Table of Contents

1 Syllabus .......................................................... 2
  1.1 Website ....................................................... 2
  1.2 Class ......................................................... 2
  1.3 Book .......................................................... 2
  1.4 Teaching Assistants ......................................... 2
  1.5 Office Hours .................................................. 2
  1.6 Grading ........................................................ 3
  1.7 Pass/Fail ...................................................... 3
  1.8 Late Policy .................................................... 3
  1.9 Disability Services .......................................... 3
  1.10 Cheating ..................................................... 4

2 Introduction ...................................................... 5
  2.1 Notation ....................................................... 5
  2.2 What is Reinforcement Learning (RL)? ....................... 6
  2.3 687-Gridworld: A Simple Environment ....................... 8
  2.4 Describing the Agent and Environment Mathematically ...... 9
1 Syllabus

1.1 Website

Class materials will be posted on Moodle. For quick reference, these course notes will also be hosted here.

1.2 Class

Class will be held on Tuesdays and Thursdays from 4:00pm–5:15pm on Zoom. A link to the Zoom room can be found on Moodle. During lectures I will write on a virtual whiteboard here. If you are attending the lecture live, you can follow along using this link. Lectures will be recorded and the recordings (along with a .pdf of the final whiteboard) will be posted on Moodle.

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 Teaching Assistants

The TAs for this course are Scott Jordan (sjordan@cs.umass.edu) and Yash Chandak (ychandak@cs.umass.edu).

1.5 Office Hours

Office hours will not be recorded. Links to Zoom rooms for office hours can be found on Moodle. Office hours will be held according to the following schedule:

<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>9am–10am</td>
<td>Phil</td>
<td>Yash</td>
<td>Scott</td>
<td>Yash</td>
<td>Scott</td>
</tr>
<tr>
<td>10am–11am</td>
<td>Phil</td>
<td>Yash</td>
<td>Scott</td>
<td>Yash</td>
<td>Scott</td>
</tr>
<tr>
<td>11am–noon</td>
<td>Phil</td>
<td>Yash</td>
<td>Scott</td>
<td>Yash</td>
<td>Scott</td>
</tr>
<tr>
<td>noon–1pm</td>
<td>Yash</td>
<td>Scott</td>
<td>Yash</td>
<td>Scott</td>
<td></td>
</tr>
<tr>
<td>5:15pm–5:30pm</td>
<td>Phil</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Although lectures end at 5:15pm, I will stop recording and open the floor for questions / a brief office hours until 5:30pm after lecture. This is intended for follow-up questions pertaining to the lecture.
1.6 Grading

Your grade will have two components:

1. **Homework Assignments** (80%): There will be frequent homework assignments, both written and programming. All assignments will have an equal weight.

2. **Project** (20%): As reinforcement learning transitions from an academic curiosity to practical tools that you may use in your professional lives, it is critical that we study the safe and responsible use of reinforcement learning methods. This project will simulate a real-world application of reinforcement learning, and your grades will be decreased proportional to the amount of (simulated) harm that your reinforcement learning agents cause. You will be free to apply *any* methods (even if they are not RL!) that you program. Further details will be available when the project is assigned after the most relevant course material has been covered.

There will be no exams. 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-). Some extra credit opportunities may be given. Your grade may be reduced by any amount at my discretion due to inappropriate behavior.

1.7 Pass/Fail

At some time near the end of the semester (likely around the last day of class), you will be given the option to take the class pass/fail rather than for a letter grade. If you plan to take the course pass/fail, keep an eye out for an email from me around the end of the semester with instructions for requesting pass/fail. If you elect pass/fail, you will receive a P if your letter grade would have been a B+ or higher, and you will receive an F if your letter grade would have been a B or lower.

1.8 Late Policy

Deadlines in this course are strict. A submission one minute after the deadline will receive zero credit. You are strongly encouraged to submit hours before any deadline.

1.9 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).
1.10 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 an F for the course.
2 Introduction

This document contains notes related to what we cover in class. This is intended as a replacement for posting student notes each time the course is offered (see, for example, the hand-written notes from the 2017 offering here). This is not meant to be a standalone document like a textbook.

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.

We write \( f : \mathcal{X} \rightarrow \mathcal{Y} \) 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 \(|·|\) 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 := to denote is defined to be. In lecture we may write \( \triangleq \) rather than := since the triangle is easier to see when reading my (sometimes sloppy) handwriting.

If \( f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z} \) for any sets \( \mathcal{X} \), \( \mathcal{Y} \), and \( \mathcal{Z} \), then we write \( f(·,y) \) to denote a function, \( g : \mathcal{X} \rightarrow \mathcal{Z} \), 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.*

–Wikipedia, Sutton and Barto (1998), Phil

![Agent-environment diagram](image)

Figure 1: Agent-environment diagram. Examples of **agents** include a child, dog, robot, program, etc. Examples of **environments** include the world, lab, software environment, etc.

---End of Lecture 1, August 25, 2020---

**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. They are the study of some examples of learning and intelligence. RL 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?
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

**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 from the intended direction—moves $+90^\circ$ 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 $-90^\circ$ 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.

<table>
<thead>
<tr>
<th>Start State 1</th>
<th>State 2</th>
<th>State 3</th>
<th>State 4</th>
<th>State 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 6</td>
<td>State 8</td>
<td>State 9</td>
<td>State 10</td>
<td></td>
</tr>
<tr>
<td>State 11</td>
<td>State 12</td>
<td><strong>Obstacle</strong></td>
<td>State 13</td>
<td>State 14</td>
</tr>
<tr>
<td>State 15</td>
<td>State 16</td>
<td><strong>Obstacle</strong></td>
<td>State 17</td>
<td>State 18</td>
</tr>
<tr>
<td>State 19</td>
<td>State 20</td>
<td>State 22</td>
<td><strong>End</strong></td>
<td>State 23</td>
</tr>
</tbody>
</table>

Figure 2: 687-Gridworld, a simple example environment we will reference often.
The robot does not have a direction that it is facing, only a position indicated by the state number.

