Planning in MDPs

Description

You are given an N-sided die, along with a corresponding Boolean mask vector, `is_bad_side` (i.e., a vector of ones and zeros). You can assume that $1 < N \leqslant 30$, and the vector `is_bad_side` is also of size N and 1 indexed (since there is no 0 side on the die). The game of DieN is played as follows:

1. You start with 0 dollars.
2. At any time, you have the option to roll the die or to quit the game.
   A. **ROLL**:
      a. If you roll a number not in `is_bad_side`, you receive that many dollars (e.g., if you roll the number 2 and 2 is not a bad side -- meaning the second element of the vector `is_bad_side` is 0, then you receive 2 dollars). Repeat step 2.
      b. If you roll a number in `is_bad_side`, then you lose all the money obtained in previous rolls and the game ends.
   B. **QUIT**:
      a. You keep all the money gained from previous rolls and the game ends.

Procedure

- You will implement your solution using the `solve()` method in the code below.

- Your return value should be the number of dollars you expect to win for a specific value of `is_bad_side`, if you follow an optimal policy. That is, what is the value of the optimal state-value function for the initial state of the game (starting with 0 dollars)? Your answer must be correct to 3 decimal places, truncated (e.g., 3.14159265 becomes 3.141).

- To solve this problem, you will need to determine an optimal policy for the game of DieN, given a particular configuration of the die. As you will see, the action that is optimal will depend on your current bankroll (i.e., how much money you've won so far).

- You can try solving this problem by creating an MDP of the game (states, actions, transition function, reward function, and assume a discount rate of Υ=1) and then calculating the optimal state-value function.

Resources

The concepts explored in this homework are covered by:

- Chapter 3 (3.6 Optimal Policies and Optimal Value Functions) and Chapter 4 (4.3-4.4 Policy Iteration, Value Iteration) of http://incompleteideas.net/book/the-book-2nd.html (http://incompleteideas.net/book/the-book-2nd.html)
- Chapters 1-2 of 'Algorithms for Sequential Decision Making', M. Littman, 1996

Submission

- The due date is indicated on the Syllabus page for this assignment.
- Use the template below to implement your code. We have also provided some test cases for you.
- Please use *python 3.6.x* and *numpy==1.18.0*, and you can use any core library (i.e., anything in the Python standard library). No other library can be used.