CMPSCI 687 Homework 1 - Fall 2023

Due **September 30, 2023**, 11:55pm Eastern Time

Note: we have added comments, below, to clarify some of the problems/questions in this homework. All parts of this document that were changed with respect to the original one are shown in blue.

1 Instructions

This homework assignment consists of a written portion and a programming portion. While you may discuss problems with your peers (e.g., to discuss high-level approaches), you must answer the questions on your own. In your submission, do explicitly list all students with whom you discussed this assignment. Submissions must be typed (handwritten and scanned submissions will not be accepted). You must use LaTeX. The assignment should be submitted on Gradescope as PDF with marked answers via the Gradescope interface. The source code should be submitted via the Gradescope programming assignment as a .zip file. Include with your source code instructions for how to run your code. You **must** use Python 3 for your homework code. You may not use any reinforcement learning or machine learning specific libraries in your code, e.g., TensorFlow, PyTorch, or scikit-learn. You *may* use libraries like numpy and matplotlib, though. The automated system will not accept assignments after 11:55pm on September 30. The tex file for this homework can be found here.

2 Hints and Probability Review

- **Write Probabilities of Events:** In some of the probability hints below that are not specific to RL, we use expressions like $\Pr(a|b)$, where a and b are events. Remember that in the RL notation used for this class, the values of $\Pr(s_0)$, $\Pr(a_0)$, $\Pr(A_0)$, or $\Pr(A_0|S_0)$ are all undefined, since those are simply states, actions, or random variables (not events). Instead, we ***must*** write about the probabilities of events. For example: $\Pr(A_0 = a_0)$ or $\Pr(A_0 = a_0|S_0 = s_0)$.

- **Bayes' Theorem:** $\Pr(a|b) = \frac{\Pr(b|a)\Pr(a)}{\Pr(b)}$. This is useful for dealing with conditional probabilities $\Pr(a|b)$ if the event a occurs *before* event b. For example, it is often difficult to work with an expression like $\Pr(S_0 = s_0|A_0 = a_0)$, because the agent *first* observes the current state, S_0, and only afterwards selects an action, A_0; in this case, it is much easier to deal with the 3 terms in $\frac{\Pr(A_0=a_0|S_0=s_0)\Pr(S_0=s_0)}{\Pr(A_0=a_0)}$.

- **The law of total probability:** For event a, and a set of events $\mathcal{B}$,

$$\Pr(a) = \sum_{b\in\mathcal{B}} \Pr(b)\Pr(a|b).$$

  See the example below for several useful applications of this property.

- **"Extra" given terms:** Remember that when applying laws of probability, any "extra" given terms stay in the result. For example, applying the law of total probability:

$$\Pr(a|c,d) = \sum_{b\in\mathcal{B}} \Pr(b|c,d)\Pr(a|c,d,b).$$

- **Conditional Probabilities - Useful property #1:** If you need to move terms from the "right-hand side" of a conditional probability to the "left-hand side", you can use the following identity:
  $\Pr(a|b,c) = \frac{\Pr(a,b|c)}{\Pr(b|c)}$

- **Conditional Probabilities - Useful property #2:** If you need to move terms from the "left-hand side" of a conditional probability to the "right-hand side", you can use the following identity:
  $\Pr(a,b|c) = \Pr(a|b,c)P(b|c)$.

- **Expected Values:** The expected value of a random variable X with possible outcomes in $\mathcal{X}$ is

$$\mathbb{E}[X] = \sum_{x\in\mathcal{X}} x\Pr(X = x).$$

- **Conditional Expected Values:** The expected value of a random variable X with possible outcomes in $\mathcal{X}$, conditioned on an event $A = a$, is

$$\mathbb{E}[X \mid A = a] = \sum_{x \in \mathcal{X}} x \Pr(X = x \mid A = a).$$

- **Example problem:** The probability that the state at time $t = 1$ is $s \in \mathcal{S}$.

$$\begin{aligned}
\Pr(S_1 = s) &= \sum_{s_0 \in \mathcal{S}} \Pr(S_0 = s_0) \Pr(S_1 = s | S_0 = s_0) && (1)\\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \Pr(S_1 = s | S_0 = s_0) && (2)\\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \Pr(A_0 = a_0 | S_0 = s_0) && (3)\\
&\quad \times \Pr(S_1 = s | S_0 = s_0, A_0 = a_0) && (4)\\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0)\, p(s_0, a_0, s). && (5)
\end{aligned}$$

Part One: Written (55 Points Total)

1. ***(Your grade will be a zero on this assignment if this question is not answered correctly)*** *Read the class syllabus carefully, including the academic honesty policy. To affirm that you have read the syllabus, type your name as the answer to this problem.*

   Answer: Ojas Jeetendra Raundale.

2. (**14 Points**) Given an MDP $M = (\mathcal{S}, \mathcal{A}, p, R, d_0, \gamma)$ and a fixed policy, π, the probability that the action at time $t = 0$ is $a \in \mathcal{A}$ is:

$$\Pr(A_0 = a) = \sum_{s \in \mathcal{S}} d_0(s) \pi(s, a). \tag{6}$$

   Write similar expressions (using only $\mathcal{S}, \mathcal{A}, p, R, d_0, \gamma$, and π) for the following problems.

   Important:

   - *Assume, below, that the reward function will be in the form $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. That is, the reward at time t depends only on the state at time t and action at time t.*
   - *All solutions below need to be derived from "first principles": you should repeatedly apply definitions and properties of probability distributions such as the ones discussed in Section 2, as well as the Markov Property (when appropriate), and then replace the relevant quantities with their corresponding definitions in RL (e.g., you can substitute $\Pr(A_0 = a | S_0 = s)$ with $\pi(s, a)$).*
   - *Remember that the Markov Property allows you to ignore history information, prior to time t, if you know S_t (that is, if the probability term is conditioned on S_t). It does not allow you to ignore variables associated with time t or any future times ($t + 1$, $t + 2$, etc). For instance:*

$$\Pr(S_1 = s_1 | A_1 = a_1, S_0 = s_0) \neq \Pr(S_1 = s_1 | S_0 = s_0)$$

   and

$$\Pr(S_2 = s_2 | S_4 = s_4, S_1 = s_1) \neq \Pr(S_2 = s_2 | S_1 = s_1).$$

- *When writing the final answers to the problems below (2a-2d), please reorganize your terms and summations in "temporal" order. For instance, instead of presenting your final answer as*

$$\sum_{s_1} p(s_1, a_1, s_2)\pi(s_1, a_1) \sum_{a_0} p(s_0, a_0, s_1)\pi(s_0, a_0)$$

rewrite it as follows:

$$\sum_{a_0} \pi(s_0, a_0) \sum_{s_1} p(s_0, a_0, a_1)\pi(s_1, a_1)p(s_1, a_1, s_2).$$

Problems:

- **(Question 2a. 3 Points)** What is the probability that the state at time $t = 2$ is s_2 given that the state at time $t = 0$ is s_0?
  Answer:

  $\Pr(S_2 = s_2|S_0 = s_0) = ?$

  Adding extra term a_0 to the expression

$$\begin{aligned}
&= \sum_{a_0 \in \mathcal{A}} Pr(S_2 = s_2|A_0 = a_0, S_0 = s_0)Pr(A_0 = a_0|S_0 = s_0) \\
&= \sum_{a_0 \in \mathcal{A}} Pr(S_2 = s_2|A_0 = a_0, S_0 = s_0)\pi(s_0, a_0) \\
&= \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} \Pr(S_2 = s_2|S_1 = s_1, A_0 = a_0, S_0 = s_0) \Pr(S_1 = s_1|A_0 = a_0, S_0 = s_0)
\end{aligned}$$

  Replacing with transition function p.

$$= \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} \Pr(S_2 = s_2|S_1 = s_1, A_0 = a_0, S_0 = s_0)p(s_0, a_0, s_1)$$

  Markov Principle and Rearranging.

$$= \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} p(s_0, a_0, s_1) \Pr(S_2 = s_2|S_1 = s_1)$$

  Adding extra term a_1 to the expression.

$$= \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} p(s_0, a_0, s_1) \sum_{a_1 \in \mathcal{A}} \Pr(A_1 = a_1|S_1 = s_1) \Pr(S_2 = s_2|A_1 = a_1, S_1 = s_1)$$

  Replacing with policy π and transition function p.

$$= \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} p(s_0, a_0, s_1) \sum_{a_1 \in \mathcal{A}} \pi(s_1, a_1)\, p(s_1, a_1, s_2) \tag{7}$$

  Equation 7 is the Answer.

- **(Question 2b. 3 Points)** What is the probability that the action at time $t = 1$ is a_1?
  Answer:

$$
\begin{aligned}
&\Pr(A_1 = a_1) = \ ? \\
&= \sum_{s_0 \in \mathcal{S}} \Pr(S_0 = s_0) \Pr(A_1 = a_1 | S_0 = s_0) \\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \Pr(A_1 = a_1 | S_0 = s_0) \\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \Pr(A_0 = a_0 | S_0 = s_0) \Pr(A1 = a_1 | A_0 = a_0, S_0 = s_0) \\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \Pr(A1 = a_1 | A_0 = a_0, S_0 = s_0) \\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} \Pr(S1 = s_1 | A_0 = a_0, S_0 = s_0) \Pr(A1 = a_1 | S1 = s_1, A_0 = a_0, S_0 = s_0) \\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} p(s_0, a_0, s_1) \Pr(A1 = a_1 | S1 = s_1, A_0 = a_0, S_0 = s_0) \\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} p(s_0, a_0, s_1) \Pr(A1 = a_1 | S1 = s_1) \\
&= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} p(s_0, a_0, s_1) \pi(s_1, a_1) \qquad (8)
\end{aligned}
$$

  Equation 8 is the final answer.

