Project #1

Amir Sadovnik

ECE 414/517: Reinforcement Learning in Artificial Intelligence (Fall 2021)

---

1 Overview

The goal of this project is to get hands on experience with writing code for a Markov Decision Process (MDP) in a manner that can be solved using dynamic programming (DP). The assignment comprises of writing a Python-based simulation, and then writing a report which analyzes and explains the results. The project should be done individually.

2 Problem Description

Your goal is to find the optimal policy for a late student arriving at class as shown in the diagram below. Since the student arrives late in class he must choose one of the seats in the right most column (which are all empty). The goal of the student is to sit as far up as front as possible while reducing the probability of getting sick.

1. There are N empty seats at the far right of the class (as shown in the diagram).
2. The student enters the class from the back of the room.
3. When the student walks by the spot they can choose to either take the seat or move on to the next seat (the student cannot walk backwards).
4. When at a row the student can observe the seat next to the empty one and asses if it is empty, has a student with a mask, or has a non masked student.
5. When making a decision the student is not able to see the situation in the next rows (cannot tell if anyone is sitting in the next row).
6. The student would like to sit as far up as front as possible.
7. If the student is not able to find a spot he will need to exit the class from the front door which will be very embarrassing.

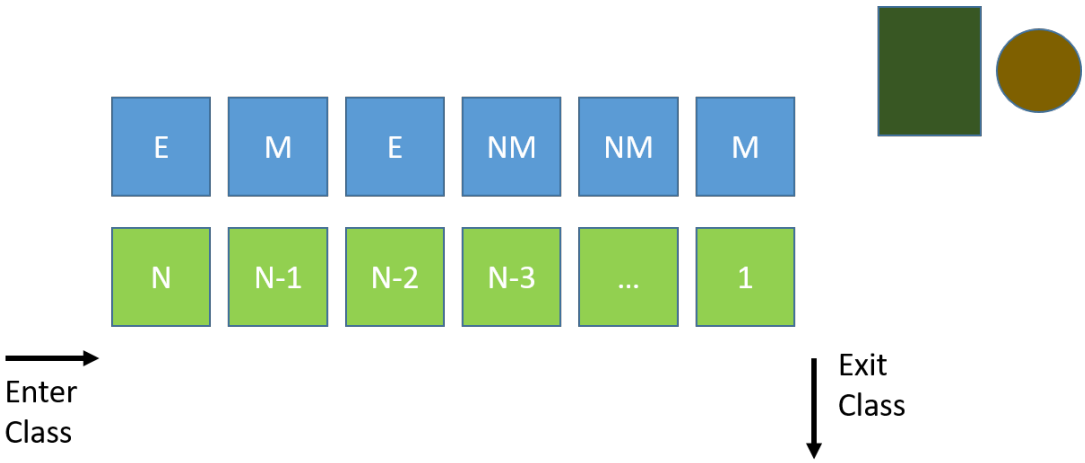

Figure 1: A diagram of the problem. Note that the agent enters the class from the back and at each row gets to choose if to sit or not. The agent can only see the seat next to the empty one when they stand in that row (showing all rows just for illustration).

You are tasked with writing a computer program which will guide the student's behavior. More specifically you are asked to model the problem as an MDP, and find the optimal policy using DP. You can make the following assumptions:

1. There is a probability of a student being seated at each neighboring seat p_{taken}, and given that a student is seated, that the student is masked p_{mask}.
2. Depending on who the student sits next to (no one, masked, non-masked) there is a different probability they would get sick ($p_{sick-empty}, p_{sick-masked}, p_{sick-nonmasked}$ respectively) .
3. Your learning ability is linear in the row you decide to sit at. That is you can measure it by $R_{learn} \times row$, where row is the row you decide to sit at.
4. However, getting sick is bad and should be rewarded negatively with R_{sick}.
5. If leaving the classroom without sitting you will not get sick, but that should be rewarded negatively as well since there is no learning R_{exit}.

3 Example Setup

For the report you will be required to show results on this specific setup:

1. $N = 12$ rows in the class (rows are numbered 1-12).
2. The learning benefit gained from sitting in each row is $(12 - rownumber) \times 2$
3. Getting sick (R_{sick}) and missing class (R_{exit}) both are -100.
4. $p_{taken} = 0.5$, $p_{mask} = 0.5$,$p_{sick-empty} = 0.01$, $p_{sick-masked} = 0.1$,$p_{sick-nonmasked} = 0.5$

4 Coding

Write a program which solves the problem using dynamic programming. You should be able to run your program with the following parameters:

`python3 myProgram.py` N p_{taken} p_{mask} $p_{sick-empty}$ $p_{sick-masked}$ $p_{sick-nonmasked}$ R_{sick} R_{exit} R_{learn}

Your code should have at least 3 functions: one which calculates the expected return given a state and action and current state value function, one which performs value iteration (by calling the previous function), and one which calculates the policy given a value function. The output of your program should be the value function and the policy.

You are required to write your code in python3. The only external library you are allowed to use in numpy (and any graphing libraries).

5 Report

Your report should include the following:

1. A short introduction to the problem you are trying to solve.
2. A description of how you framed the problem as an MDP (including a diagram of your MDP).
3. A description of your code design and your choice of data structures.
4. Correctness Results - in this section you should present the results of your algorithm for the example setup in Sec. 3.
   (a) Show the optimal value function for this setup and describe why it makes sense (specifically explain the results from the front row since those should be easy to explain).
   (b) Show the optimal policy for this setup and describe why it makes sense.
5. Results Analysis - Come up with a few interesting questions (at least 2) and plot the results for those questions. These can be things like examining runtime as a function of parameters, comparing value iteration and policy iteration, effects of the parameters on the policy, etc.
6. Conclusion - What have you achieved in this project? What have you learned?
7. References - If necessary.

6 Submission

You are required to submit one zip file with both he report and your code.