A1

1 Effect of Effective Horizon [8 pts]

Consider an agent managing inventory for a store, which is represented as an MDP. The stock level s refers to the number of items currently in stock (between 0 and 10, inclusive). At any time, the agent has two actions: sell (decrease stock by one, if possible) or buy (increase stock by one, if possible).

- If $s > 0$ and the agent sells, it receives +1 reward for the sale and the stock level transitions to $s - 1$. If $s = 0$, nothing happens.
- If $s < 9$ and the agent buys, it receives no reward and the stock level transitions to $s + 1$.
- The owner of the store likes to see a fully stocked inventory at the end of the day, so the agent is rewarded with $+100$ if the stock level ever reaches the maximum level $s = 10$.
- $s = 10$ is also a terminal state and the problem ends if it is reached.

The reward function, denoted as $r(s, a, s')$, can be summarized concisely as follows:

- $r(s, \text{sell}, s - 1) = 1$ for $s > 0$ and $r(0, \text{sell}, 0) = 0$
- $r(s, \text{buy}, s + 1) = 0$ for $s < 9$ and $r(9, \text{buy}, 10) = 100$. The last condition indicates that transitioning from $s = 9$ to $s = 10$ (fully stocked) yields $+100$ reward.

The stock level is assumed to always start at $s = 3$ at the beginning of the day. We will consider how the agent's optimal policy changes as we adjust the finite horizon H of the problem. Recall that the horizon H refers to a limit on the number of time steps the agent can interact with the MDP before the episode terminates, regardless of whether it has reached a terminal state. We will explore properties of the optimal policy (the policy that achieves highest episode reward) as the horizon H changes.

Consider, for example, $H = 4$. The agent can sell for three steps, transitioning from $s = 3$ to $s = 2$ to $s = 1$ to $s = 0$ receiving rewards $+1$, $+1$, and $+1$ for each sell action. At the fourth step, the inventory is empty so it can sell or buy, receiving no reward regardless. Then the problem terminates since time has expired.

Questions:

a. Starting from the initial state $s = 3$, is it possible to choose a value of H that results in the optimal policy taking both buy and sell steps during its execution? Explain why or why not. [2 pts]

b. In the infinite-horizon discounted setting, is it possible to choose a fixed value of $\gamma \in [0, 1)$ such that the optimal policy starting from $s = 3$ never fully stocks the inventory? You do not need to propose a specific value, but simply explain your reasoning either way. [2 pts]

c. Does there **ever** exist a γ such that the optimal policy for a MDP with a gamma is the same as a MDP with a finite horizon H? Please give an example of a particular γ if there exists one. [2 pts]

d. Does there **always** exist a γ such that the optimal policy for the MDP with γ is the same as an MDP with finite horizon H? Please provide a discussion (1-2 sentences) describing your reasoning. [2 pts]

Answers:

a. Overall it is. Intuitively if $\gamma = 1$, it is not possible, because the reward at $s = 9$ with action buy has a high reward $+100$, and the median reward $+1$ happens in a "one direction" fashion. There are no ties. With this trend, it is natrual to feel that the agent, starting from $s = 3$, would either always buy to get that high reward, or always sell to get some median rewards. And indeed with the insight of $10 - 3 = 7$, we can divide finite H into 2 cases: $0 \leq H < 7$ and $H \geq 7$. For the first case the agent cannot see the big reward and will always sell, for the second case the agent will always buy until $s = 10$ and stays there.
However, if $\gamma < 1$, it is possible. Because if the weight of the biggest reward is significantly lighter, then the agent will be motivated to sell for some nearer rewards, then buy to reach $s = 10$. For example when $\gamma = 0.5$, we can find $H = 9$ that motivates the agent to sell first then buy until reaching $s = 10$.

Python function for calculation

```
print(f"Gamma = {gamma}, Horizon H = {H}")
print("Sell steps (x) | Total Expected Reward")
print("-" * 35)
for x in range(H + 1):
    val = discounted_reward(x, gamma)
    if val != float('-inf'):
        print(f"{x:^14} | {val:.4f}")
```

```
Gamma = 0.5, Horizon H = 9
Sell steps (x) | Total Expected Reward
-----------------------------------
      0        | 1.5625
      1        | 1.7812
      2        | 1.8906
```

Note that the `Sell steps (0)` above indicates the all buy policy.