- **(Question 2c. 4 Points)** What is the expected reward at time $t = 5$ given that the action at time $t = 4$ is a_4 and the state at time $t = 3$ is s_3?
  Answer:

$$\mathbb{E}[R_5 | A_4 = a_4, S_3 = s_3] = ?$$

$$= \sum_r r \Pr(R_5 = r | A_4 = a_4, S_3 = s_3)$$

Total probability on $S_4 = s_4$

$$= \sum_r r \sum_{s_4 \in \mathcal{S}} \Pr(S_4 = s_4 | A_4 = a_4, S_3 = s_3) \Pr(R_5 = r | A_4 = a_4, S_4 = s_4, S_3 = s_3)$$

Markov Property

$$= \sum_r r \sum_{s_4 \in \mathcal{S}} \Pr(S_4 = s_4 | A_4 = a_4, S_3 = s_3) \Pr(R_5 = r | A_4 = a_4, S_4 = s_4)$$

Using $\Pr(A|B,C) = \frac{\Pr(A,B,C)}{\Pr(B,C)} = \frac{\Pr(A,B,C)}{\Pr(B,C)} \times \frac{\Pr(A,C)}{\Pr(A,C)} \times \frac{\Pr(C)}{\Pr(C)} = \Pr(B|A,C) \times \Pr(A|C) \times \frac{1}{\Pr(B|C)}$

$$= \sum_r r \sum_{s_4 \in \mathcal{S}} \Pr(A_4 = a_4 | S_4 = s_4, S_3 = s_3) \Pr(S_4 = s_4 | S_3 = s_3) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)} \Pr(R_5 = r | A_4 = a_4, S_4 = s_4)$$

Markov Property

$$= \sum_r r \sum_{s_4 \in \mathcal{S}} \Pr(A_4 = a_4 | S_4 = s_4) \Pr(S_4 = s_4 | S_3 = s_3) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)} \Pr(R_5 = r | A_4 = a_4, S_4 = s_4)$$

$$= \sum_r r \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) \Pr(S_4 = s_4 | S_3 = s_3) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)} \Pr(R_5 = r | A_4 = a_4, S_4 = s_4)$$

$$= \sum_r r \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) \sum_{a_3 \in \mathcal{A}} \Pr(A_3 = a_3 | S_3 = s_3) \Pr(S_4 = s_4 | A_3 = a_3, S_3 = s_3)$$
$$\times \Pr(R_5 = r | A_4 = a_4, S_4 = s_4) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$

$$= \sum_r r \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) p(s_3, a_3, s_4) \Pr(R_5 = r | A_4 = a_4, S_4 = s_4) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$

$$= \sum_r r \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \Pr(R_5 = r | A_4 = a_4, S_4 = s_4) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$

$$= \sum_r r \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$
$$\times \sum_{s_5 \in \mathcal{S}} \Pr(S_5 = s_5 | A_4 = a_4, S_4 = s_4) \Pr(R_5 = r | S_5 = s_5, A_4 = a_4, S_4 = s_4)$$

Markov Property and Transition Function p

$$= \sum_r r \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$
$$\times \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \Pr(R_5 = r | S_5 = s_5)$$

$$= \sum_r r \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$
$$\times \Pr(R_5 = r | S_5 = s_5)$$

$$= \sum_r r \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$
$$\times \sum_{a_5 \in \mathcal{A}} \Pr(A_5 = a_5 | S_5 = s_5) \Pr(R_5 = r | A_5 = a_5, S_5 = s_5)$$

$$= \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$
$$\times \sum_{a_5 \in \mathcal{A}} \pi(s_5, a_5) \sum_r r \Pr(R_5 = r | A_5 = a_5, S_5 = s_5)$$

Reward function R and rearranging

$$= \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \sum_{a_5 \in \mathcal{A}} \pi(s_5, a_5) R(s_5, a_5)$$
$$\times \frac{1}{\Pr(A_4 = a_4 | S_3 = s_3)}$$

$$= \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \sum_{a_5 \in \mathcal{A}} \pi(s_5, a_5) R(s_5, a_5)$$
$$\times \frac{1}{\sum_{s'_4 \in \mathcal{S}} \Pr(S_4 = s'_4 | S_3 = s_3) \Pr(A_4 = a_4 | S_4 = s'_4, S_3 = s_3)}$$

Markov Property and rearranging

$$= \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \sum_{a_5 \in \mathcal{A}} \pi(s_5, a_5)\, R(s_5, a_5)$$
$$\times \frac{1}{\sum_{s'_4 \in \mathcal{S}} \Pr(A_4 = a_4 | S_4 = s'_4) \Pr(S_4 = s'_4 | S_3 = s_3)}$$
$$= \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \sum_{a_5 \in \mathcal{A}} \pi(s_5, a_5)\, R(s_5, a_5)$$
$$\times \frac{1}{\sum_{s'_4 \in \mathcal{S}} \Pr(A_4 = a_4 | S_4 = s'_4) \sum_{a'_3 \in \mathcal{A}} \Pr(A_3 = a'_3 | S_3 = s_3) \Pr(S_4 = s'_4 | A_3 = a'_3, S_3 = s_3)}$$
$$= \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} \pi(s_4, a_4) p(s_3, a_3, s_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \sum_{a_5 \in \mathcal{A}} \pi(s_5, a_5)\, R(s_5, a_5)$$
$$\times \frac{1}{\sum_{s'_4 \in \mathcal{S}} \pi(s'_4, a_4) \sum_{a'_3 \in \mathcal{A}} \pi(s_3, a'_3) p(s_3, a'_3, s'_4)}$$

$$= \sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} p(s_3, a_3, s_4) \pi(s_4, a_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \sum_{a_5 \in \mathcal{A}} \pi(s_5, a_5)\, R(s_5, a_5)$$
$$\times \frac{1}{\sum_{a'_3 \in \mathcal{A}} \pi(s_3, a'_3) \sum_{s'_4 \in \mathcal{S}} p(s_3, a'_3, s'_4) \pi(s'_4, a_4)}$$

(OR)

$$= \frac{\sum_{a_3 \in \mathcal{A}} \pi(s_3, a_3) \sum_{s_4 \in \mathcal{S}} p(s_3, a_3, s_4) \pi(s_4, a_4) \sum_{s_5 \in \mathcal{S}} p(s_4, a_4, s_5) \sum_{a_5 \in \mathcal{A}} \pi(s_5, a_5)\, R(s_5, a_5)}{\sum_{a'_3 \in \mathcal{A}} \pi(s_3, a'_3) \sum_{s'_4 \in \mathcal{S}} p(s_3, a'_3, s'_4) \pi(s'_4, a_4)} \tag{9}$$

Equation 9 is the final answer.

- **(Question 2d. 4 Points)** Based on the expression you obtained in question **Q2b**, compute what is $\Pr(A_{t+1} = \text{AD})$ in the 687-Gridworld. Assume that the agent starts in state $S_t = s_1$ and that it follows a policy π described in the relevant states of Figure 1. In particular, $\pi(s_1, AR) = 0.2$, $\pi(s_1, AD) = 0.8$, $\pi(s2, AR) = 0.4$, $\pi(s2, AD) = 0.6$, $\pi(s_6, AR) = 0.7$, and $\pi(s_6, AD) = 0.3$. Show every step in your derivation.

  Answer:

  From Answer 2b (8),

$$\Pr(A_1 = a_1) = \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathcal{S}} p(s_0, a_0, s_1)\, \pi(s_1, a_1)$$

  In this question $A_1 = A_{t+1}, a_1 = AD, s_0 \in \{s_1\}, s_1 \in \{s_1, s_2, s_6\}, A = \{AD, AR, AL, AU\}$.
  Which means,
  $\sum_{s_0 \in \{s1\}} d_0(s_0) = 1$

  Substituting all the other values according to this question, we get:

$$Pr(A_{t+1} = AD) = \sum_{a_0 \in \{AD, AR, AL, AU\}} \pi(s_1, a_0) \sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, a_0, s_1')\, \pi(s_1', AD)$$

$$\begin{aligned}
&= \pi(s_1, AD) \sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, AD, s_1')\, \pi(s_1', AD) \\
&+ \pi(s_1, AR) \sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, AR, s_1')\, \pi(s_1', AD) \\
&+ \pi(s_1, AU) \sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, AU, s_1')\, \pi(s_1', AD) \\
&+ \pi(s_1, AL) \sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, AL, s_1')\, \pi(s_1', AD) \\
&= 0.8 \sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, AD, s_1')\, \pi(s_1', AD) \\
&+ 0.2 \sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, AR, s_1')\, \pi(s_1', AD) \\
&+ 0 \\
&+ 0 \\
&= 0.8(p(s_1, AD, s_1)\, \pi(s_1, AD) + p(s_1, AD, s_2)\, \pi(s_2, AD) + p(s_1, AD, s_6)\, \pi(s_6, AD)) \\
&+ 0.2(p(s_1, AR, s_1)\, \pi(s_1, AD) + p(s_1, AR, s_2)\, \pi(s_2, AD) + p(s_1, AR, s_6)\, \pi(s_6, AD))
\end{aligned}$$

  Note that $p(s_1, AD, s_1)$ and $p(s_1, AR, s_1)$ are not 10% but rather 15%. This is because there's a chance that the agent hits the wall (5%) even if the action it took didn't hit the wall. For eg, in s_1 if it takes AD, there's a 5% chance it actually tries to go left and hits the wall and stays in S1. Also, there's a 10% probability that it does nothing and stays in the same state. So by executing AD, there's a 15% chance it stays in the same state. Doing this also ensures that $\sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, AR, s_1')$ and $\sum_{s_1' \in \{s_1, s_2, s_6\}} p(s_1, AD, s_1')$ are both equal to 1 and not less than 1. Substituting all the values according to the question and Gridworld 687, we get:

$$\begin{aligned}
&= 0.8(0.15 \times 0.8 + 0.05 \times 0.6 + 0.8 \times 0.3) \\
&+ 0.2(0.15 \times 0.8 + 0.8 \times 0.6 + 0.05 \times 0.3) \\
&= 0.8(0.12 + 0.03 + 0.24) + 0.2(0.12 + 0.48 + 0.015) \\
&= 0.8(0.39) + 0.2(0.615) \\
&= 0.312 + 0.123 \\
&= 0.435
\end{aligned}$$

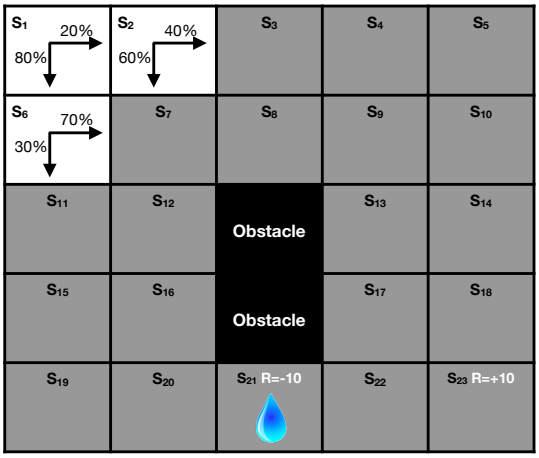

Figure 1: A partially-defined policy for the 687-Gridworld.

