SMU: Project 4

Michal Najman

State Representation

Let us define few sets needed.

D is a deck of 52 cards, $d \in D$ is the dealer's card, $H \subset D$ is a set of cards an agent holds in his hand.

In the following lines, the descriptions of various non-terminal states representation are given. To define terminal states, we only need to have a state for each possible outcome

$$T = \{Win, Lose, Tie\}$$

Naïve Representation

To start off, one could represent a state storing the full information. The state is as follows:

$$s = \langle d, H \rangle$$

This representation is rather exhaustive, making a set of states $S = 2^D$ meaning $|S| = 2^{52}$ which is intractable.

Reduced Representation

Note that in Black Jack only the value of a card is important (in the game version we play). Thus, the naïve representation is arguably redundant. In this representation, let us reduce card number by introducing 10 value groups $R = \{Ace, 2,3,4,5,6,7,8,9,10\}$, $n_i(H)$ is a counter stating the number of cards in H, that belong to the same value group $i \in R$, $g: D \longrightarrow R$.

$$s = \langle g(d), n_{Ace}(H), \dots, n_{10}(H) \rangle$$

$$|S| = |R| \cdot \prod_{i \in R} n_i(D) > 10^8$$

Although many of those states will never be visited, this is still intractable.

Value Representation

Notice that having a four and a five in hand is practically not distinguishable from having three threes as they have the same value. Of course, if an agent is playing counting cards strategy, it cannot afford to lose such a valuable information about the cards still being in a deck.

But let us now relax the game in such a way, the counting cards strategy is not feasible. Then we can only store the value the agent has in hand and the dealer's card. Function $v: 2^D \longrightarrow \mathbb{N}$ denotes the value of a set of cards, $H_{non-Ace}$ is a hand without Aces.

$$s = \langle g(d), v(H_{non-Ace}), n_{Ace}(H) \rangle$$

Note that we have to distinguish the aces as those have two values. Empirically, this is also the way a human player would build up his/her strategy.

$$|S| = |R| \cdot \max_{X \subset D_{non-Ace}} v(X) \cdot n_{Ace}(D) = 10 \cdot 376 \cdot 4 = 15{,}000$$

However, since we only care about states in which the hand is 21 or less, the actual number of states will be significantly smaller, roughly 2000.

In this work, Value Representation is used due to its small state spaces.

Is Value Representation MDP-compliant?

Naïve Representation (NR) is naturally MDP-compliant as the game itself is MDP and NR captures the full information. Since we designed Reduced Representation (RR) in such a way it does not lose information about the game, it is also MDP-compliant.

Now, note that the mapping from RR to Value Representation (VR) is surjective, i.e. for each state in RR there is only one state in VR, vice-versa: for each state in VR, there is a unique set of states in RR. This means that even though we lose information about the cards in the deck, we are still able to stay in which disjunctive set of RR states we are.

For that, the utility of a VR state is a convex combination of utility RR states that are mapped onto the VR state (proof is simple and utilizes transition probabilities and the above-mentioned fact).

For these reasons, we call VR states fully observable and, thus, MDP-compliant. Also, since those are non-terminal states their reward is always zero. In conclusion, VR does not influence the exact methods.

Results

Number of epochs = 100,000.

In the TD agent, the following alpha function is used:

$$\alpha(s) = \frac{10}{9 + n(s)}$$

In the SARSA Agent, having tried several values, these functions have been used:

$$\alpha(s, a) = \frac{2}{1 + n(s, a)}$$

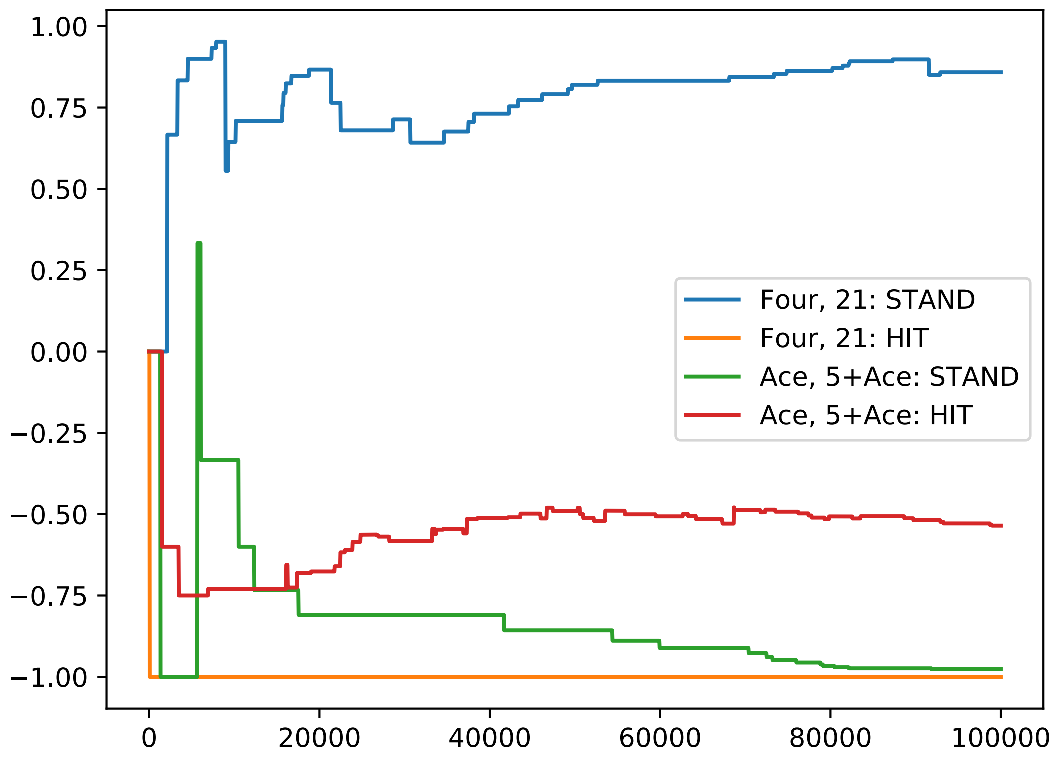
$$\varepsilon(s) = \frac{20}{19 + n(s)}$$

In the table below training utilities are given.

Agent	***Average Training Reward***
Random Agent	-0.39
Dealer Agent	**-0.08**
TD Agent	-0.08
SARSA Agent	-0.20

Note that the average reward of the SARSA Agent is worse than the Dealer Agent's which is counterintuitive. I assume it is either caused by incorrect constants setting on my side or this is actually expected behaviour which I cannot explain, sadly.

In the picture below, utility functions of specific states and actions of the SARSA Agent are depicted. The hands proposed in the task are chosen. Notice, that it takes significantly more time to find out which of the two actions STAND or HIT is better given the hand Ace and Five while a dealer has Ace than for the 21-in-hand state.



Since a dealer is advantageous over the agent (when they both go bust, the dealer wins), expected utilities of an agent are negative for most states.

SARSA and TD agents both converged.

Lastly, the Wikipedia strategy is roughly followed. A csv file final_strategy.csv is attached to the report, giving the table of actions that shall be played depending on the state of the game. This strategy was derived using SARSA in 1 mil. epochs. (Having more than 2 aces is considered anomalous and is not covered in the table.)