b. Yes. Similar with the example proposed in a, if γ is very small then the agent will be attracted by nearer rewards that will be gained by selling, and continue oscillating between selling and buying at low s states, thus never fully stocking.

c. Yes. As an extreme example, for a discounted MDP with infinite horizon we can set $\gamma = 0$, then for a non-discounted MDP with finite horizon we set $H = 1$, then these two MDPs will result in the same optimal strategy.

d. No. The finite horizon MDPs' optimal policies are horizon dependent, but the infinite horizon MDPs' are not, so there are cases where no γ can make an infinite horizon MDP produce the same optimal policies as a finite one. Using the setting in this problem, the optimal policy under $H = 9$ can be either all sell or first sell then buy, but with infinite horizon the optimal policy is fixed with a fixed γ. More

generally speaking, there is not a one-to-one mapping from a finite horizon H to some discount factor γ.

2 Reward Hacking [5 pts]

Q1 illustrates how the particular horizon and discount factor may lead to to very different policies, even with the same reward and dynamics model. This maylead to unintentional reward hacking, where the resulting policy does not match a human stakeholder's intended outcome. This problem asks you to think about an example where reward hacking may occur, introduced by Pan, Bhatia and Steinhardt. Consider designing RL for autonomous cars where the goal is to have decision policies that minimize the mean commute for all drivers (those driven by humans andthose driven by AI). This reward might be tricky to specify (it depends on the destination of each car, etc) but a simpler reward (called the reward "proxy") is to maximize the mean velocity of all cars. Now consider a scenario where there is a single AI car (the red car in the figure) and many cars driven by humans (the grey car).

In this setting, under this simpler "proxy" reward, the optimal policy for the red(AI) car is to park and not merge onto the highway.[1]

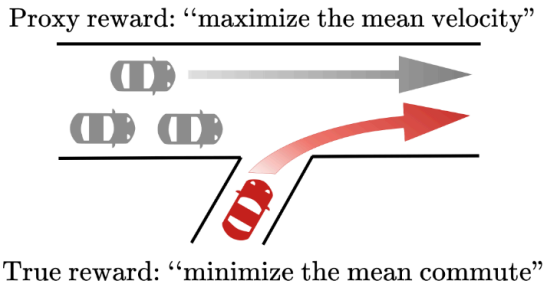

Figure 1: Pan, Bhatia, Steinhardt ICLR 2022; source

Questions:

a. Explain why the optimal policy for the AI car is not to merge onto the highway. [2 pts]

b. Note this behavior is not aligned with the true reward function. Share some ideas about alternate reward functions (that are not minimizing commute) that might still be easier to optimize, but would not result in the AI car never merging. Your answer should be 2-5 sentences and can include equations: there is not a single answer and reasonable solutions will be given full credit. [3 pts]

Answers:

a. Because if the agent just let the red car park and never merge, its velocity stays at 0, and the human drivers maintain higher average speed; while if the agent lets it merge, the average velocity of all cars would be lowered. Following this, the agent finds that not merging the red car is optimal under this flawed proxy reward.

b. Naively I thought about maximizing the minimum velocity of all cars. That does counter the hacking mentioned in subquestion a, but after a while it occured to me that this probably will lead to overly

aggressive policies.
So a better proxy reward may be to make the metric time-dependent, like maximizing time-weighted average velocity of all cars, which is, in non-rigorous but plain words, to only count velocity of cars that has been moving for a while at a time point. In math-y expression:

$$r_t = \frac{\sum_{i=1}^{N} p(i) \cdot v_i(t)}{\sum_{i=1}^{N} p(i)},$$

where i is the index of a car, N is total number of cars, and p is the predicate that:

$$p(i) = \begin{cases} 0, & \text{if car } i \text{ is not moving} \\ 1, & \text{if car } i \text{ is moving} \end{cases}.$$

3 Bellman Residuals and performance bounds [30 pts]

In this problem, we will study value functions and properties of the Bellman backup operator.

Definitions: Recall that a value function is a $|S|$-dimensional vector where $|S|$ is the number of states of the MDP. When we use the term V in these expressions as an ``arbitrary value function'', we mean that V is an arbitrary $|S|$-dimensional vector which need not be aligned with the definition of the MDP at all.

On the other hand, V^π is a value function that is achieved by some policy π in the MDP.

For example, say the MDP has 2 states and only negative immediate rewards. $V = [1, 1]$ would be a valid choice for V even though this value function can never be achieved by any policy π, but we can never have a $V^\pi = [1, 1]$. This distinction between V and V^π is important for this question and more broadly in reinforcement learning.