3. (**7 Points**) In class we discussed how reward functions can be used to specify what is the "goal" (or objective) of the agent. We presented three ways in which the reward function can be specified: some are extremely general, but not necessarily easy to define in practice; and some are less general but can be defined more intuitively in real-world problems:

   - The most general formulation of the reward function is given by d_R, which specifies an arbitrary distribution over rewards given that the agent is in some state s, executes an action a, and transitions to some state s'.
   - Alternatively, in some problems the reward function can be defined in a way that it (deterministically) returns a scalar number based on s, a, and s'. That is, R can be defined as a function of the form $R(s, a, s')$.
   - Finally, an even simpler formulation of reward functions can be constructed that depends only on the state of the agent (s) and the action that it executed (a). That is, R can be defined as a function of the form $R(s, a)$.

   (Question 3a. 2 Points) First, show from "first principles", *step by step*, how to derive an equation for $R(s)$ in terms of d_R. Recall that, by definition, $R(s) = \mathbb{E}[R_t|S_t = s]$.

   Answer:

$$
\begin{aligned}
R(s) &= \mathbb{E}[R_t|S_t = s] \\
&= \sum_r r \Pr(R_t = r|S_t = s) \\
&= \sum_r r \sum_{a\in\mathcal{A}} \Pr(A_t = a|S_t = s) \Pr(R_t = r|A_t = a, S_t = s) \\
&= \sum_r r \sum_{a\in\mathcal{A}} \Pr(A_t = a|S_t = s) \sum_{s'\in\mathcal{S}} \Pr(S_{t+1} = s'|A_t = a, S_t = s) \Pr(R_t = r|S_{t+1} = s', A_t = a, S_t = s) \\
&= \sum_{a\in\mathcal{A}} \Pr(A_t = a|S_t = s) \sum_{s'\in\mathcal{S}} \Pr(S_{t+1} = s'|A_t = a, S_t = s) \sum_r r \Pr(R_t = r|S_{t+1} = s', A_t = a, S_t = s) \\
R(s) &= \sum_{a\in\mathcal{A}} \pi(s, a) \sum_{s'\in\mathcal{S}} p(s, a, s') \sum_r r d_R(r; s, a, s') \qquad (10)
\end{aligned}
$$

   Expression 10 is the answer.

   (Question 3b. 5 Points) As we studied in class, d_R is the most general way of specifying a reward function. It is often impractical, though, or unnecessarily complex. Instead, we often define reward functions as $R(s, a)$ or

$R(s,a,s')$. We showed in class how all of these can always be rewritten in terms of d_R. In the 687-Gridworld, the reward received by the agent depends only on the state it entered after executing a given action. In this case, we could define a reward function of the type $R(s')$, corresponding to the expected reward given that $S_{t+1} = s'$. Show from "first principles", *step by step*, how to derive an equation for $R(s')$ in terms of d_R. Start by writing $R(s')$ in terms of a conditional expected value and then expand it and manipulate it algebraically. Your final answer should depend on π, p, and d_R. If in your final solution you encounter terms like $\Pr(S_k = s_k)$ for some k (that is, expressions such as $\Pr(S_t = s)$ or $\Pr(S_{t+1} = s')$), you do not need to further simplify them.

Answer:

$$
\begin{aligned}
R(s') &= \mathbb{E}[R_t | S_{t+1} = s'] \\
&= \sum_r r \Pr(R_t = r | S_{t+1} = s') \\
&= \sum_r r \sum_{s \in \mathcal{S}} \Pr(S_t = s | S_{t+1} = s') \Pr(R_t = r | S_{t+1} = s', S_t = s) \\
&= \sum_r r \sum_{s \in \mathcal{S}} \Pr(S_t = s | S_{t+1} = s') \sum_{a \in \mathcal{A}} \Pr(A_t = a | S_{t+1} = s', S_t = s) \Pr(R_t = r | S_{t+1} = s', A_t = a, S_t = s)
\end{aligned}
$$

Rearranging and substituting in $d_R(r; s, a, s')$

$$
= \sum_{s \in \mathcal{S}} \Pr(S_t = s | S_{t+1} = s') \sum_{a \in \mathcal{A}} \Pr(A_t = a | S_{t+1} = s', S_t = s) \sum_r r d_R(r; s, a, s')
$$

Applying property $\Pr(A|B) = \frac{\Pr(A,B)}{\Pr(B)}$ on both the terms.

$$
= \sum_{s \in \mathcal{S}} \frac{\Pr(S_t = s, S_{t+1} = s')}{Pr(S_{t+1} = s')} \sum_{a \in \mathcal{A}} \frac{\Pr(A_t = a, S_{t+1} = s', S_t = s)}{Pr(S_{t+1} = s', S_t = s)} \sum_r r d_R(r; s, a, s')
$$

Rearranging and canceling out in numerator and denominator.

$$
= \sum_{s \in \mathcal{S}} \frac{\cancel{\Pr(S_t = s, S_{t+1} = s')}}{\cancel{Pr(S_{t+1} = s', S_t = s)}} \sum_{a \in \mathcal{A}} \frac{\Pr(A_t = a, S_{t+1} = s', S_t = s)}{Pr(S_{t+1} = s')} \sum_r r d_R(r; s, a, s')
$$

Adding a few extra terms in numerator and denominator.

$$
= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \frac{\Pr(A_t = a, S_{t+1} = s', S_t = s)}{Pr(S_{t+1} = s')} \frac{\Pr(A_t = a, S_t = s)}{Pr(A_t = a, S_t = s)} \frac{\Pr(S_t = s)}{Pr(S_t = s)} \sum_r r d_R(r; s, a, s')
$$

Rearranging suitably to apply the identity $\Pr(A|B) = \frac{\Pr(A,B)}{\Pr(B)}$.

$$
\begin{aligned}
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \frac{\Pr(A_t = a, S_{t+1} = s', S_t = s)}{Pr(A_t = a, S_t = s)} \frac{\Pr(A_t = a, S_t = s)}{Pr(S_t = s)} \frac{\Pr(S_t = s)}{Pr(S_{t+1} = s')} \sum_r r d_R(r; s, a, s') \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \Pr(S_{t+1} = s' | A_t = a, S_t = s) \Pr(A_t = a | S_t = s)) \frac{\Pr(S_t = s)}{Pr(S_{t+1} = s')} \sum_r r d_R(r; s, a, s')
\end{aligned}
$$

Replacing with suitable terms and rearranging

$$
R(s') = \sum_{s \in \mathcal{S}} \frac{\Pr(S_t = s)}{Pr(S_{t+1} = s')} \sum_{a \in \mathcal{A}} \pi(s,a)\, p(s,a,s') \sum_r r d_R(r; s, a, s') \tag{11}
$$

Expression 11 is the final answer

- **Hint 1**: *In this question, the Law of Total Probability, when written in the form* $\Pr(X = x) = \sum_y \Pr(X = x, Y = y)$*, may be useful.*
- **Hint 2**: *In this question, the identity* $\Pr(A|B) = \frac{\Pr(A,B)}{\Pr(B)}$ *(or, equivalently,* $\Pr(A,B) = \Pr(A|B)\Pr(B)$*) may be useful.*

4. (**7 Points**) Suppose you wish to identify an optimal *deterministic* policy for a variant of the Gridworld MDP that consists of 9 states and 4 actions. One approach to finding the optimal policy is through a brute-force search. This involves evaluating the performance, $J(\pi)$, for *all* possible deterministic policies π and selecting the one with the highest expected return. However, this method can be highly time-consuming. In this question, we explore an alternative algorithm. This algorithm *randomly* generates N deterministic policies and returns the best one encountered. To generate a deterministic policy randomly, the algorithm sets $\pi(s) \leftarrow$ random_choice(AU, AD, AL, AR) for every state $s \in \mathcal{S}$.

   In this question, we will theoretically analyze an algorithm that attempts to identify the optimal policy by generating $N = 200$ random policies and selecting the best one. It is worth noting that since only a limited number of policies are considered (in this case, 200), there is a chance the best policy identified by the algorithm is not an optimal policy.

Algorithm 1: A randomized algorithm for identifying optimal policies.

```
best_policy ← None
best_policy_performance ← −∞
N ← 200
for i from 1 to N do
    for each s ∈ S do
        π(s) ← random_choice(UP, UD, UL, UR)    // Generates a random policy
    perf ← J(π)
    if perf > best_policy_performance then
        best_policy_performance ← perf
        best_policy ← π
return best_policy
```

   We aim to ensure that this algorithm returns an optimal policy with a probability of at least 50%. Let ε represent the fraction of all possible deterministic policies that are optimal. To characterize the probability that the algorithm will succeed (i.e., it will identify an optimal policy), note that the probability of *one* randomly-generated policy *not* being optimal is $(1 - \varepsilon)$. Given this, consider the probability that *none* of the policies generated across the $N = 200$ iterations are optimal. This line of reasoning should help deduce the probability that the algorithm *will* find an optimal policy after $N = 200$ iterations, as a function of ε.

