# Table of Contents

1 Syllabus ........................................ 2
   1.1 Class ........................................ 2
   1.2 Website ...................................... 2
   1.3 Book .......................................... 2
   1.4 Office Hours .................................. 2
   1.5 Teaching Assistants and Office Hours ...... 2
   1.6 Piazza ........................................ 2
   1.7 Support Summary .............................. 3
   1.8 Grading ....................................... 3
   1.9 Late Policy .................................... 4
   1.10 Missing Class / Assignments ............... 4
   1.11 Disability Services ............................ 4
   1.12 Cheating ..................................... 4
   1.13 LATEX ........................................ 4

2 Introduction ..................................... 6
   2.1 Notation ...................................... 6
   2.2 What is Reinforcement Learning (RL)? ....... 6
   2.3 687-Gridworld: A Simple Environment ...... 10
   2.4 Describing the Agent and Environment Mathematically ... 11

Homework 1 ........................................... 17
Pop Quiz 1 ........................................... 23
2.5 Additional Terminology, Notation, and Assumptions .. 24
1 Syllabus

1.1 Class

Class will be held on Tuesdays and Thursdays from 4:00pm–5:15pm in Engineering Lab II, Room 119. Lectures will be given primarily on the whiteboard, with typed notes provided below and updated throughout the semester as additional material is covered. These notes are not a complete summary of all material that students are responsible for—you are responsible for all material covered in class, even if it is not present in these notes.

1.2 Website

The class website is https://people.cs.umass.edu/~pthomas/courses/CMPSCI_687_Fall2019.html. All homework assignments, due dates, and notes will be posted there.

1.3 Book

The start of the course will be roughly based on the first edition of Sutton and Barto’s book, Reinforcement Learning: An Introduction. It can be found on Amazon here. It is also available for free online here. Although the book is a fantastic introduction to the topic (and I encourage purchasing a copy if you plan to study reinforcement learning), owning the book is not a requirement.

1.4 Office Hours

Prof. Thomas’ office hours will be Mondays from 1:30pm–3:00pm in his office, room 346 of the computer science building. Office hours will follow the academic calendar: they will be offered from 1:30-3:00pm on (and only on) all days that are a Monday schedule.

1.5 Teaching Assistants and Office Hours

The teaching assistants (TAs) this semester will be Blossom Metevier (bmetevier@umass.edu) and Scott Jordan (sjordan@cs.umass.edu). Blossom will have office hours on Thursdays from 8:55am–9:55am in CS Building room 207. Scott will have office hours on Tuesdays from 3pm–4pm in CS Building room 207.

1.6 Piazza

The course will use Piazza as a forum where you can ask questions. Every afternoon (at some time between 1pm and 6pm), Prof. Thomas or one of the TAs will go through and answer all of the questions on Piazza. If you asked
a question before 1pm on a weekday that is not a holiday, and did not get a response by 6pm, please e-mail Prof. Thomas directly (pthomas@cs.umass.edu), as this should not occur.

1.7 Support Summary
Office hours:

- **Mondays**: 1:30pm-3:00pm, Prof. Thomas’ office hours, CS room 346.
- **Tuesdays**: 3:00pm–4:00pm, Scott Jordan’s office hours, CS room 207.
- **Thursdays**: 8:55am–9:55am, Blossom Metevier’s office hours, CS room 207.

Piazza question answering (some time between 1pm and 6pm):

- **Mondays**: Blossom Metevier
- **Tuesdays**: Scott Jordan
- **Wednesdays**: Blossom Metevier
- **Thursdays**: Prof. Thomas
- **Fridays**: Scott Jordan

1.8 Grading
Your grade will have four components:

1. **Homework Assignments (50%)**: There will be roughly seven homework assignments. Each problem in an assignment will specify its point value, and not all homework assignments will necessarily have the same point value (i.e., some homework assignments may be smaller and worth less than others). All assignments will have total point values of at most 100 (thus, the last assignment will not have a point value so high that previous assignments are irrelevant).

2. **Pop Quizzes (15%)**: There will be pop-quizzes given in class without prior announcement. They will typically take about 10 minutes to complete and will be given at the start of class. If you know in advance that you will miss class, please e-mail both TAs, and you may be excused from any quizzes that occur that day.

3. **Midterm Exam (15%)**: There will be a midterm exam relatively late in the semester (precise date TBD).

4. **Project (20%)**: There will be a course project. The details of the project will be announced later in the semester, and may depend on how much content is covered.
A cumulative grade in $[90\%-100\%]$ will be an A- or A, $[80\%,90\%)$ will be a B-, B, or B+, and $[70\%,80\%)$ will be a C-, C, or C+. Course grades will be curved only in students’ favor (that is, these thresholds may be lowered, but a grade of 90% will not be lower than an A-).

1.9 Late Policy

Late homework assignments will not be accepted. An assignment submitted one minute late is late, and will not be accepted. I recommend submitting homework well in advance of the due date and time.

1.10 Missing Class / Assignments

If you are going to miss class, e-mail the TAs (not Prof. Thomas) before the start of class letting them know. You will then be excused from any pop-quizzes that occur on that day (your grade will be computed as though that quiz did not occur).

Sometimes things come up that prevent you from completing an assignment well or at all. To handle this, your homework assignment with the lowest score will be dropped. To avoid encouraging skipping the final assignment, if you perform consistently on all assignments without any clearly low outliers, Prof. Thomas will consider this when assigning grades (it may bump you up if you’re near a boundary).

1.11 Disability Services

If you have a disability and require accommodations, please let me know as soon as possible. You will need to register with Disability Services (161 Whitmore Administration Building; phone (413) 545–0892). Information on services and materials for registering are also available on their website: [www.umass.edu/disability](http://www.umass.edu/disability).