Properties of Bellman Operators: In the first part of this problem, we will explore some general and useful properties of the Bellman backup operator, which was introduced during lecture. We know that the Bellman backup operator B, defined below is a contraction with the fixed point as V^*, the optimal value function of the MDP. The symbols have their usual meanings. γ is the discount factor and $0 \leq \gamma < 1$. In all parts, $||v|| = \max_s |v(s)|$ is the infinity norm of the vector.

$$(BV)(s) = \max_a \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V(s') \right).$$

We also saw the contraction operator B^π with the fixed point V^π, which is the Bellman backup operator for a particular policy given below:

$$(B^\pi V)(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V(s').$$

In this case, we'll assume π is deterministic, but it doesn't have to be in general. In class, we showed that $||BV - BV'|| \leq \gamma ||V - V'||$ for two arbitrary value functions V and V'.

Questions a-c:

a. Show that the analogous inequality, $||B^\pi V - B^\pi V'|| \leq \gamma ||V - V'||$, also holds. [3 pts].

b. Prove that the fixed point for B^π is unique. Recall that the fixed point is defined as V satisfying $V = B^\pi V$. You may assume that a fixed point exists. Consider proof by contradiction. [3 pts].

c. Suppose that V and V' are vectors satisfying $V(s) \leq V'(s)$ for all s. Show that $B^\pi V(s) \leq B^\pi V'(s)$ for all s. Note that all of these inequalities are element-wise. [3 pts].

Answers:

a. By the definition of Bellman Operator with any value function for a particular policy, we can write the LHS $||B^\pi V - B^\pi V'||$ as

$$\begin{aligned}
||B^\pi V - B^\pi V'|| = \quad & ||r(s,\pi(s)) + \gamma \textstyle\sum_{s' \in S} p(s'|s,\pi(s)) V(s') \\
& - \big(r(s,\pi(s)) + \gamma \textstyle\sum_{s' \in S} p(s'|s,\pi(s)) V'(s')\big)|| \\
= \quad & ||\gamma \textstyle\sum_{s' \in S} p(s'|s,\pi(s)) V(s') - \gamma \sum_{s' \in S} p(s'|s,\pi(s)) V'(s')|| \\
= \quad & \gamma || \textstyle\sum_{s' \in S} \Big[p(s'|s,\pi(s)) \cdot (V(s') - V'(s')) \Big]|| \quad (\text{for } 0 \leq \gamma < 1) \\
\leq \quad & \gamma \cdot \textstyle\sum_{s' \in S} p(s'|s,\pi(s)) \cdot || \sum_{s' \in S} (V(s) - V'(s))|| \quad (\text{triangle inequality}) \\
= \quad & \gamma || \textstyle\sum_{s' \in S} (V(s) - V'(s))|| \quad (\text{sum of probabilities is 1}) \\
\leq \quad & \gamma ||V - V'|| \qquad (\text{by the definition of infinity norm})
\end{aligned}$$

As such, the inequality holds true.

b. Use proof by contradiction. According to the premise, a fixed point exists. Denote it as V_1. Assume there exists another fixed point V_2 that is different than V_1. By the definition of fixed point, we then have $V_1 = B^\pi V_1$ and $V_2 = B^\pi V_2$, which leads to $||B^\pi V_1 - B^\pi V_2|| = ||V_1 - V_2||$. Substitute it into the general inequality derived in subproblem a, we obtain:

$$\begin{aligned}
||B^\pi V_1 - B^\pi V_2|| = \quad & ||V_1 - V_2|| \leq \gamma ||V_1 - V_2|| \\
\Rightarrow \quad & 1 \leq \gamma \quad , \text{by the assumption } V_1 \neq V_2.
\end{aligned}$$