   In this question, you should:

   - **(Question 4a. 5 Points)** Determine the value of ε (i.e., the fraction of optimal policies among all possible deterministic policies) necessary to ensure the algorithm has at least a 50% probability of returning an optimal policy. Detail your calculations step-by-step.
     Answer:

$$\begin{aligned}
\Pr(\text{given policy is optimal}) &= \epsilon \\
\Pr(\text{given policy is } \textit{not} \text{ optimal}) &= 1 - \epsilon \\
\Pr(\text{None of 200 given policies are } \textit{not} \text{ optimal}) &= \Pr(\text{given policy is } \textit{not} \text{ optimal})^{200} \\
&= (1 - \epsilon)^{200}
\end{aligned}$$

$$
\begin{aligned}
\Pr(\text{Atleast one out of 200 policies is optimal}) &= 1 - \Pr(\text{None of 200 given policies are } \textit{not} \text{ optimal}) \\
0.5 &= 1 - (1-\epsilon)^{200} \\
0.5 &= 1 - (1-\epsilon)^{200} \\
(1-\epsilon)^{200} &= 1 - 0.5 \\
(1-\epsilon)^{200} &= 0.5 \\
200 \log(1-\epsilon) &= \log 0.5 \\
\log(1-\epsilon) &= \log 0.5/200 = -0.69314718055995/200 = -0.003465736 \\
(1-\epsilon) &= e^{-0.003465736} \\
\epsilon &= 1 - e^{-0.003465736} = 1 - 0.996540263 \\
\epsilon &= 0.003459737
\end{aligned}
$$

- **(Question 4b. 2 Points)** Specify *how many* policies, out of all potential deterministic policies for the MDP under consideration, must be optimal to ensure the algorithm's success.

Answer:

We can infer from the last question that if $\epsilon > 0.003459737$, then our algorithm will work 50% of the time. My assumption is that this condition is our algorithm's 'success' because to design an algorithm that gives at least 1 optimal policy 100% of all the times it is run, we would require the MDP to have at most 199 sub-optimal policies out of all possible deterministic policies. (As this ensures the model will have atleast 1 optimal policy out of the 200 in even the worst case).

Now, total no. of possible deterministic policies = (no. of actions)$^{\text{no. of states}} = 4^9$

ϵ = Fraction of all possible policies that are optimal.

$\epsilon = \frac{\text{no. of optimal policies}}{\text{total policies}}$

$0.003459737 = \frac{\text{no. of optimal policies}}{4^9}$

No. of optimal policies $= 4^9 * 0.003459737$

No. of optimal policies $= 2,62,144 * 0.003459737$

No. of optimal policies $= 906.949296128$

Thus the number of optimal policies should at least be 907 so that our algorithm is successful.

5. (**3 Points**) To fully specify an MDP we have to define $\mathcal{S}$, $\mathcal{A}$, p, R, d_0, and γ. In many real-world applications, however, it may not be possible (or it might be incredibly challenging) to *specify an explicit/analytic* equation for p. Give an example of a real-world problem that can be modeled as an MDP but where p is not known *a priori* by the agent (or cannot be easily specified analytically) and explain why that is the case. Also, describe one possible way by which the agent could incrementally learn/estimate p based on its interactions with the environment.

   Answer:

   Example: A Robot learning how to walk. The robot has four limbs and other body parts. It has different types of motors and sensors at different parts of its body. The actions space may consist of different activations of different motors and various powers. State space may contain variables for its orientation, variables for the location of each body part, and translational and rotational speeds in all axes at which the robot and its parts are moving.

   In this example, it is nearly impossible to say which state the robot will be in the next time instant given an action, say "agent moved ankle motor by 2 degrees and to tell whether the robot will fall, stand or do a backflip?!". Still, the robot can manage to learn how to walk slowly and surely like a toddler when it finds patterns. I am not fully sure how exactly the robot learns using RL, but by my understanding, there could be reward functions for smaller achievements, for eg. "to stand up", to "balance", "to maintain balance while very little" etc. The robot may look for combinations of actions that lead to desirable state. Maybe "being upright and balanced" is a more suitable state leading to "walking" as compared to "fallen" state. There could also be a penalty for such undesirable states.

6. (**9 Points**) The objective of standard RL algorithms is to find policies that maximize expected return. However, this objective does not take into account the possible *risks* of deploying such policies. Consider, for example, an optimal policy (π_1) that, when executed by the agent, results in a return of $+20$ with 50% probability, and a return of -12 with 50% probability. Consider an alternative optimal policy (π_2) that, when executed, produces a return of $+4$ deterministically. The expected return of both policies is the same. One could argue, however, that even though both policies have the same average performance, an agent could prefer policy π_2 since deploying it would never result in a (possibly catastrophic) low performance—in this particular case, a return of -12. Consider the MDP depicted in Figure 2, where $\mathcal{S} = \{s_1, \ldots, s_8\}$, $\mathcal{A} = \{a_1, a_2\}$, $\gamma = 1$, and whose transition and reward functions are described, in Figure 2, by the edges connecting pairs of states. In this figure, an edge connecting states x and y is annotated with information regarding one possible outcome of executing an action a when in state x. Specifically, each edge is annotated with $(a,\ p(x,a,y),\ R(x,a,y))$. Notice that although the transition dynamics of this MDP have already been defined, the particular rewards the agent may receive have not: they are represented by variables $A, B, \ldots, H$.

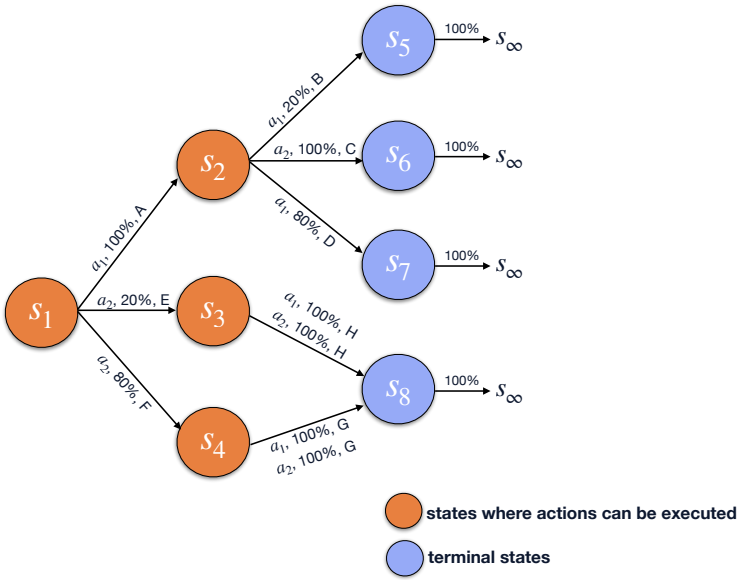

Figure 2: An MDP.

Consider the four possible deterministic policies that may be defined over this MDP:[1]

- π_1, which selects a_1 in s_1 and a_1 in s_2.
- π_2, which selects a_1 in s_1 and a_2 in s_2.
- π_3, which selects a_2 in s_1, a_1 in s_3, and a_1 in s_4.
- π_4, which selects a_2 in s_1, a_2 in s_3, and a_2 in s_4.

Your goal is to define a reward function (by assigning concrete reward values to the variables $A, B, \ldots, H$) such that both π_1 and π_2 are optimal (they share the same optimal expected return), but such that the variance of their returns is different. Furthermore, define rewards such that both π_3 and π_4 are suboptimal policies.

- **Hint**: *An intuitive way of computing the expected return of a given policy is to enumerate all possible paths/trajectories (sequences of states) that can be produced by executing the policy, and annotating each such path with its corresponding return and the probability that it may occur. Then, the expected return of policy π_i, denoted $\mathbb{E}[J(\pi_i)]$, is the weighted average of the returns from possible trajectories, with weights being the probabilities of each trajectory occurring. The variance of the return can be understood similarly. It represents the expected squared difference between the return of each potential trajectory and the policy's expected return. Concretely, $Var[J(\pi_i)] = \sum_\tau \Pr(\tau)(G_\tau - \mathbb{E}[J(\pi_i)])^2$, where τ is one possible trajectory/sequence of states that may be produced by executing π_i, $\Pr(\tau)$ is the probability of this trajectory occurring, and G_τ is the return (i.e., the discounted sum of rewards) associated with τ.*

In this question, you should specify the values of rewards $A, B, \ldots, H$ that you selected and:

- **(Question 6a. 4 Points)** Present the corresponding expected returns of policies $\pi_1, \ldots, \pi_4$.
  Answer: Let's try to find all possible paths for each journey and try to find the expected return in terms of A, B,..., G.
  $\pi_1 : s_1, a_1, A, s_2(1), a_1, s_5(0.2), B$; Probability this happens = 1 * 0.2 ; Total Reward = A+B
  $\pi_1 : s_1, a_1, A, s_2(1), a_1, s_6(0.8), D$; Probability this happens = 1 * 0.8 ; Total Reward = A+D
  Expected Reward = 0.2*(A+B) + 0.8(A+D) = A + 0.2B + 0.8D

  $\pi_2 : s_1, a_1, A, s_3(1), a_2, s_6(1), C$; Probability this happens = 1; Total Reward = A + C
  Expected Reward = A + C

  $\pi_3 : s_1, a_2, s_3(0.2), E, a_1, s_8(1), H$; Probability this happens = 1 * 0.2; Total Reward = E + H
  $\pi_3 : s_1, a_2, s_4(0.8), F, a_1, s_8(1), G$; Probability this happens = 1 * 0.8; Total Reward = F + G
  Expected Reward = 0.2*(E + H) + 0.8*(F + G)