1.12 Cheating

Cheating will not be tolerated. Each assignment includes instructions about what forms of collaboration are allowed. Copying answers or code from online sources or from solutions to assignments from previous years is always considered cheating. All instances of cheating will be reported to the university’s Academic Honesty Board, and will result in a failing grade letter grade for the course.

1.13 LATEX

Your homework submissions must be typed using \textsc{LaTeX}. If you have not used \textsc{LaTeX} before, you may want to complete an online tutorial now. Also, the instructor and TAs are prepared to help you learn about \textsc{LaTeX} during their office hours. Note: The formatting of math using editors like Microsoft Word
is not as clear as \LaTeX. Assignments created using other editors will not be accepted.
2 Introduction

2.1 Notation

When possible, sets will be denoted by calligraphic capital letters (e.g., $\mathcal{X}$), elements of sets by lowercase letters (e.g., $x \in \mathcal{X}$), random variables by capital letters (e.g., $X$), and functions by lowercase letters (e.g., $f$). This will not always be possible, so keep an eye out for exceptions (e.g., later $P$ will be a function).

We write $f : \mathcal{X} \to \mathcal{Y}$ to denote that $f$ is a function with domain $\mathcal{X}$ and range $\mathcal{Y}$. That is, it takes as input an element of the set $\mathcal{X}$ and produces as output an element of $\mathcal{Y}$. We write $|\mathcal{X}|$ to denote the cardinality of the set $\mathcal{X}$—the number of elements in $\mathcal{X}$, and $|x|$ to denote the absolute value of $x$ (thus the meaning of $|\cdot|$ depends on context).

We typically use capital letters for matrices (e.g., $A$) and lowercase letters for vectors (e.g., $b$). We write $A^\top$ to denote the transpose of $A$. Vectors are assumed to be column vectors. Unless otherwise specified, $\|b\|$ denotes the $l^2$-norm (Euclidean norm) of the vector $v$.

We write $\mathbb{N}_{>0}$ to denote the natural numbers not including zero, and $\mathbb{N}_{\geq 0}$ to denote the natural numbers including zero.

We write $:= \triangleq$ to denote is defined to be. In lecture we may write $\triangleq$ rather than $\eqdef$ since the triangle is easier to see when reading handwriting from the back of the room.

If $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ for any sets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, then we write $f(\cdot, y)$ to denote a function, $g : \mathcal{X} \to \mathcal{Z}$, such that $g(x) = f(x, y)$ for all $x \in \mathcal{X}$.

We denote sets using brackets, e.g., $\{1, 2, 3\}$, and sequences and tuples using parentheses, e.g., $(x_1, x_2, \ldots)$.

The notation that we use is not the same as that of the book or other sources (papers and books often use different notations, and there is no agreed-upon standard). Our notation is a mix between the notations of the first and second editions of Sutton and Barto’s book.

2.2 What is Reinforcement Learning (RL)?

Reinforcement learning is an area of machine learning, inspired by behaviorist psychology, concerned with how an agent can learn from interactions with an environment.
Agent-environment diagram.

**Agent**: Child, dog, robot, program, etc.

**Environment**: World, lab, software environment, etc.

**Evaluative Feedback**: Rewards convey how “good” an agent’s actions are, not what the best actions would have been. If the agent was given instructive feedback (what action it should have taken) this would be a supervised learning problem, not a reinforcement learning problem.

**Sequential**: The entire sequence of actions must be optimized to maximize the “total” reward the agent obtains. This might require forgoing immediate rewards to obtain larger rewards later. Also, the way that the agent makes decisions (selects actions) changes the distribution of states that it sees. This means that RL problems aren’t provided as fixed data sets like in supervised learning, but instead as code or descriptions of the entire environment.

**Question 1.** If the agent-environment diagram describes a child learning to walk, what exactly is the “Agent” block? Is it the child’s brain, and its body is part of the environment? Is the agent the entire physical child? If the diagram describes a robot, are its sensors part of the environment or the agent?

Neuroscience and psychology ask how animals learn. It is the study of some examples of learning and intelligence. Reinforcement learning asks how we can make an agent that learns. It is the study of learning and intelligence in general (animal, computer, match-boxes, purely theoretical, etc.). In this course we may discuss the relationship between RL and computational neuroscience in one lecture, but in general will not concern ourselves with how animals learn (other than, perhaps, for intuition and motivation).

There are many other fields that are similar and related to RL. Separate research fields often do not communicate much, resulting in different language and approaches. Other notable fields related to RL include operations research.
and control (classical, adaptive, etc.). Although these fields are similar to RL, there are often subtle but impactful differences between the problems studied in these other fields and in RL. Examples include whether the dynamics of the environment are known to the agent a priori (they are not in RL), and whether the dynamics of the environment will be estimated by the agent (many, but not all, RL agents do not directly estimate the dynamics of the environment). There are also many less-impactful differences, like differences in notation (in control, the environment is called the plant, the agent the controller, the reward the (negative) cost, the state the feedback, etc.).

A common misconception is that RL is an alternative to supervised learning—that one might take a supervised learning problem and convert it into an RL problem in order to apply sophisticated RL methods. For example, one might treat the state as the input to a classifier, the action as a label, and the reward as $-1$ if the label is correct and 1 otherwise. Although this is technically possible and a valid use of RL, it should not be done. In a sense, RL should be a last resort—the tool that you use when supervised learning algorithms cannot solve the problem you are interested in. If you have labels for your data, do not discard them and convert the feedback from instructive feedback (telling the agent what label it should have given) to evaluative feedback (telling the agent if it was right or wrong). The RL methods will likely be far worse than standard supervised learning algorithms. However, if you have a sequential problem or a problem where only evaluative feedback is available (or both!), then you cannot apply supervised learning methods and you should use RL.