**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 $\gamma = 0.9$.

### 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 $R_{t+1}$.

There are many definitions of MDPs used in the literature, which share common terms. In each case an MDP is a tuple. Four examples are:

1. $(S, A, p, R)$
2. $(S, A, p, R, \gamma)$
3. $(S, A, p, R, d_0, \gamma)$
4. $(S, A, p, d_R, d_0, \gamma)$.
We will discuss the differences between these definitions in a moment, but first let’s define each of the terms. Notice that the unique terms in these definitions are: $S, A, p, d_R, R, d_0$, and $\gamma$. We define each of these below:

- $S$ is the set of all possible states of the environment. The state at time $t$, $S_t$, always takes values in $S$. For now we will assume that $|S| < \infty$—that the set of states is finite. We call $S$ the “state set”.

- $A$ is the set of all possible actions the agent can take. The action at time $t$, $A_t$, always takes values in $A$. For now we will assume that $|A| < \infty$.

- $P$ is called the transition function, and it describes how the state of the environment changes.

  \[ P : S \times A \times S \to [0, 1]. \]  

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

  \[ P(s, a, s') \triangleq \Pr(S_{t+1} = s'|S_t = s, A_t = a). \] (2)

  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 $S_{t+1}$. That is, $R_t \sim d_R(S_t, A_t, S_{t+1})$. 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 $R_{\text{max}} \in \mathbb{R}$.\footnote{In the remainder of the course, we will very rarely use $d_R$—typically we will work with $R$.}

- $R$ 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

  \[ R : S \times A \to \mathbb{R}, \] (3)

and

  \[ R(s, a) \triangleq \mathbb{E}[R_t|S_t = s, A_t = a], \] (4)

  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:
  \[
  d_0 : \mathcal{S} \rightarrow [0, 1],
  \]
  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.

Recall now our earlier list of four common ways of defining an MDP. These different definitions vary in how precisely they define the environment. The definition \((\mathcal{S}, \mathcal{A}, p, R, \gamma)\) contains all of the terms necessary for us to reason about optimal behavior of an agent. The definition \((\mathcal{S}, \mathcal{A}, p, R)\) still actually includes \( \gamma \), it just makes it implicit. That is, this definition assumes that \( \gamma \) is still present, but doesn’t write it as one of the terms in the MDP definition. On the other extreme, the definition \((\mathcal{S}, \mathcal{A}, p, d_R, d_0, \gamma)\) fully specifies how the environment behaves.

This distinction is most clear when considering the inclusion of \( d_R \) rather than \( R \). As we will see later, the expected rewards described by \( R \) are all that is needed to reason about what behavior is optimal. However, to fully characterize how rewards are generated in an environment, we must specify \( d_R \).

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:

\[
\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1],
\]

and for all \( s \in \mathcal{S}, a \in \mathcal{A}, \) 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 \( a \)). We denote the set of all policies by \( \Pi \). 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
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, \( \Pi \) is the set of all \(|S| \times |A|\) matrices with non-negative entries and rows that all sum to one.

according to \( d_R \):

\[
\begin{align*}
S_0 & \sim d_0 \\
A_0 & \sim \pi(S_0, \cdot) \\
S_1 & \sim P(S_0, A_0, \cdot) \\
R_0 & \sim d_R(S_0, A_0, S_1, \cdot) \\
A_1 & \sim \pi(S_1, \cdot) \\
S_2 & \sim P(S_1, A_1, \cdot) \\
& \quad \vdots
\end{align*}
\]

In pseudocode:

<table>
<thead>
<tr>
<th>Algorithm 1: General flow of agent-environment interaction.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_0 \sim d_0; )</td>
</tr>
<tr>
<td>( \textbf{for } t = 0 \textbf{ to } \infty \textbf{ do} )</td>
</tr>
<tr>
<td>( A_t \sim \pi(S_t, \cdot); )</td>
</tr>
<tr>
<td>( S_{t+1} \sim P(S_t, A_t, \cdot); )</td>
</tr>
<tr>
<td>( R_t \sim d_R(S_t, A_t, S_{t+1}, \cdot); )</td>
</tr>
</tbody>
</table>

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

Figure 4: Bayesian network depicted the running of an MDP.
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 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 \( \pi \in \Pi \),

\[
J(\pi) := E \left[ \sum_{t=0}^{\infty} R_t \mid \pi \right].
\] (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 \( \mid \pi \) 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 \( t \in \mathbb{N}_0, A_t \sim \pi(S_t, \cdot) \).

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

\[
\pi^* \in \arg \max_{\pi \in \Pi} J(\pi).
\] (17)

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

**Question 3.** Is the optimal policy always unique when it exists?

**Answer:** No. For example, in 687 Gridworld (if actions always succeed), then AD and AR would both be equally “good” in state 1, and so any optimal policy could be modeled by assigning probability \( \frac{1}{2} \) to AD and \( \frac{1}{2} \) to AR, respectively. For example, in 687 Gridworld (if actions always succeed), AD and AR would both be equally “good” in state 1, and so any optimal policy could be modeled by assigning probability \( \frac{1}{2} \) to AD and \( \frac{1}{2} \) to AR, respectively.

**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:

$$J(\pi) := E \left[ \sum_{t=0}^{\infty} \gamma^t R_t \right] \mid \pi,$$  \hspace{1cm} (18)

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 $\gamma$?

References