  $\pi_4 : s_1, a_2, s_3(0.2), E, a_2, s_8(1), H$; Probability this happens = 1 * 0.2; Total Reward = E + H
  $\pi_4 : s_1, a_2, s_4(0.8), F, a_2, s_8(1), G$; Probability this happens = 1 * 0.8; Total Reward = F + G
  Expected Reward = 0.2*(E + H) + 0.8*(F + G)

  Now we have to satisfy the following conditions, namely pi_1 and pi_2 are both equal, optimal, and have different variances; pi_3 and pi_4 are both suboptimal. Following combinations of A,B,..,H do the trick:
  A = 100, B = 60, C=100, D=110, E = 0, F = 0, H = 10, G = 20
  Now,
  Expected Reward of $\pi_1 = A + 0.2B + 0.8D = 100 + 12 + 88 =$ **200**
  Expected Reward of $\pi_2 = A + C = 100 + 100 =$ **200**
  Expected Reward of $\pi_3 = 0.2 * (E + H) + 0.8 * (F + G) = 0.2 * 10 + 0.8 * 20 =$ **18**
  Expected Reward of $\pi_4 = 0.2 * (E + H) + 0.8 * (F + G) = 0.2 * 10 + 0.8 * 20 =$ **18**

- **(Question 6b. 5 Points)** Present the variances of the return of these policies. **Answer:** For calculating Variance we need all the possible paths and their paths for all π_is. We also need the expected returns.
  $\pi_1 : s_1, a_1, A, s_2(1), a_1, s_5(0.2), B$; Probability this happens = 0.2 ; Total Reward = 160
  $\pi_1 : s_1, a_1, A, s_2(1), a_1, s_6(0.8), D$; Probability this happens = 0.8 ; Total Reward = 210

[1] Here we only define the actions taken by a policy in states that can be reached when executing the policy.

Expected Reward = 200
Variance = 0.2 * $(210-200)^2$ + 0.8 * $(160-200)^2$
Variance = 0.2 * 100 + 0.8 * 1600
Variance(π_1) = 1300

$\pi_2 : s_1, a_1, A, s_3(1), a_2, s_6(1), C$; Probability this happens = 1; Total Reward = 200
Expected Reward = 200
Variance = 1 * $(0-0)^2$
Variance(π_2) = 0

$\pi_3 : s_1, a_2, s_3(0.2), E, a_1, s_8(1), H$; Probability this happens = 0.2; Total Reward = 10
$\pi_3 : s_1, a_2, s_4(0.8), F, a_1, s_8(1), G$; Probability this happens = 0.8; Total Reward = 20
Expected Reward = 18
Variance = 0.2 * $(10-18)^2$ + 0.8 * $(20-18)^2$
Variance = 0.2 * 64 + 0.8 * 4
Variance(π_3) = 16

$\pi_4 : s_1, a_2, s_3(0.2), E, a_2, s_8(1), H$; Probability this happens = 0.2; Total Reward = 10
$\pi_4 : s_1, a_2, s_4(0.8), F, a_2, s_8(1), G$; Probability this happens = 0.8; Total Reward = 20
Expected Reward = 18 Variance = 0.2 * $(10-18)^2$ + 0.8 * $(20-18)^2$
Variance = 0.2 * 64 + 0.8 * 4
Variance(π_4) = 16

Notice that you are required to provide step-by-step calculations for both the expected returns and return variances of all policies.

7. (**5 Points**) As discussed in class, non-stationary MDPs refer to those where either the environment's dynamics or the reward function can change over time. Consider a game where an RL agent controls an airplane. The airplane's velocity depends on the agent's chosen action, such as *speedUp*, *slowDown*, or *doNothing*. However, external factors like wind speed, which the agent cannot directly measure or control, can also affect the airplane's velocity. To increase the game's difficulty, its developers introduced a dynamic element: the wind speed changes automatically every 100 steps. For instance, at time $t = 23$, if the agent chooses the *doNothing* action and there's no wind, the airplane's velocity remains unchanged. But by time $t = 140$, the wind speed might have shifted, and now, the same action (*doNothing*) could result in an increase of the airplane's velocity by 50km/h. Formally, this means that $\Pr(S_{t+1} = s' | S_t = s, A_t = a) \neq \Pr(S_{t+k+1} = s' | S_{t+k} = s, A_{t+k} = a)$. In other words, when facing the same state s, and executing the same action a, the distribution over the agent's next states can differ between time t and time $t + k$.

   In the example above, the non-stationarity resulted from a factor that was neither directly observable to the agent nor under its control: the wind speed. A significantly more challenging problem occurs when the non-stationarity does not result from external factors but is a *direct result* of the agent's actions. Describe a real-world example where the environment's transition dynamics could be affected by this type of non-stationarity, and explain why that is the case. Discuss the potential consequences if an agent overlooks this non-stationarity while learning a policy. Lastly, propose one potential solution to this challenge—even if your solution is heuristic and not guaranteed to always work.

   Answer:

   A real-world example will be to design an autonomous system to learn how to race on a track using a manual car. Here the car is a normal manual transmission car that we use in our day-to-day lives. And say the track is a curvy F1 track. The objective of the car is to complete 10,000 laps as fast as possible. The system has access to all the information that a normal human driver has - Cameras and gyroscopic sensors to mimic the human sense.

   In this example, due to the sheer size of the task, there will be a lot of wear and tear to the car's parts. For eg. the tires may start losing their grip, and the clutch plate might start getting worn. Even the engine components may start accumulating dirt and there may be a lot of internal heat generation. This might impact the performance of the car as it reaches the end of the task. And the strategy/policy that might work best for starting laps might not be the best towards the end. The perfect lap - with the fastest acceleration in long

stretches in the fastest slowdown along with the optimal race lines may not work towards the end. The car may even crash due to not being able to break as precisely as it used to do initially. Pushing the car to its limit during the start may actually be a bad strategy as it would bring this low-performance state a lot faster. This means that the agent should actually take this non-stationarity into account.

One potential solution to this challenge may be to include more sensors - for eg, audio, and smell. This is to mimic human senses, as we as humans can sometimes judge the state of the car just by listening to the engine or smell if something is going down as well. Another heuristic may be keeping track of the number of laps as well as historical data like the time taken at key areas of the race just to see how well the car is responding now as compared to earlier. Based on these inputs, the agent might be able to adapt its strategy as time goes on.

8. (**10 Points**) In reinforcement learning, we often employ the formalism of Markov Decision Processes (MDPs) to model decision-making problems. However, this approach has its constraints and may not be suited for certain situations.

- **(Question 8a. 4 Points)** First, describe a concrete real-world scenario where a learning agent's objective cannot be described by a scalar reward function—even if the agent can observe all relevant features of the environment and has a perfect memory of its history/past.

  Answer: I would like to take the example of an autonomous car. Here the learning objective of the agent is to learn to drive the car autonomously without any manual intervention. There cannot be a single scalar reward function in this example. Because there are a lot of variables to keep track of. Firstly the agent needs to learn how to drive, follow all the traffic rules, drive safely such that no accidents are caused, and also not drive too slow as to cause further delays resulting in rash driving from the drivers around the taxi. There cannot be a single scalar reward function defining all of these points.

- **(Question 8b. 3 Points)** Next, discuss a seemingly reasonable hypothetical reward function that might be proposed to capture this learning objective. Then, in plain terms, explain why this approach would not succeed.

  Answer: We could hypothetically create a linear combination of all of these tasks to create a reasonable reward function. The weights could be adjusted by priority. For eg. driving safely without any accidents should be prioritized more than driving fast.

  But the problem here is that these variables aren't independent of each other. This means a decrease in one can affect another variable as well. For instance, driving too safely would mean going too slow. But as mentioned in the previous answer, this could lead other drivers to start getting annoyed and start rash driving making it more unsafe. This means a decrease in safe driving doesn't result in a safe environment as much as expected. But rather decreasing the safety. There will always be trade-offs like this whenever we try to create to single scalar reward function.

- **(Question 8c. 3 Points)** Finally, discuss what property of the learning objective prevents it from being captured effectively using conventional scalar reward functions.

  Answer: The scalar reward functions are typically used to guide an agent towards achieving a single, well-defined objective. But in this example, our learning objective isn't a single value - which can be either high or low. It is an amalgamation of multiple objectives - (safety, speed, and efficiency in our example). There are unknown trade-offs in the environment that need to be kept track of. These prevent the objective to captured effectively using conventional scalar reward functions.

Hint: *There are many approaches to this question. Here are three high-level questions that may be good starting points when reasoning about this problem:*

- *What would happen if you wanted an RL agent to learn to reach a goal state while ensuring with 100% probability that it would avoid visiting a specific dangerous state? What if you wanted to ensure that the agent would avoid visiting the dangerous state with 95% probability?*
- *How would you use scalar reward functions to model settings where the agent has conflicting goals and where the trade-off between them is unspecified? Consider, for instance, an agent that aims to learn a policy that maximizes its fun, and that also minimizes any risks to its life. If a trade-off between these objectives is not specified, are there any policies that you can say with certainty are better or worse than any others?*
- *What if you wanted an RL agent to have an equal probability of visiting every state in the environment? How would you enforce this objective via a standard scalar reward function?*

Your answer to questions **Q8a**–**Q8c** should be based on a *concrete* real-world learning problem that requires dealing with challenges such as the ones discussed above.