**Question 2. [Puzzle]** There are 100 pirates. They have 10,000 gold pieces. These pirates are ranked from most fearsome (1) to least fearsome (100). To divide the gold, the most fearsome pirate comes up with a method (e.g., split it evenly, or I get half and the second most fearsome gets the other half). The pirates then vote on this plan. If 50% or more vote in favor of the plan, then that is how the gold is divided. If $>50\%$ vote against the plan, the most fearsome pirate is thrown off the boat and the next most fearsome comes up with a plan, etc. The pirates are perfectly rational. You are the most fearsome pirate. How much of the gold can you get? How?

**Answer 2.** You should be able to keep 9,951 pieces of gold.

If you solved the above puzzle, you very likely did so by first solving easier versions. What if there were only two pirates? What if there were three? This is what we will do in this course. We will study and understand an easier version of the problem and then will build up to more complex and interesting cases over the semester.
2.3 687-Gridworld: A Simple Environment

![Figure 2: 687-Gridworld, a simple example environment we will reference often.](image)

**State**: Position of robot. The robot does not have a direction that it is facing.

**Actions**: Attempt Up, Attempt Down, Attempt Left, Attempt Right. We abbreviate these as: AU, AD, AL, AR.

**Environment Dynamics**: With probability 0.8 the robot moves in the specified direction. With probability 0.05 it gets confused and veers to the right—moves +90° from where it attempted to move (that is, AU results in the robot moving right, AL results in the robot moving up, etc.). With probability 0.05 it gets confused and veers to the left—moves −90° from where it attempted to move (that is, AU results in the robot moving left, AL results in the robot moving down, etc.). With probability 0.1 the robot temporarily breaks and does not move at all. If the movement defined by these dynamics would cause the agent to exit the grid (e.g., move up from state 2) or hit an obstacle (e.g., move right from state 12), then the agent does not move. The robot starts in state 1, and the process ends when the robot reaches state 23.

**Rewards**: The agent receives a reward of −10 for entering the state with the water and a reward of +10 for entering the goal state. Entering any other state results in a reward of zero. If the agent is in the state with the water (state 21) and stays in state 21 for any reason (hitting a wall, temporarily breaking), it counts as “entering” the water state again and results in an additional reward of −10. We use a reward discount parameter (the purpose of which is described later) of γ = 0.9.
2.4 Describing the Agent and Environment Mathematically

In order to reason about learning, we will describe the environment (and soon the agent) using math. Of the many different mathematical models that can be used to describe the environment (POMDPs, DEC-POMDPs, SMDPs, etc.), we will initially focus on Markov decision processes (MDPs). Despite their apparent simplicity, we will see that they capture a wide range of real and interesting problems, including problems that might at first appear to be outside their scope (e.g., problems where the agent makes observations about the state using sensors that might be incomplete and noisy descriptions of the state). Also, a common misconception is that RL is only about MDPs. This is not the case: MDPs are just one way of formalizing the environment of an RL problem.

- An MDP is a mathematical specification of both the environment and what we want the agent to learn.
- Let $t \in \mathbb{N}_{\geq 0}$ be the time step (iteration of the agent-environment loop).
- Let $S_t$ be the state of the environment at time $t$.
- Let $A_t$ be the action taken by the agent at time $t$.
- Let $R_t \in \mathbb{R}$ be the reward received by the agent at time $t$. That is, when the state of the environment is $S_t$, the agent takes action $A_t$, and the environment transitions to state $S_{t+1}$, the agent receives the reward $R_t$. This differs from some other sources wherein this reward is called $R_{t+1}$.

Formally, a finite MDP is a tuple, $(\mathcal{S}, \mathcal{A}, P, d_0, \gamma)$, where:

- $\mathcal{S}$ is the set of all possible states of the environment. The state at time $t$, $S_t$, always takes values in $\mathcal{S}$. For now we will assume that $|\mathcal{S}| < \infty$—that the set of states is finite.
- $\mathcal{A}$ is the set of all possible actions the agent can take. The action at time $t$, $A_t$, always takes values in $\mathcal{A}$. For now we will assume that $|\mathcal{A}| < \infty$.
- $P$ is called the transition function, and it describes how the state of the environment changes.

$$P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1].$$

For all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $s' \in \mathcal{S}$, and $t \in \mathbb{N}_{\geq 0}$:

$$P(s, a, s') := \Pr(S_{t+1} = s' | S_t = s, A_t = a).$$

Hereafter we suppress the sets when writing quantifiers (like $\exists$ and $\forall$)—these should be clear from context. We say that the transition function is deterministic if $P(s, a, s') \in \{0, 1\}$ for all $s, a$, and $s'$. Recall that we will use lower-case letters to denote functions when possible—notice that $P$ is an exception to this rule (due to historical usage and to avoid using $p$, a commonly used symbol otherwise).
\( d_R \) describes how rewards are generated. Intuitively, it is a conditional distribution over \( R_t \) given \( S_t, A_t, \) and \( S_{t+1} \). For now we assume that the rewards are bounded—that \( |R_t| \leq R_{\text{max}} \) always, for all \( t \in \mathbb{N}_{\geq 0} \) and some constant \( R_{\text{max}} \in \mathbb{R} \).