However, this contradicts with $0 \leq \gamma < 1$. Therefore, the original statement is true. ■

c. For verifying inequality, consider using subtraction. By the definition of Bellman Operator, compare the difference between $B^\pi V(s) - B^\pi V'(s)$ for all s:

$$\begin{aligned}
B^\pi V(s) - B^\pi V'(s) = \quad & r(s,\pi(s)) + \gamma \textstyle\sum_{s' \in S} p(s'|s,\pi(s)) V(s') \\
& - \big(r(s,\pi(s)) + \gamma \textstyle\sum_{s' \in S} p(s'|s,\pi(s)) V'(s')\big) \\
= \quad & \gamma \big(\textstyle\sum_{s' \in S} p(s'|s,\pi(s)) V(s') - \sum_{s' \in S} p(s'|s,\pi(s)) V'(s') \big) \\
= \quad & \gamma \cdot \big(\textstyle\sum_{s' \in S} p(s'|s,\pi(s)) \big) \cdot \big(V(s') - V'(s') \big) \\
= \quad & \gamma \cdot \big(V(s') - V'(s') \big) \\
\leq \quad & 0 \qquad (\gamma \in [0,1) \text{ and } V(s') - V'(s') \leq 0 \; \forall s').
\end{aligned}$$

So we have $B^\pi V(s) \leq B^\pi V'(s)$.

Bellman Residuals: Having gained some intuition for value functions and the Bellman operators, we now turn to understanding how policies can be extracted and what their performance might look like. We can extract a greedy policy π from an arbitrary value function V using the equation below.

$$\pi(s) = \arg\max_a [r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)V(s')]$$

It is often helpful to know what the performance will be if we extract a greedy policy from an arbitrary value function. To see this, we introduce the notion of a Bellman residual.

Define the Bellman residual to be $(BV - V)$ and the Bellman error magnitude to be $||BV - V||$.

Questions d-e:

d. For what value function V does the Bellman error magnitude $||BV - V||$ equal 0? Why? [2 pts]

e. Prove the following statements for an arbitrary value function V and any policy π. [5 pts]
Hint: Try leveraging the triangle inequality by inserting a zero term.

$$||V - V^\pi|| \leq \frac{||V - B^\pi V||}{1 - \gamma}$$

$$||V - V^*|| \leq \frac{||V - BV||}{1 - \gamma}.$$

Answers:

d. From previous subproblem b we know that by definition, a fixed point for B^π satisfies $V = B^\pi V$, so for this fixed point value function V, the Bellman error magnitude equal 0. Because we have shown that such a V is unique, this is the only V that makes the Bellman error magnitude equal 0.

e. For the first inequality:

$$||V - V^\pi|| = ||V - B^\pi V + B^\pi V - V^\pi|| \leq ||V - B^\pi V|| + ||B^\pi V - V^\pi||$$

, and

$$\begin{aligned} ||B^\pi V - V^\pi|| = \quad & ||B^\pi V - B^\pi V^\pi|| \quad \text{(by definition of fixed point)} \\ & \leq \gamma ||V - V^\pi|| \quad \text{(by contraction property)} \end{aligned}$$

, so there is

$$||V - V^\pi|| \leq ||V - B^\pi V|| + \gamma ||V - V^\pi||$$

$$\Rightarrow (1 - \gamma)||V - V^\pi|| \leq ||V - B^\pi V||$$

$$\Rightarrow ||V - V^\pi|| \leq \frac{||V - B^\pi V||}{(1 - \gamma)}.$$

Similarly, for the second inequality:

$$||V - V^*|| = ||V - BV + BV - V^*|| \leq ||V - BV|| + ||BV - V^*||$$

, and

$$\begin{aligned} ||BV - V^*|| = &\quad ||BV - BV^*|| \quad (\text{by optimality of } V^*) \\ &\leq \gamma ||V - V^\pi|| \quad (\text{by contraction property}) \end{aligned}$$

, so there is

$$\begin{aligned} &||V - V^*|| \leq ||V - BV|| + \gamma ||V - V^*|| \\ &\Rightarrow (1-\gamma)||V - V^*|| \leq ||V - BV|| \\ &\Rightarrow ||V - V^*|| \leq \frac{||V - BV||}{(1-\gamma)}. \quad \blacksquare \end{aligned}$$

The result you proved in part (e) will be useful in proving a bound on the policy performance in the next few parts. Given the Bellman residual, we will now try to derive a bound on the policy performance, V^π.

f. Let V be an arbitrary value function and π be the greedy policy extracted from V. Let $\epsilon = ||BV - V||$ be the Bellman error magnitude for V. Prove the following for any state s. [5 pts]\ *Hint:* Try to use the results from part (e).

$$V^\pi(s) \geq V^*(s) - \frac{2\epsilon}{1-\gamma}$$

g. Give an example real-world application or domain where having a lower bound on $V^\pi(s)$ would be useful. [2 pt]

h. Suppose we have another value function V' and extract its greedy policy π'. $\|BV' - V'\| = \epsilon = \|BV - V\|$. Does the above lower bound imply that $V^\pi(s) = V^{\pi'}(s)$ at any s? [2 pts]