Part Two: Programming (45 Points Total)

Consider the following MDP:

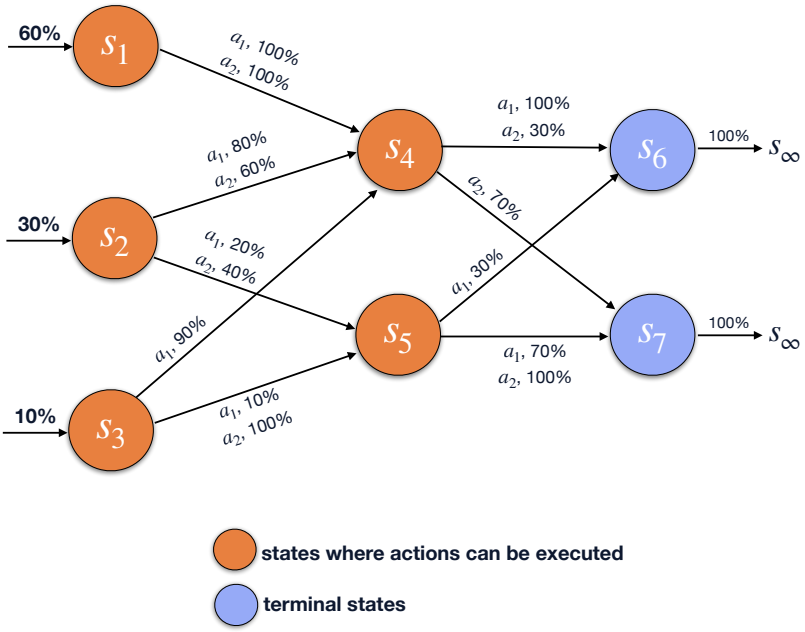

Figure 3: An MDP.

Assume the following initial state distribution, d_0, and transition function, p, where all transition probabilities not indicated in the table below are 0.

$d_0(s_1) = 0.6$	$d_0(s_2) = 0.3$	$d_0(s_3) = 0.1$

$p(s_1, a_1, s_4) = 1.0$	
$p(s_1, a_2, s_4) = 1.0$	
$p(s_2, a_1, s_4) = 0.8$	$p(s_2, a_1, s_5) = 0.2$
$p(s_2, a_2, s_4) = 0.6$	$p(s_2, a_2, s_5) = 0.4$
$p(s_3, a_1, s_4) = 0.9$	$p(s_3, a_1, s_5) = 0.1$
$p(s_3, a_2, s_5) = 1.0$	
$p(s_4, a_1, s_6) = 1.0$	
$p(s_4, a_2, s_6) = 0.3$	$p(s_4, a_2, s_7) = 0.7$
$p(s_5, a_1, s_6) = 0.3$	$p(s_5, a_1, s_7) = 0.7$
$p(s_5, a_2, s_7) = 1.0$	

Assume the following reward function:

$R(s_1, a_1) = 7$	$R(s_1, a_2) = 10$
$R(s_2, a_1) = -3$	$R(s_2, a_2) = 5$
$R(s_3, a_1) = 4$	$R(s_3, a_2) = -6$
$R(s_4, a_1) = 9$	$R(s_4, a_2) = -1$
$R(s_5, a_1) = -8$	$R(s_5, a_2) = 2$

Finally, consider the following stochastic policy, π:

$\pi(s_1, a_1) = 0.5$	$\pi(s_1, a_2) = 0.5$
$\pi(s_2, a_1) = 0.7$	$\pi(s_2, a_2) = 0.3$
$\pi(s_3, a_1) = 0.9$	$\pi(s_3, a_2) = 0.1$
$\pi(s_4, a_1) = 0.4$	$\pi(s_4, a_2) = 0.6$
$\pi(s_5, a_1) = 0.2$	$\pi(s_5, a_2) = 0.8$

(Question 1. 15 Points) Find an *analytic, closed-form* expression for

$$J(\pi) \quad = \quad \mathbb{E}\left[\sum_{t=0}^{1} \gamma^t R_t\right]$$

as a function of γ. To do this, you *may* choose do it from "first principles", by repeatedly using the properties of probability distributions and expected values introduced in Section 2. If you do not wish to derive a closed-form expression for $J(\pi)$ this way, you are also allowed to write it directly as a function of d_0, p, π, and R, similarly to the final equation described in the "**Example problem**" introduced in Section 2. You must present your derivation step-by-step.

- **Hint:** *Start by applying the property of linearity of expectation to the definition of $J(\pi)$ and then derive separate equations for each of the resulting terms.*

Your final answer **must** be in the form of $J(\pi) = c_1 + \gamma c_2$, where each c_i is a real-valued constant. To obtain an equation of this form, start from the analytic, closed-form expression you derived earlier and plug in the particular transition probabilities, rewards, and policy probabilities specified above. You must present your final answer in the form discussed above, showing your work step-by-step.

Answer:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{1} \gamma^t R_t\right]$$

$$J(\pi) = \mathbb{E}\left[\gamma^1 R_1 + \gamma^0 R_0\right]$$

$$J(\pi) = \gamma \mathbb{E}\left[R_1\right] + \mathbb{E}\left[R_0\right]$$

Now let's solve term 1 and term 2 separately.

$$
\begin{aligned}
\gamma\mathbb{E}\left[R_1\right] &= \gamma \sum_r r \Pr[R_1 = r] \\
&= \gamma \sum_r r \sum_{s_1 \in \mathbb{S}} \Pr[S_1 = s_1] \Pr[R_1 = r | S_1 = s_1] \\
&= \gamma \sum_r r \sum_{s_1 \in \mathbb{S}} \Pr[S_1 = s_1] \sum_{a_1 \in \mathbb{A}} \Pr[A1 = a_1 | S_1 = s_1] \Pr[R_1 = r | S_1 = s_1, A_1 = a_1] \\
&= \gamma \sum_{s_1 \in \mathbb{S}} \Pr[S_1 = s_1] \sum_{a_1 \in \mathbb{A}} \sum_r r \Pr[R_1 = r | S_1 = s_1, A_1 = a_1] \pi(s_1, a_1) \\
&= \gamma \sum_{s_1 \in \mathbb{S}} \sum_{a_1 \in \mathbb{A}} R(s_1, a_1) \pi(s_1, a_1) \Pr[S_1 = s_1] \\
&= \gamma \sum_{s_1 \in \mathbb{S}} \sum_{a_1 \in \mathbb{A}} R(s_1, a_1) \, \pi(s_1, a_1) \sum_{s_0 \in \mathbb{S}} \Pr[S_0 = s_0] \Pr[S_1 = s_1 | S_0 = s_0] \\
&= \gamma \sum_{s_1 \in \mathbb{S}} \sum_{a_1 \in \mathbb{A}} R(s_1, a_1) \, \pi(s_1, a_1) \sum_{s_0 \in \mathbb{S}} d_0(s_0) \Pr[S_1 = s_1 | S_0 = s_0] \\
&= \gamma \sum_{s_1 \in \mathbb{S}} \sum_{a_1 \in \mathbb{A}} R(s_1, a_1) \, \pi(s_1, a_1) \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \Pr[A_0 = a_0 | S_0 = s_0] \Pr[S_1 = s_1 | A_0 = a_0, S_0 = s_0] \\
&= \gamma \sum_{s_1 \in \mathbb{S}} \sum_{a_1 \in \mathbb{A}} R(s_1, a_1) \, \pi(s_1, a_1) \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \pi(s_0, a_0) \, p(s_0, a_0, s_1) \\
&= \gamma \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathbb{S}} p(s_0, a_0, s_1) \sum_{a_1 \in \mathbb{A}} \pi(s_1, a_1) R(s_1, a_1)
\end{aligned}
$$

Term 2:

$$
\begin{aligned}
\mathbb{E}\left[R_0\right] &= \sum_r r \Pr[R_0 = r] \\
&= \sum_r r \sum_{s_0 \in \mathbb{S}} \Pr[S_0 = s_0] \Pr[R_0 = r | S_0 = s_0] \\
&= \sum_r r \sum_{s_0 \in \mathbb{S}} d_0(s_0) \Pr[R_0 = r | S_0 = s_0] \\
&= \sum_r r \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \Pr[A_0 = a_0 | S_0 = s_0] \Pr[R_0 = r | A_0 = a_0, S_0 = s_0] \\
&= \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \pi(s_0, a_0) \sum_r r \Pr[R_0 = r | A_0 = a_0, S_0 = s_0] \\
&= \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \pi(s_0, a_0) R(s_0, a_0)
\end{aligned}
$$

Combining term 1 and term 2:

$$
J(\pi) = \gamma \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \pi(s_0, a_0) \sum_{s_1 \in \mathbb{S}} p(s_0, a_0, s_1) \sum_{a_1 \in \mathbb{A}} \pi(s_1, a_1) R(s_1, a_1) + \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \pi(s_0, a_0) R(s_0, a_0)
$$

Taking in common:

$$
\sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \pi(s_0, a_0)
$$

$$
J(\pi) = \sum_{s_0 \in \mathbb{S}} d_0(s_0) \sum_{a_0 \in \mathbb{A}} \pi(s_0, a_0) \left(\gamma \sum_{s_1 \in \mathbb{S}} p(s_0, a_0, s_1) \sum_{a_1 \in \mathbb{A}} \pi(s_1, a_1) R(s_1, a_1) + R(s_0, a_0) \right) \tag{12}
$$

Equation 12 is the analytical equation.