- \( R \) is a function called the reward function, which is implicitly defined by \( d_R \). Other sources often define an MDP to contain \( R \) rather than \( d_R \). Formally

\[
R : S \times A \to \mathbb{R},
\]

and

\[
R(s,a) := \mathbb{E}[R_t|S_t = s, A_t = a],
\]

for all \( s, a, \) and \( t \). Although the reward function, \( R \), does not precisely define how the rewards, \( R_t \), are generated (and thus a definition of an MDP with \( R \) in place of \( d_R \) would in a way be incomplete), it is often all that is necessary to reason about how an agent should act. Like \( P \), notice that \( R \) is a function despite being a capital letter. This is also due to a long history of this notation, and also because we will use \( r \) to denote a particular reward, e.g., when writing \((s,a,r,s',a')\) later.

- \( d_0 \) is the initial state distribution:

\[
d_0 : S \to [0,1],
\]

and for all \( s \):

\[
d_0(s) = \Pr(S_0 = s).
\]

- \( \gamma \in [0,1] \) is a parameter called the reward discount parameter, and which we discuss later.

Just as we have defined the environment mathematically, we now define the agent mathematically. A policy is a decision rule—a way that the agent can select actions. Formally, a policy, \( \pi \), is a function:

\[
\pi : S \times A \to [0,1],
\]

and for all \( s \in S, a \in A, \) and \( t \in \mathbb{N}_{\geq 0} \),

\[
\pi(s,a) := \Pr(A_t = a|S_t = s).
\]

Thus, a policy is the conditional distribution over actions given the state. That is, \( \pi \) is not a distribution, but a collection of distributions over the action set—one per state. There are an infinite number of possible policies, but a finite number of deterministic policies (policies for which \( \pi(s,a) \in \{0,1\} \) for all \( s \) and \( a \)).
Figure 3: Example of a tabular policy. Each cell denotes the probability of the action (specified by the column) in each state (specified by the row). In this format, Π is the set of all $|S| \times |A|$ matrices with non-negative entries and rows that all sum to one.

denote the set of all policies by Π. Figure 3 presents an example of a policy for 687-Gridworld.

To summarize so far, the interaction between the agent and environment proceeds as follows (where $R_t \sim d_R(S_t, A_t, S_{t+1}, \cdot)$ denotes that $R_t$ is sampled according to $d_R$):

$$S_0 \sim d_0$$

$$A_0 \sim \pi(S_0, \cdot)$$

$$S_1 \sim P(S_0, A_0, \cdot)$$

$$R_0 \sim d_R(S_0, A_0, S_1, \cdot)$$

$$A_1 \sim \pi(S_1, \cdot)$$

$$S_2 \sim P(S_1, A_1, \cdot)$$

$$\ldots$$

In pseudocode:

**Algorithm 1:** General flow of agent-environment interaction.

1. $S_0 \sim d_0$;
2. for $t = 0$ to $\infty$ do
3.   $A_t \sim \pi(S_t, \cdot)$;
4.   $S_{t+1} \sim P(S_t, A_t, \cdot)$;
5.   $R_t \sim d_R(S_t, A_t, S_{t+1}, \cdot)$;

The running of an MDP is also presented as a Bayesian network in Figure 4.

Notice that we have defined rewards so that $R_0$ is the first reward, while Sutton and Barto (1998) define rewards such that $R_1$ is the first reward. We do this because $S_0$, $A_0$, and $t = 0$ are the first state, action, and time, and so $R_0$ will place probabilities on a small number of rewards (often the reward may be a deterministic function of $S_t$, $A_t$, and $S_{t+1}$). Sometimes $d_R$ will characterize a continuous distribution. Defining $d_R$ properly therefore requires the use of measure theory for probability. To keep things simple, we will not do this—we will not define $d_R$ more formally.

<table>
<thead>
<tr>
<th>AU</th>
<th>AD</th>
<th>AL</th>
<th>AR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>5</td>
<td>0.25</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

1Sometimes $d_R$ will place probabilities on a small number of rewards (often the reward may be a deterministic function of $S_t$, $A_t$, and $S_{t+1}$). Sometimes $d_R$ will characterize a continuous distribution. Defining $d_R$ properly therefore requires the use of measure theory for probability. To keep things simple, we will not do this—we will not define $d_R$ more formally.
having $R_1$ be the first reward would be inconsistent. Furthermore, this causes indices to align better later on. However, when comparing notes from the course to the book, be sure to account for this notational discrepancy.

**Agent's goal**: Find a policy, $\pi^*$, called an *optimal policy*. Intuitively, an optimal policy maximizes the expected total amount of reward that the agent will obtain.

**Objective function**: $J : \Pi \to \mathbb{R}$, where for all $\pi \in \Pi$,

$$J(\pi) := \mathbb{E}\left[\sum_{t=0}^{\infty} R_t \middle| \pi\right]. \quad (16)$$

**Note**: Later we will revise this definition—if you are skimming looking for the correct definition of $J$, it is in (18).

**Note**: Expectations and probabilities can be conditioned on events. A policy, $\pi$, is not an event. Conditioning on $\pi$, e.g., when we wrote $|\pi$ in the definition of $J$ above, denotes that all actions (the distributions or values of which are not otherwise explicitly specified) are sampled according to $\pi$. That is, for all $t \in \mathbb{N}_{\geq 0}$, $A_t \sim \pi(S_t, \cdot)$.

**Optimal Policy**: An optimal policy, $\pi^*$, is any policy that satisfies:

$$\pi^* \in \arg\max_{\pi \in \Pi} J(\pi). \quad (17)$$

**Note**: Much later we will define an optimal policy in a different and more strict way.

**Property 1** (Existence of an optimal policy). If $|S| < \infty$, $|A| < \infty$, $R_{\max} < \infty$, and $\gamma < 1$, then an optimal policy exists.$^2$

We will prove Property 1 later.

