015	0.100	-0.261	-0.334
13	-0.224	0.051	-0.168	0.326	-0.023	0.210	-0.130	0.118	-0.117	-0.330
14	-0.266	-0.092	-0.201	-0.144	-0.030	-0.098	-0.082	0.178	-0.070	-0.358
15	-0.268	-0.240	-0.151	0.078	-0.269	-0.169	0.090	0.321	-0.141	-0.478
16	-0.087	0.067	-0.108	-0.174	-0.165	0.280	-0.010	0.173	-0.098	-0.398
17	-0.263	-0.271	-0.219	0.250	-0.705	0.144	-0.515	-0.992	-0.593	-0.834
18	0.377	-0.376	0.669	0.799	0.263	0.675	-0.051	-0.002	-0.025	-0.499
19	0.250	0.246	0.731	0.544	0.627	0.790	0.705	0.276	0.077	-0.073
20	0.763	0.293	0.744	0.816	0.844	0.908	0.870	0.343	0.511	0.163
21	0.942	0.814	0.877	0.968	0.917	0.871	0.921	0.858	0.918	0.690

Figure 1: Utilities of states

So it seems that states that are more likely to win, as when player has 21 for example, they tend to be near one, which is expected. And for example states

15 and 16 have lowest values, again that is expected, since hitting when having e.g. 16 can be risky and lead to more loses.

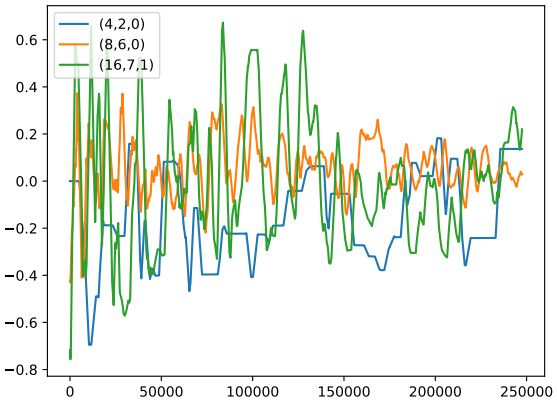


Figure 2: Convergence of utilities

On the plot above we can also see that utility values of some states converge to some steady values. Which also tells us that training is going fine.

6 SARSA

Let's firstly look at rewards during training of SARSA

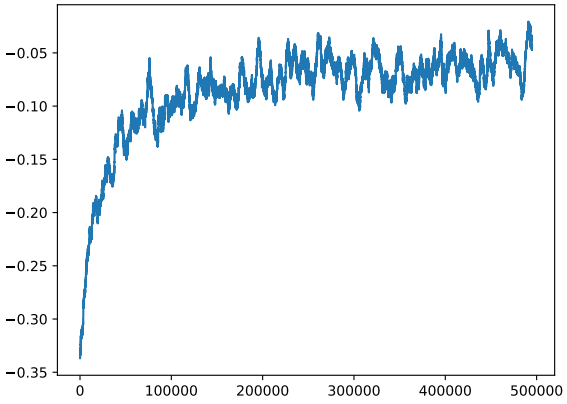
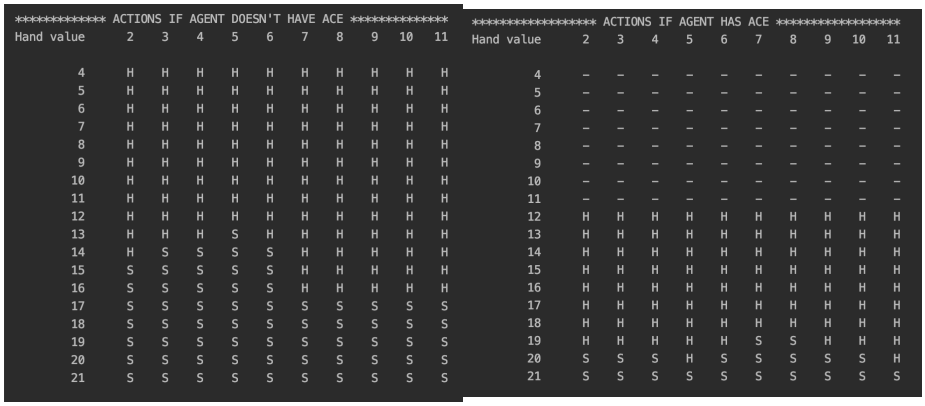


Figure 3: SARSA training (processed with moving average)

Here we can see that as training progresses agents starts to get better average rewards, which means he leans some efficient strategies. Also blackjack is a game where a player has disadvantage, so crossing the -0.1 is really good, since we approach to basically maximum what we can get.

Also I compared strategies of dealer, random agent and sarsa and I saw that my sarsa had average reward of -0.0609, dealer had -0.0824 and the random startegy was -0.394, they all played 50k games.

Also, we can compare strategies that learned my sarsa with optimal blackjack strategies



********** ACTIONS IF AGENT DOESN'T HAVE ACE **********

Hand value	2	3	4	5	6	7	8	9	10	11
4	H	H	H	H	H	H	H	H	H	H
5	H	H	H	H	H	H	H	H	H	H
6	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	H	H	H	H	H	H	H	H	H
10	H	H	H	H	H	H	H	H	H	H
11	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H
13	H	H	H	S	H	H	H	H	H	H
14	H	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	H	H	H	H
17	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S

********** ACTIONS IF AGENT HAS ACE **********

Hand value	2	3	4	5	6	7	8	9	10	11
4	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	-	-
12	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
17	H	H	H	H	H	H	H	H	H	H
18	H	H	H	H	H	H	H	H	H	H
19	H	H	H	H	H	S	S	H	H	H
20	S	S	S	H	S	S	S	S	S	H
21	S	S	S	S	S	S	S	S	S	S

Figure 4: SARSA learned actions

In situations without ace, my agent performs almost the same as optimal. But in situations when it has ace, its strategy is more off, it maybe be because hands with aces are more rare, hence it wasn't able to learn better policy.