Now, $s_0 \in \mathbb{S} = s_1, s_2, s_3$; $a_0 \in \mathbb{A} = a_1, a_2$; $s_1 \in \mathbb{S} = s_4, s_5$; $a_1 \in \mathbb{A} = a_1, a_2$

Substituting these values back in 12, we get:

Also renaming s_0, a_0, s_1, a_1 in original equation to s'_0, a'_0, s'_1, a'_1 to avoid confusion between summation variables and MDP variables.

$$
\begin{aligned}
J(\pi) &= \sum_{s'_0 \in \mathbb{S}} d_0(s'_0) \sum_{a'_0 \in \mathbb{A}} \pi(s'_0, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s'_0, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s'_0, a'_0) \right) \\
&= d_0(s_1) \sum_{a'_0 \in \mathbb{A}} \pi(s_1, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_1, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_1, a'_0) \right) \\
&+ d_0(s_2) \sum_{a'_0 \in \mathbb{A}} \pi(s_2, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_2, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_2, a'_0) \right) \\
&+ d_0(s_3) \sum_{a'_0 \in \mathbb{A}} \pi(s_3, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_3, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_3, a'_0) \right) \\
&= 0.6 \sum_{a'_0 \in \mathbb{A}} \pi(s_1, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_1, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_1, a'_0) \right) \\
&+ 0.3 \sum_{a'_0 \in \mathbb{A}} \pi(s_2, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_2, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_2, a'_0) \right) \\
&+ 0.1 \sum_{a'_0 \in \mathbb{A}} \pi(s_3, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_3, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_3, a'_0) \right) \\
&= 0.6 \left[0.5 \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_1, a_1, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_1, a_1) \right) \right] \\
&+ 0.6 \left[0.5 \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_1, a_2, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_1, a_2) \right) \right] \\
&+ 0.3 \sum_{a'_0 \in \mathbb{A}} \pi(s_2, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_2, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_2, a'_0) \right) \\
&+ 0.1 \sum_{a'_0 \in \mathbb{A}} \pi(s_3, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_3, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_3, a'_0) \right) \\
&= 0.6 \left[0.5 \left(\gamma \sum_{a'_1 \in \mathbb{A}} \pi(s_4, a'_1) R(s_4, a'_1) + 7 \right) \right] \\
&+ 0.6 \left[0.5 \left(\gamma \sum_{a'_1 \in \mathbb{A}} \pi(s_4, a'_1) R(s_4, a'_1) + 10 \right) \right] \\
&+ 0.3 \sum_{a'_0 \in \mathbb{A}} \pi(s_2, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_2, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_2, a'_0) \right) \\
&+ 0.1 \sum_{a'_0 \in \mathbb{A}} \pi(s_3, a'_0) \left(\gamma \sum_{s'_1 \in \mathbb{S}} p(s_3, a'_0, s'_1) \sum_{a'_1 \in \mathbb{A}} \pi(s'_1, a'_1) R(s'_1, a'_1) + R(s_3, a'_0) \right)
\end{aligned}
$$

As expanding in latex is taking too much time and effort, I substituted the values in my notebook and got the following equation:

$$\begin{aligned}J(\pi) &= 0.6\,[0.5\,(7+\gamma[R(s_4)]) + 0.5\,(10+\gamma[R(s_4)])] \\ &\quad + 0.3\,[0.7\,(-3+\gamma[0.8*R(s_4)+0.2*R(s_5)]) + 0.3\,(5+\gamma[0.6*R(s_4)+0.4*R(s_5)])] \\ &\quad + 0.1\,[0.9\,(4+\gamma[0.9*R(s_4)+0.1*R(s_5)]) + 0.1\,(-6+\gamma[R(s_5)])]\end{aligned}$$

Where R(s4) = reward at s4, = 0.4*9 + 0.6(-1) = 3 and R(s5) = 0.2*(-8)+0.8(2) = 0

$$\begin{aligned}J(\pi) &= 0.6\,[0.5\,(7+\gamma[3]) + 0.5\,(10+\gamma[3])] \\ &\quad + 0.3\,[0.7\,(-3+\gamma[0.8*3]) + 0.3\,(5+\gamma[0.6*3])] \\ &\quad + 0.1\,[0.9\,(4+\gamma[0.9*3]) + 0.1\,(-6)] \\ J(\pi) &= 0.6\,[3\gamma+8.5] \\ &\quad + 0.3\,[(-0.6)+2.22\gamma] \\ &\quad + 0.1\,[2.43\gamma+3] \\ J(\pi) &= 5.22+2.709\gamma \qquad (13)\end{aligned}$$

Therefore $c_1 =$ **5.22** and $c_2 =$ **2.709**

(Question 2 - *Programming Question*. 30 Points)

In this question, you will write a program to estimate $J(\pi)$ by simulating many of the possible outcomes (returns) that might result from running π on the previously-defined MDP. Each simulation will produce a particular sequence of states, actions, and rewards, and, thus, a particular discounted return. Since $J(\pi)$ is defined as the *expected* discounted return, you can construct an estimate of $J(\pi)$, $\hat{J}(\pi)$, by averaging the discounted returns observed across N simulations.

In particular:

- To run one simulation (or episode, or trial), you should follow the "Agent-Environment Interaction" procedure introduced in Lecture #3.
- Start by creating a function called *runEpisode* that takes as input a policy and a value of γ, and that returns the empirical discounted return resulting from that episode.
- Let G^i be the discounted return of the i^{th} episode. You will estimate $J(\pi)$ by computing $\hat{J}(\pi) := \frac{1}{N}\sum_{i=1}^{N} G^i$.

(Question 2a. 8 Points). Construct $\hat{J}(\pi)$ by running 150,000 simulations/episodes. You should then create a graph where the x axis shows the number of episodes, and the y axis presents the estimate $\hat{J}(\pi)$ constructed based on all episodes executed up to that point. As an example, the point $x = 100$ in this graph should have as its corresponding y coordinate the estimate $\hat{J}(\pi)$ built using the discount returns from the first 100 simulations. Your plot should include the results of this simulation up to 150,000 simulations/episodes. You should use $\gamma = 0.9$.

Answer: Plot : Figure 4 and Figure 5

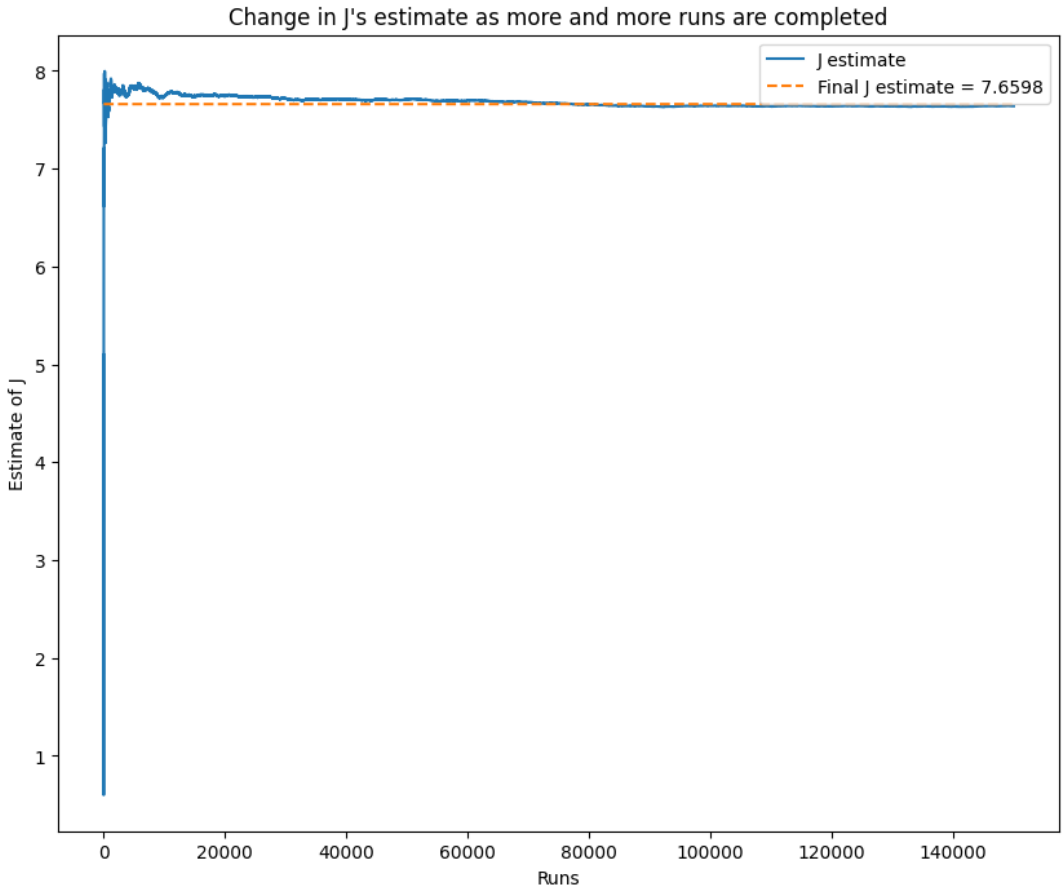

Figure 4: Q.2a: Linear Scale: J's estimate as more and more runs are completed

(Question 2b. 5 Points). Report the average discounted return, as well as its variance, at the end of this process. In other words, report the value of $\hat{J}(\pi)$ after executing 150,000 episodes, as well as the variance of the return of all simulations used to compute this last estimate of the return of the policy.

Answer: Answer calculated from the data in Jupyter Notebook.
Average discounted return: 7.659743999995969
Variance: 45.38499287750001

(Question 2c. 5 Points). Estimate $\hat{J}(\pi)$ using different discount rates: $\gamma \in \{0.25, 0.5, 0.75, 0.99\}$. Compare these estimates with their true values, computed according to the closed-form solution for $J(\pi)$ found in the first part of this question. These values should approximately match (i.e., $J(\pi) \approx \hat{J}(\pi)$).
Answer:
Table 1 contains the comparisons of $\hat{J}(\pi)$ w.r.t γ from analytical and simulated values. We can

γ	From Analatical Equation 13	From Program
1.0	7.929	7.921926666666667
0.99	7.90191	7.902285533336392
0.9	7.6581	7.659743999995969
0.75	7.25175	7.255098333333334
0.5	6.5745	6.590343333333333
0.25	5.89725	5.904346666666667
0.0	5.22	5.23526

Table 1: Table containing Comparisons of $\hat{J}(\pi)$ from analytical equation and from the simulation code.