$^2$The restriction of $\gamma$ is not necessary, but is present because our proof of this property will rely on the assumption that $\gamma < 1$. 

---

Figure 4: Bayesian network depicted the running of an MDP.
Question 3. Is the optimal policy always unique when it exists?

Answer 3. No. For example, in 687-Gridworld (if actions always succeeded), then AD and AR would both be equally "good" in state 1, and so any optimal policy could be modified by shifting probability from AD to AR (or vice versa). No. For example, in 687-Gridworld (all actions always succeed), then AD and AR would both be equally good in state 1, and so any optimal policy could be modified by shifting probability from AD to AR (or vice versa).

Reward Discounting: If you could have one cookie today or two cookies on the last day of class, which would you pick? Many people pick one cookie today when actually presented with these options. This suggests that rewards that are obtained in the distant future are worth less to us than rewards in the near future. The reward discount parameter, \( \gamma \), allows us to encode, within the objective function, this discounting of rewards based on how distant in the future they occur.

Recall that \( \gamma \in [0, 1] \). We redefine the objective function, \( J \), as:

\[
J(\pi) := \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R_t \bigg| \pi \right],
\]

for all \( \pi \in \Pi \). So, \( \gamma < 1 \) means that rewards that occur later are worth less to the agent—the utility of a reward, \( r \), \( t \) time steps in the future is \( \gamma^t r \). Including \( \gamma \) also ensures that \( J(\pi) \) is bounded, and later we will see that smaller values of \( \gamma \) make the MDP easier to solve (solving an MDP refers to finding or approximating an optimal policy).

To summarize, the agent’s goal is to find (or approximate) an optimal policy, \( \pi^* \), as defined in (17), using the definition of \( J \) that includes reward discounting—(18).

Question 4. What is an optimal policy for 687-Gridworld? Is it unique? How does the optimal action in state 20 change if we were to change the value of \( \gamma \)?

Question 5. Consider two MDPs that are identical, except for their initial state distributions, \( d_0 \). Let \( \pi^* \) and \( \mu^* \) be optimal policies for the first and second MDP, respectively. Let \( s^* \in S \) be a state that has a non-zero probability of occurring when using \( \pi^* \) on the first MDP and a non-zero probability of occurring when using \( \mu^* \) on the second MDP. Consider a new policy, \( \pi' \) such that \( \pi'(s, a) = \pi^*(s, a) \) for all \( s \in S \setminus \{s^*\} \) and \( a \in A \) and \( \pi'(s^*, a) = \mu^*(s^*, a) \) for all \( a \in A \). Is \( \pi' \) an optimal policy for the first MDP?
When we introduced 687-Gridworld, we said that the agent-environment interactions terminate when the agent reaches state 23, which we called the goal. This notion of a terminal state can be encoded using our definition of an MDP above. Specifically, we define a terminal state to be any state that always transitions to a special state, \( s_\infty \), called the terminal absorbing state. Once in \( s_\infty \), the agent can never leave (\( s_\infty \) is absorbing)—the agent will forever continue to transition from \( s_\infty \) back into \( s_\infty \). Transitioning from \( s_\infty \) to \( s_\infty \) always results in a reward of zero. Effectively, when the agent enters a terminal state the process ends. There are no more decisions to make (since all actions have the same outcome) or rewards to collect. Thus, an episode terminates when the agent enters \( s_\infty \). Notice that terminal states are optional—MDPs need not have any terminal states. Also, there may be states that only sometimes transition to \( s_\infty \), and we do not call these terminal states. Notice also that \( s_\infty \) is an element of \( S \). Lastly, although terminal states are defined, goal states are not defined—the notion of a goal in 687-Gridworld is simply for our own intuition.

When the agent reaches \( s_\infty \), the current trial, called an episode, ends and a new one begins. This means that \( t \) is reset to zero, the initial state, \( S_0 \), is sampled from \( d_0 \), and the next episode begins (the agent selects \( A_0 \), gets reward \( R_0 \), and transitions to state \( S_1 \)). The agent is notified that this has occurred, since this reset may change its behavior (e.g., it might clear some sort of short-term memory).
CMPSCI 687 Homework 1
Due September 19, 2019, 11:55pm Eastern Time

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. Submissions must be typed (hand written 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 or PyTorch (you may use libraries like numpy and matplotlib though). The automated system will not accept assignments after 11:55pm on September 19. The tex file for this homework can be found here.

Part One: Written (63 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.

2. (15 Points) Given an MDP $M = (S, A, P, d, R, d_0, \gamma)$ and a fixed policy, $\pi$, the probability that the action at time $t = 0$ is $a \in A$ is:

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

Write similar expressions (using only $S, A, P, R, d_0, \gamma$, and $\pi$) for the following problems.

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 $\Pr(a|b)$, where event $a$ occurs before event $b$. For example, it is often difficult to work with an expression like $\Pr(S_0 = s_0|A_0 = a_0)$, but much easier to deal with the 3 terms in $\frac{\Pr(A_0 = a_0|S_0 = s_0) \Pr(S_0 = s_0)}{\Pr(A_0 = a_0)}$. 

