GSEM, University of Geneva — Spring Semester 2024
Advanced Topics in Machine Learning (S411021) — **RL Project**
Prof. Sebastian Engelke — Assistants: Olivier Pasche, Manuel Hentschel

RL Project: Dice Game

1 The Game

In this project, we consider the following "Dice Game". The objective of the game is to make money by outscoring the dealer or by rolling doubles. Each round, the player starts with a score of zero. Since this is an episodic task, discounting is not necessary.
Each turn, the player has to choose between rolling their dice, or betting money on the dealer's roll. If they decide to roll themselves, they can either roll one or two dice (simultaneously), which costs them CHF 1 for one dice, and CHF 2 for two dice. The number(s) shown by the dice are added to the player's score. If the player rolls a double (i.e., two dice showing the same numbers), they get an immediate bonus payout of CHF 10, independent of the rest of the game or their score. If the player reaches a score of 31 or more, they lose the round and have to pay CHF 10.
If the player chooses to bet on the dealer's roll, they have to specify a bet-multiplicator of 1, 2, or 3. The dealer then rolls their dice and the player is paid/has to pay according to the formula

$$(\text{“Player Score”} - \text{“Dealer Dice Result”}) \cdot \text{“Bet-Multiplicator”},$$

and the round is over.
The player has two identical dice, showing the numbers $\{1, 2, 3, 4, 5, 6\}$. The dealer has one dice, showing the numbers $\{25, 26, 27, 28, 29, 30\}$. All three dice are weighted, such that their highest number has twice the probability of each of the smaller numbers.

2 Tasks

Throughout, the state space should have (at most) one terminal state.

1. Considering the game as a Markov decision process, identify the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the reward set $\mathcal{R}$.
2. Implement a Python class that represents the game as a reinforcement learning task. The class should contain all the information about the game state, and should provide a "step" method that takes an action as input and returns the reward and next state, as well as a "reset" method that resets the game to its initial state.
3. Using dynamic programming, compute the value functions under the following policies.
   - "R1": The player always rolls a single dice.
   - "R2": The player always rolls both dice.
   - "RR": If the player's score is strictly smaller than 20, they roll either one or two dice with equal probability. Otherwise, they choose one of the three bet-multiplicators uniformly at random.

   Explain the results and represent them graphically.
4. Find the optimal policy using dynamic programming. Represent the action-value function under the optimal policy graphically. Explain the results and compare them to those of the previous task.

BONUS. Use the class you implemented in the first task for the following Monte Carlo simulation: estimate the value of the initial state under each of the policies from the previous tasks ("R1", "R2", "RR", "Optimal"). Illustrate the results and compare them to the results of the previous tasks.

3 Deliverables

You need to create a Jupyter Notebook containing your solutions to the tasks above. Text answers and plots must be included in the notebook. Implementations of classes and functions can also be in (one or more) separate files that you import into the notebook.
You need to submit the Jupyter Notebook, a PDF export of the notebook, and any Python files you use in the notebook. Rerunning the Jupyter Notebook should produce the same results as the PDF export (up to rounding errors and random differences in the Monte Carlo simulation).
Submission deadline is the **27.03.2024, 08:00h** (before the seminar!), via Moodle.

4 Grading

Grading will be on a scale of 20 points (out of 100 possible points for all projects/exams in this class), based on the following criteria:

- correctness of computations and numerical results;
- correctness and quality of text answers;
- presentation of results, including plots, structure of the notebook, etc.;
- correctness and quality of code: the code must run without errors, be easy to read and understand, and should be reasonably efficient.

5 Rules

1. This is an individual project. While you can of course discuss with other students, your code and the report must be written individually.
2. No cheating of any kind.
3. You must use the Python programming language.
4. You can use common Python libraries (e.g., numpy, matplotlib) but you are not allowed to use any machine learning or optimization libraries. If you are unsure, ask the assistants.