Answers:

f. Following the hint, try to connect $V^\pi(s)$ and $V^*(s)$ with an arbitrary $V(s)$ and write $||V^\pi(s) - V^*(s)|| = ||V^\pi(s) - V(s) + V(s) - V^*(s)||$. Then we have

$$||V^\pi(s) - V^*(s)|| \leq ||V^\pi(s) - V(s)|| + ||V(s) - V^*(s)|| \quad (\text{triangle inequality}).$$

Integrating the inequalities we proved in e, the RHS would be

$$||V^\pi(s) - V(s)|| + ||V(s) - V^*(s)|| \leq \frac{||V(s) - B^\pi V(s)||}{1-\gamma} + \frac{||V(s) - BV(s)||}{1-\gamma}.$$

Notice that since B is the general Bellman operator, and B^π is Bellman operator of some particular (greedy) policy, the error magnitude with B^π should be covered by ϵ. That is, $||V(s) - B^\pi V(s)|| \leq \epsilon$. Therefore

$$\frac{||V(s) - B^\pi V(s)||}{1-\gamma} + \frac{||V(s) - BV(s)||}{1-\gamma} \leq \frac{2\epsilon}{1-\gamma}.$$

Combining the above we have

$$||V^{\pi}(s) - V^{*}(s)|| \leq \frac{2\epsilon}{1-\gamma}.$$

Since $V^{*}(s)$ is the optimal policy's value function, there must be $V^{\pi} < V^{*}$. Use this fact on the previous inequality to obtain:

$$V^{*}(s) - V^{\pi}(s) \leq \frac{2\epsilon}{1-\gamma}$$

$$\Rightarrow V^{\pi}(s) \geq V^{*}(s) - \frac{2\epsilon}{1-\gamma}. \quad \blacksquare$$

g. Inspired by the self-driving car example in Q1, I think a real-world application can be autonomous driving. When there is guarateed to be a lower bound on an greedy policy, it means we can instruct the agent to find a greedy policy, and the policy's performance would not fall below a certain predicatable threshold, which is a useful reassurance or indicator for significant error.
There must be many other examples in health care, manufacture, security, etc.

h. Probably not. Generally a lower bound on infinity norm cannot directly lead to tight restrictions on the vectors themselves. More specifically, a lower bound on $||BV - V||$ bounds the largest element-wise distance between V and BV, but does not limit the shape of V. There can be a V' that incurs the same Bellman residue but is different in shape, which means the extracted greedy policies π and π' are going to be different.

A little bit more notation: define $V \leq V'$ if $\forall s$, $V(s) \leq V'(s)$.

What if our algorithm returns a V that satisfies $V^{*} \leq V$? I.e., it returns a value function that is better than the optimal value function of the MDP. Once again, remember that V can be any vector, not necessarily achievable in the MDP but we would still like to bound the performance of V^{π} where π is extracted from said V. We will show that if this condition is met, then we can achieve an even tighter bound on policy performance.

Question i:

i. Using the same notation and setup as part (e), if $V^{*} \leq V$, show the following holds for any state s.
Hint: Recall that $\forall \pi,\ V^{\pi} \leq V^{*}$.

$$V^{\pi}(s) \geq V^{*}(s) - \frac{\epsilon}{1-\gamma}, \quad \epsilon = ||BV - V||.$$

Intuition: A useful way to interpret the results from parts (h) (and (i)) is based on the observation that a constant immediate reward of r at every time-step leads to an overall discounted reward of $r + \gamma r + \gamma^2 r + \ldots = \frac{r}{1-\gamma}$. Thus, the above results say that a state value function V with Bellman error magnitude ϵ yields a greedy policy whose reward per step (on average), differs from optimal by at most 2ϵ. So, if we develop an algorithm that reduces the Bellman residual, we're also able to bound the performance of the policy extracted from the value function outputted by that algorithm, which is very useful!