17
- The law of total probability: For event $a$, and a set of events $B$,

$$\Pr(a) = \sum_{b \in B} \Pr(b) \Pr(a \mid b)$$

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:

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

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

$$\Pr(S_1 = s) = \sum_{s_0 \in \mathcal{S}} \Pr(S_0 = s_0) \Pr(S_1 = s \mid S_0 = s_0) \quad (20)$$

$$= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \Pr(S_1 = s \mid S_0 = s_0) \quad (21)$$

$$= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \Pr(A_0 = a_0 \mid S_0 = s_0) \quad (22)$$

$$\times \Pr(S_1 = s \mid S_0 = s_0, A_0 = a_0) \quad (23)$$

$$= \sum_{s_0 \in \mathcal{S}} d_0(s_0) \sum_{a_0 \in \mathcal{A}} \pi(s_0, a_0) P(s_0, a_0, s). \quad (24)$$

Problems:

A The probability that the action at time $t = 3$ is either $a \in \mathcal{A}$ or $a' \in \mathcal{A}$, with $a \neq a'$.

B The expected reward at time $t = 6$ given that the action at time $t = 5$ is $a \in \mathcal{A}$ and the state at time $t = 4$ is $s \in \mathcal{S}$.

C The probability that the action at time $t = 16$ is $a' \in \mathcal{A}$ given that the action at time $t = 14$ is $a \in \mathcal{A}$, and the state at time $t = 15$ is $s \in \mathcal{S}$.

D The probability that the initial state was $s \in \mathcal{S}$ given that the action at time $t = 1$ is $a' \in \mathcal{A}$.

E The expected reward at time $t = 3$ given that the initial state is $s \in \mathcal{S}$, the state at time $t = 3$ is $s' \in \mathcal{S}$, and the action at time $t = 4$ is $a' \in \mathcal{A}$.

3. (3 Points) In 687-Gridworld, if we changed how rewards are generated so that hitting a wall (i.e., when the agent would enter an obstacle state, and
is placed back where it started) results in a reward of $-10$, then what is $\mathbb{E}[R_t | S_t = 17, A_t = \text{AL}, S_{t+1} = 17]$?

4. (2 Points) How many deterministic policies are there for an MDP with $|S| < \infty$ and $|A| < \infty$? (You may write your answer in terms of $|S|$ and $|A|$).

5. Give an example of an MDP with $|S| < \infty, |A| = \infty$, and $\gamma < 1$ such that an optimal policy does not exist. Give an example of an MDP with $|S| = \infty, |A| < \infty$, and $\gamma < 1$ such that an optimal policy exists.

6. (3 Points) Read about the Pendulum domain, described in Section 5.1 of this paper (Reinforcement Learning in Continuous Time and Space by Kenji Doya). Consider a variant where the initial state has the pendulum hanging down with zero angular velocity always (a deterministic initial state where the pendulum is hanging straight down with no velocity) and a variant where the initial angle is chosen uniformly randomly in $[-\pi, \pi]$ and the initial velocity is zero. Which variant do you expect an agent to require more episodes to solve? Why? Note: We did not talk about the complexity of solving MDPs in class yet—we want you to provide your best guess here.

7. (1 Point) How many episodes do you expect an agent should need in order to find near-optimal policies for the gridworld and pendulum domains? Note: We did not talk about the complexity of solving MDPs in class yet—we want you to provide your best guess here.

8. (5 Points) Select a problem that we have not talked about in class, where the agent does not make Markovian observations about the world around it. Describe how the environment for this problem can be formulated as an MDP by specifying $(S, A, P, [d_r \text{ or } R], d_0, \gamma)$ (your specifications of these terms may use English rather than math, but be precise).

9. (5 Points) We refer to the discounted sum of rewards, $\sum_{t=0}^{\infty} \gamma^t R_t$, as the return. Let an MDP exist such that it has two optimal policies. Can the expected value of their returns differ? If so, give an example. If not, explain why. Can the variance of their returns differ? If so, give an example. If not, explain why.

10. (2 Points) Consider the one state MDP where in $s_0$ there are three actions, $a_0, a_1, a_2$, and all actions transition to $s_\infty$ with probability $0.5$ and stay
in $s_0$ otherwise. The reward for taking actions $a_0$, $a_1$ are drawn from
the uniform distribution on $[0, 1]$ and the normal distribution $\mathcal{N}(0.5, 1)$,
respectively. The reward for $a_3$ is always 0.25. What are all the optimal
policies of this MDP?

11. (2 Points) Read the Wikipedia page on Markov chains. A state in a Markov
chain is irreducible if it is possible to get to any state from any state. An
MDP is irreducible if the Markov chain associated with every deterministic
policy is irreducible. A state in a Markov chain has period $k$ if every return
to the state must occur in multiples of $k$ time steps. More formally,

$$k = \gcd\{t > 0 : \Pr(S_t = s|S_0 = s) > 0\}.$$  

A Markov chain is aperiodic if the period of every state is $k = 1$. Can an
MDP be aperiodic and not irreducible? If so, give an example. If not,
explain your reasoning.

12. (5 Points) The state of a Markov chain is positive recurrent if the expected
time until the state recurs is finite. A Markov chain is positive recurrent if
all states are positive recurrent. Give an example of an MDP with $|S| > 2$
states that is positive recurrent and aperiodic. For any number of states
$|S|$, can you think of a simple way of defining state transitions such that
the MDP is positive recurrent and aperiodic? Explain your methodology
(a picture might be useful).

13. (1 point) Let a tabular policy representation be used to represent stochastic
policies for an MDP with $n$ states and $m$ actions. What is the sum of
every element in the matrix representation of this policy? Why?

14. (2 Points) Describe a real-world problem and how it can be reasonably
modeled as an MDP where $R_t$ is not a deterministic function of $S_t, A_t,$
and $S_{t+1}$, and is not a bandit problem.

15. (2 Points) If you know $(S, A, P, R, d_0, \gamma)$, can you derive $d_R$? Prove your
answer is correct.

16. (2 Points) If you know $(S, A, P, d_R, d_0, \gamma)$, can you derive $R$? Prove your
answer is correct.