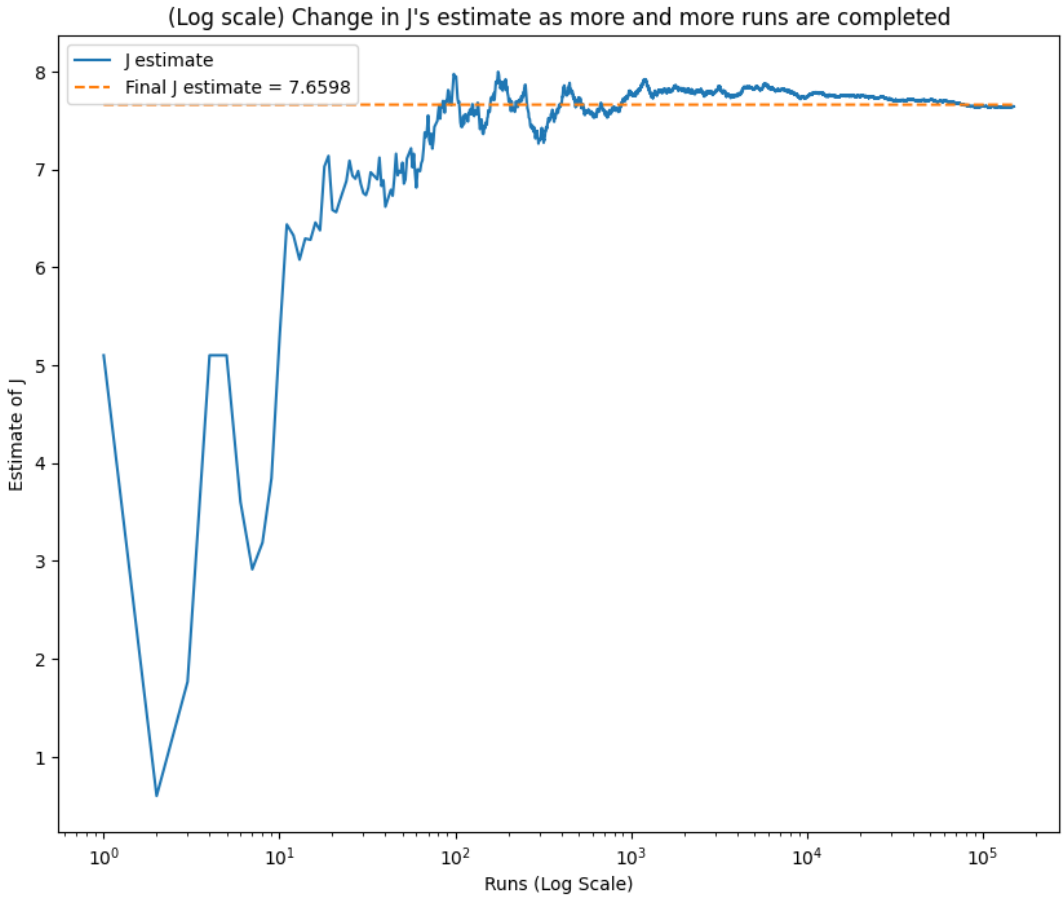

Figure 5: Q.2a: Logarithmic Scale: J's estimate as more and more runs are completed

observe that the values match approximately!

(Question 2d. 12 Points). Next, you will implement the randomized algorithm from question **Q4**. Your aim is to use it to identify an optimal policy for the MDP defined above using the following steps:

(a) Policy Generation & Evaluation

- Use your *runEpisode* function to estimate the performance of different policies. Specifically, you will use it to evaluate the performance of the random *deterministic* policies generated by the algorithm from **Q4**.
- To evaluate the performance of each random policy, execute *runEpisode* 100 times and average the resulting performance estimates. This means, for example, that with $N = 20$ you will make $20 \times 100 = 2,000$ calls to *runEpisode*.
- You should use a discount factor of $\gamma = 0.9$. Also, recall that all policies that will be generated by the algorithm from **Q4** should be deterministic policies. This constrains the search space and makes it easier for an optimal (or near-optimal) policy to be identified.

(b) Presentation of Empirical Results

- Plot a graph with the x axis representing values of N ranging from 1 to 250 in increments of 10. The y axis should indicate the performance estimate, $\hat{J}(\pi)$, of the best policy determined by the algorithm for a given N.
  - As an example, the point $x = 100$ in this graph should have as its corresponding y coordinate the estimate $\hat{J}(\pi)$ of the best policy identified by the algorithm when constructing $N = 100$ random policies.

- Present this graph in your report and then examine it to visually identify the value of N at which the algorithm starts to consistently identify an optimal (or near-optimal) policy. Specifically, pinpoint where the curve seems to stabilize.

The plot in figure 6 looked too sparse to make a judgement about N. So I plotted figure 7 with just an increment of 1 in N. I also smoothed the J-estimates so we can visually make out that N stabilized a bit after **N=124 (approx.)**. The J-estimate after this point comes out to around 15.71

The best policy I got was the following table:

$\pi(s_1, a_1) = 0$	$\pi(s_1, a_2) = 1$
$\pi(s_2, a_1) = 0$	$\pi(s_2, a_2) = 1$
$\pi(s_3, a_1) = 1$	$\pi(s_3, a_2) = 0$
$\pi(s_4, a_1) = 1$	$\pi(s_4, a_2) = 0$
$\pi(s_5, a_1) = 0$	$\pi(s_5, a_2) = 1$
$\pi(s_6, a_1) = 0$	$\pi(s_5, a_2) = 1$
$\pi(s_7, a_1) = 0$	$\pi(s_5, a_2) = 1$

Note that the values for s6 and s7 don't matter as those are end states. The code generated actions for them due to the nature of how I coded it. Every state has all the actions possible. So even end states have actions to them. But this doesn't matter as the episode ends as soon as the agent lands in any of the end state.

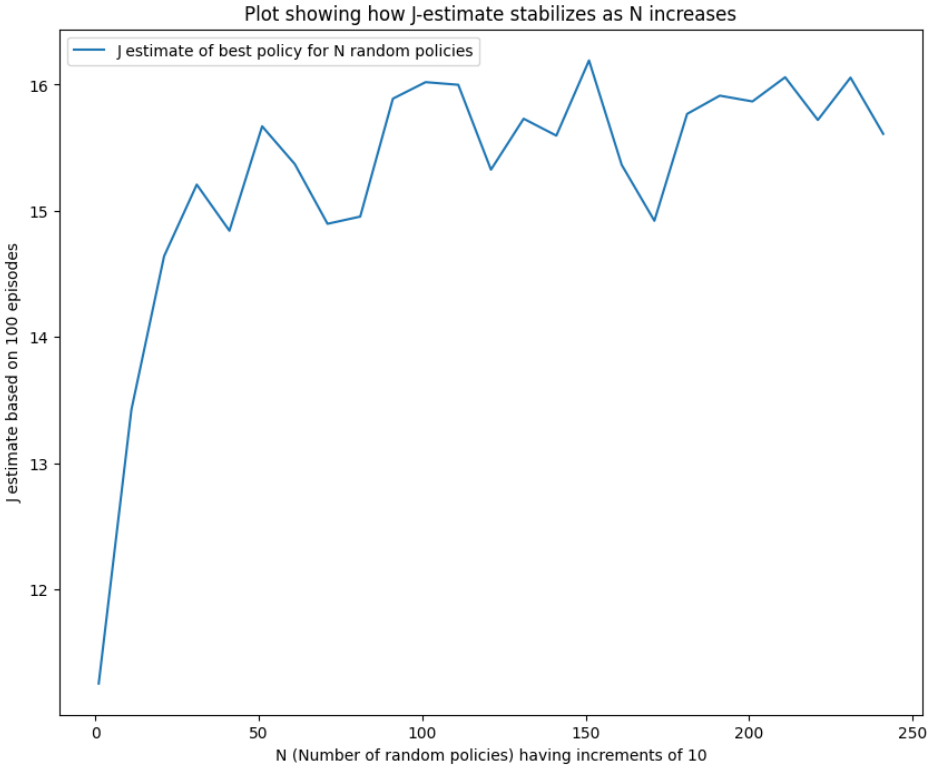

Figure 6: Q.2d: Increments of 10: J-estimate stabilizing as N increases

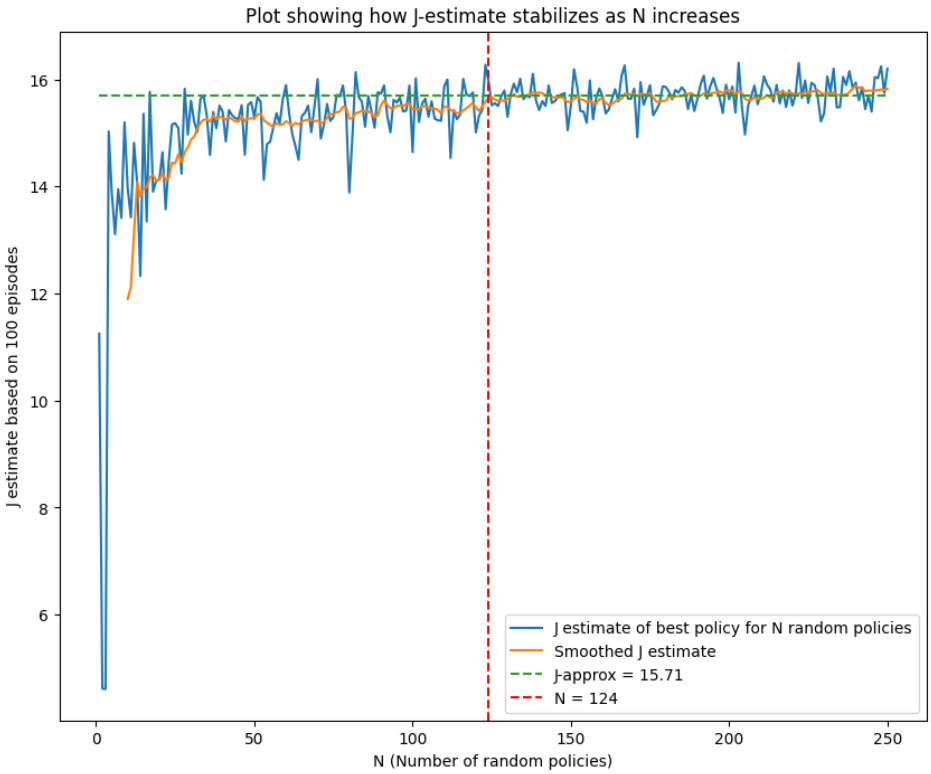

Figure 7: Q.2d: Increments of 1: J-estimate stabilizing as N increases