Assignment 6: Tabular learning and deep RL

Question 1: Jacky

Consider a simplified Blackjack variant, *Jacky*, defined as follows:

- The game starts with a sum $S = 0$.
- At each step, the agent can:
  - **Hit**: Draw a card uniformly at random from $\{1, \ldots, 10\}$ and add it to S.
  - **Stay**: Stop drawing cards.
- If at any point $S > 21$, the agent busts and the episode ends with a reward of 0.
- If the agent chooses to stay at a sum $S \leq 21$, the reward for that episode is simply S.

Task: Implement a Monte Carlo (MC) learning algorithm to find a good policy for Jacky. Use a Q-table representation where the state is just the current sum S. After training, run the learned policy for 100 episodes and report the average reward.

Question 2: Jacky with Randomized Rewards

Now, we modify the Jacky game to have random reward structure:

- The rules are as before, except instead of having $r_i = i$ for $i \in \{1, \ldots, 21\}$, we draw a random reward r_i from $\{1, \ldots, 21\}$ uniformly at the start of each episode.
- Thus, at the beginning of each episode, we sample:

$$(r_1, r_2, \ldots, r_{21}), \quad r_i \sim \text{Uniform}\{1, \ldots, 21\}.$$

- The agent **knows** these rewards before drawing any cards.
- The state now includes both the current sum S and the entire reward vector $(r_1, \ldots, r_{21})$.

Task:

1. Define a neural network whose input is the state $(S, r_1, \ldots, r_{21})$ and whose output is the Q-value estimates for the 'hit' action. Note that there is no need to approximate the value of 'stay'.
2. Consider how best to encode or represent the state. You might find that 1-hot encoding is a good choice for S.
3. Train the agent.
4. After training, evaluate the agent by running it for 100 episodes with newly drawn reward vectors each time, and report the average reward.