17. (2 Points) Describe a real-world problem and how it can be reasonably
modeled as an MDP where $R_t$ is a deterministic function of $S_t$. 

20
18. (2 Points) Describe a real-world problem and how it can be reasonably modeled as an MDP where the reward function, $R$, would not be known.

19. (2 Points) Describe a real-world problem and how it can be reasonably modeled as an MDP where the transition function, $P$, would be known.

20. (2 Points) Describe a real-world problem and how it can be reasonably modeled as an MDP where the transition function, $P$, would not be known.

Part Two: Programming (25 Points Total)

**More-Watery 687-Gridworld.** For this assignment, we will be working with a slightly modified version of the 687-Gridworld domain described in class and in the class notes. In this new Gridworld, called More-Watery Gridworld, there are two extra water states located in state 7 and state 17, as shown in Figure 5. Implement More-Watery Gridworld.

**Codebase.** We have provided a template for programming the homework on the github repository for this class located here. You do not need to use this template for the assignment. After the due date for this assignment, an example will be posted on this site.

```
+-------------------------+-------------------------+-------------------------+-------------------------+-------------------------+
| Start                   | State 2                 | State 3                 | State 4                 | State 5                 |
| State 6                 | State 8                 | State 9                 | State 10                |
| State 11                | State 12                | Obstacle                | State 13                | State 14                |
| State 15                | State 16                | Obstacle                | State 18                |
| State 19                | State 20                | State 21                | Goal                    |
+-------------------------+-------------------------+-------------------------+-------------------------+-------------------------+
```

Figure 5: A more watery version of 687-Gridworld. Water states are located in state 7, state 17, and state 21.
A (5 Points) Have the agent uniformly randomly select actions. Run 10,000 episodes. Report the mean, standard deviation, maximum, and minimum of the observed discounted returns.

B (5 Points) Find an optimal policy (you may do this any way you choose, including by reasoning through the problem yourself). Report the optimal policy here. Comment on whether it is unique.

C (5 Points) Run the optimal policy that you found in the previous question for 10,000 episodes. Report the mean, standard deviation, maximum, and minimum of the observed discounted returns.

D (5 Points) The distribution of returns is often not normal, thus it cannot be fully characterized by its mean and standard deviation. To provide more information about the performance, the empirical distribution of returns can be reported.

For a random variable, $X$, its cumulative distribution function (CDF), $F_X$, is defined as $F_X(x) := \Pr(X \leq x)$. The empirical CDF, $\hat{F}$, for a sequence of $n$ samples of $X$, $X_1, \ldots, X_n$ is given by the function

$$\hat{F}_n(x) := \frac{1}{n} \sum_{i=1}^{n} 1_{X_i \leq x},$$

where $X_i$ is the $i^{th}$ sample of $X$ and $1_A$ is the indicator function of an event $A$, i.e., $1_A = 1$ if $A$ is true and 0 otherwise.

The quantile function, also referred to as the inverse CDF, is the function $Q(\tau) := \inf\{x \in \mathbb{R} : \tau \leq F_X(x)\}$ for $\tau \in (0, 1)$. The empirical quantile function, $\hat{Q}$, can be constructed by considering the order statistics, $Z_i$, the sorted samples of $X_i$ such that $Z_1 \leq Z_2 \leq \ldots \leq Z_n$. The empirical quantile function is given by

$$\hat{Q}(\tau) := Z_{\lfloor (n+1)\tau \rfloor}.$$

Both the CDF and quantile functions capture all the information about a random variable, but for plotting purposes the quantile function is often preferred. This is because we are interested in maximizing returns, so the quantile function has a more natural interpretation as higher is better.

Plot the distribution of returns for both the random policy and the optimal policy using 10,000 trials each. You must clearly label each line and axis. Additionally, report the random seed used for the experiments.

E (5 Points) Using simulations, empirically estimate the probability that $S_{19} = 21$ given that $S_{8} = 18$ (the state above the goal) when running the uniform random policy. Describe how you estimated this quantity (there is not a typo in this problem, nor an oversight).
Instructions: You have 5 minutes to complete this quiz. This quiz is **closed** notes—do not use your notes or a laptop. Do not discuss problems with your neighbors until after everyone has handed in their quiz.

Use the notation from class. Don’t forget to capitalize your random variables. Presenting equalities that are true is not enough—you must provide the definitions of the symbols on the left.

Fill in the definitions for the following terms (your answer to 5 should be a real number, not math expressions):

[Answers in blue.]

1. \( P(s, a, s') = \Pr(S_{t+1} = s'|S_t = s, A_t = a) \)

2. \( R(s, a) = \mathbb{E}[R_t|S_t = s, A_t = a] \)

3. \( \pi(s, a) = \Pr(A_t = a|S_t = s) \)

4. \( J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t|\pi] \)

5. \( \min_{a \in A} R(s_{\infty}, a) = 0 \)
Consider again the definition of reinforcement learning. Notice the segment “learn from interactions with the environment.” If $P$ and $R$ (or $dR$) are known, then the agent does not need to interact with the environment. E.g., an agent solving 687-Gridworld can plan in its head, work out an optimal policy and execute this optimal policy from this start. This is not reinforcement learning—this is planning. More concretely, in planning problems $P$ and $R$ are known, while in reinforcement learning problems at least $P$ (and usually $R$) is not known by the agent. Instead, the agent must learn by interacting with the environment—taking different actions and seeing what happens. Most reinforcement learning algorithms will not estimate $P$. The environment is often too complex to model well, and small errors in an estimate of $P$ compound over multiple time steps making plans built from estimates of $P$ unreliable. We will discuss this more later.