Answer:

i. The second inequality from e gives

$$||V - V^{\pi}|| \leq \frac{||V - BV||}{1-\gamma},$$

and from this combined with the given $V^{\pi} \leq V^* \leq V$, it can be derive that

$$V^{\pi} \geq V - \frac{||V - BV||}{1-\gamma}.$$

Apply the given $V^* \leq V \Rightarrow V \geq V^*$ again on the RHS, we have

$$V^{\pi} \geq V - \frac{||V - BV||}{1-\gamma} \geq V^* - \frac{||V - BV||}{1-\gamma}.$$

Therefore

$$V^{\pi}(s) \geq V^*(s) - \frac{\epsilon}{1-\gamma}, \forall s.$$

4 RiverSwim MDP [25 pts]

Now you will implement value iteration and policy iteration for the RiverSwim environment (see picture below[2]) of Strehl & Littman, 2008.

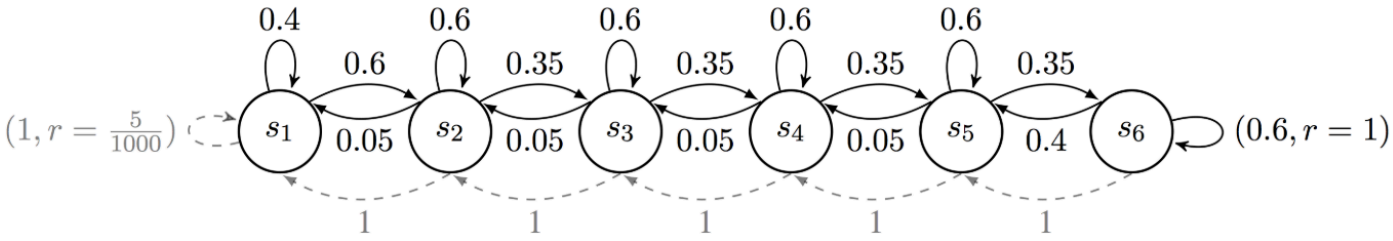

Figure 2: The RiverSwim MDP where dashed and solid arrows represent transitions for the LEFT and RIGHT actions, respectively. The assignment uses a modified, customizable version of what is shown above where there are three different strengths (WEAK, MEDIUM, or STRONG) of the current (transition probabilities for being pushed back or successfully swimming RIGHT).

Setup: This assignment needs to be completed with Python3 and `numpy`.

Submission: There is a `Makefile` provided that will help you submit the assignment. Please run `make clean` followed by `make submit` in the terminal and submit the resulting zip file on Gradescope.

a. **(coding)** Read through `vi_and_pi.py` and implement `bellman_backup`. Return the value associated with a single Bellman backup performed for an input state-action pair. [4 pts]

b. **(coding)** Implement `policy_evaluation`, `policy_improvement` and `policy_iteration` in `vi_and_pi.py`. Return the optimal value function and the optimal policy. [8pts]

c. **(coding)** Implement `value_iteration` in `vi_and_pi.py`. Return the optimal value function and the optimal policy. [8 pts]

d. **(written)** Run both methods on RiverSwim with a WEAK current strength and find the largest discount factor (**only** up to two decimal places) such that an optimal agent starting in the initial far-left state (state s_1 in Figure Figure 2) **does not** swim up the river (that is, does not go RIGHT). Using the value you find, interpret why this behavior makes sense. Now repeat this for RiverSwim with MEDIUM and STRONG currents, respectively. Describe and explain the changes in optimal values and discount factors you obtain both quantitatively and qualitatively. [5 pts]

Sanity Check: For RiverSwim with a discount factor $\gamma = 0.99$ and a WEAK current, the values for the left-most and right-most states (s_1 and s_6 in Figure Figure 2 above) are `30.328` and `36.859` when computed with a tolerance of 0.001. The value functions from VI and PI should be within error tolerance 0.001 of these values. You can use this to verify your implementation. For grading purposes, we shall test your implementation against other hidden test cases as well.

Answers:

a. Implemented.

b. Implemented.

c. Implemented.

d. Running the script gives the threshold discound as follows:

- Weak current: 0.67
- Medium current: 0.77
- Strong current: 0.93.

This makes sense because different discount factors affect the agent's decisions based on how it sees future rewards:

- For small γ, agent cares more about immediate reward, so it would prefer LEFT.
- As γ increases, future reward (at far right) becomes more valuable, so the agent prefers RIGHT.
- Stronger currents means harder to reach the far-right, so we need even higher γ to justify the risk.

Footnotes

a. Interestingly, it turns out that systems that use simpler function representations may reward hackless in this example than more complex representations. See Pan, Bhatia and Steinhardt's paper "The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models" for details. ↩

b. Figure copied from Osband & Van Roy, 2013. ↩