2.5 Additional Terminology, Notation, and Assumptions

- A **history**, $H_t$, is a recording of what has happened up to time $t$ in an episode:
  \[ H_t := (S_0, A_0, R_0, S_1, A_1, R_1, S_2, \ldots, S_t, A_t, R_t). \]  
  (25)

- A **trajectory** is the history of an entire episode: $H_\infty$.

- The return or discounted return of a trajectory is the discounted sum of rewards: $G := \sum_{t=0}^{\infty} \gamma^t R_t$. So, the objective, $J$, is the expected return or expected discounted return, and can be written as $J(\pi) := E[G | \pi]$.

- The return from time $t$ or discounted return from time $t$, $G_t$, is the discounted sum of rewards starting from time $t$:
  \[ G_t := \sum_{k=0}^{\infty} \gamma^k R_{t+k}. \]

2.5.1 Example Domain: Mountain Car

Environments studied in RL are often called **domains**. One of the most common domains is **mountain car**, wherein the agent is driving a crude approximation of a car. The car is stuck in a valley, and the agent wants to get to the top of the hill in front of the car. However, the car does not have enough power to drive straight up the hill in front, and so it must learn to reverse up the hill behind it before accelerating forwards to climb the hill in front. A diagram of the mountain car environment is depicted in Figure 6.

- **State**: $s = (x, v)$, where $x \in \mathbb{R}$ is the position of the car and $v \in \mathbb{R}$ is the velocity.

- **Actions**: $a \in \{\text{forward, neutral, reverse}\}$. These actions can be renamed to be less unwieldy: $a \in \{0, 1, 2\}$. 

24
Figure 6: Diagram of the mountain car domain.

- **Dynamics**: The dynamics are *deterministic*—taking action $a$ in state $s$ always produces the same state, $s'$. Thus, $P(s, a, s') \in \{0, 1\}$. The dynamics are characterized by:

$$v_{t+1} = v_t + 0.001a - 0.0025\cos(3x)x_{t+1} = x_t + v_{t+1}. \quad (26)$$

If these equations would cause $x_{t+1} < -1.2$ then instead $x_{t+1} = -1.2$ and the velocity is set to zero: $v_{t+1} = 0$. Similarly, if these equations would cause $x_{t+1} > 0.5$, then $x_{t+1} = 0.5$ and the velocity is set to zero: $v_{t+1} = 0$. This simulates inelastic collisions with walls at $-1.2$ and $0.5$.

- **Terminal States**: If $x_t = 0.5$, then the state is terminal (it always transitions to $s_\infty$).

- **Rewards**: $R_t = -1$ always, except when transitioning to $s_\infty$ (from $s_\infty$ or from a terminal state), in which case $R_t = 0$.

- **Discount**: $\gamma = 1.0$.

- **Initial State**: $S_0 = (-0.5, 0)$ deterministically (i.e., $\Pr(S_0 = (-0.5, 0)) = 1$).

**Question 6.** For this problem, what is an English description of the meaning behind a return? What is an episode? What is an optimal policy? How long can an episode be? What is the English meaning of $J(\pi)$?
2.5.2 Markov Property

A seemingly more general non-Markovian formulation for the transition function might be:

\[ P(h, s, a, s') := \Pr(S_{t+1} = s' | H_{t-1} = h, S_t = s, A_t = a). \]  

(27)

The **Markov assumption** is the assumption that \( S_{t+1} \) is conditionally independent of \( H_{t-1} \) given \( S_t \). That is, for all \( h, s, a, s', t: \)

\[ \Pr(S_{t+1} = s' | H_{t-1} = h, S_t = s, A_t = a) = \Pr(S_{t+1} = s' | S_t = s, A_t = a) \]  

(28)

Since we make this Markov assumption, \( P \) as defined earlier completely captures the transition dynamics of the environment, and there is no need for the alternate definition in (27). The Markov assumption is sometimes referred to as the **Markov property** (for example one would usually say that a domain has the Markov property, not that the domain satisfies the Markov assumption). It can also be stated colloquially as: the future is independent of the past given the present.

We also assume that the rewards are Markovian—\( R_t \) is conditionally independent of \( H_{t-1} \) given \( S_t \) (since \( A_t \) depends only on \( S_t \), this is equivalent to assuming that \( R_t \) is conditionally independent of \( H_{t-1} \) given both \( S_t \) and \( A_t \)). While the previous Markov assumptions apply to the environment (and are inherent assumptions in the MDP formulation of the environment), we make an additional Markov assumption about the agent: the agent’s policy is Markovian. That is, \( A_t \) is conditionally independent of \( H_{t-1} \) given \( S_t \).

**Question 7.** Can you give examples of Markovian and non-Markovian environments?

**Question 8.** Is the Markov property a property of the problem being formulated as an MDP or a property of the state representation used when...
To answer this second question, consider whether state transitions are Markovian in mountain car. It should be clear that they are as the domain has been described. What about if the state was $s = (x)$ rather than $s = (x, v)$? You could deduce $v_t$ from the previous state, $x_{t-1}$ and current state, $x_t$, but that would require part of the history before $s_t$. Thus, using $s = (x)$ mountain car is not Markovian. So, the Markov property is really a property of the state representation, not the problem being formulated as an MDP.

Notice that one can always define a Markovian state representation. Let $S_t$ be a non-Markovian state representation. Then $(S_t, H_{t-1})$ is a Markovian state representation. That is, we can include the history within the states in order to enforce the Markov property. This is typically undesirable because the size of the state set grows exponentially with the maximum episode length (a term discussed more later). This trick of adding information into the state is called state augmentation.

References