1 Solving MDPs

a) Consider the gridworld MDP for which $Left$ and $Right$ actions are 100% successful. Specifically, the available actions in each state are to move to the neighboring grid squares. From state $a$, there is also an exit action available, which results in going to the terminal state and collecting a reward of 10. Similarly, in state $e$, the reward for the exit action is 1. Exit actions are successful 100% of the time.

Let the discount factor $\gamma = 1$. Fill in the following quantities using the Bellman update rule for value iteration and keeping in mind that $V(s)$ is initialized to 0 for all states $s$.

$$V_0(d) =$$
$$V_1(d) =$$
$$V_2(d) =$$
$$V_3(d) =$$
$$V_4(d) =$$
$$V_5(d) =$$

b) For the same problem as in part a, assume that now the discount factor $\gamma = 0.2$. Fill in the following convergence values.

$$V^*(a) = V_\infty(a) =$$
$$V^*(b) = V_\infty(b) =$$
$$V^*(c) = V_\infty(c) =$$
$$V^*(d) = V_\infty(d) =$$
$$V^*(e) = V_\infty(e) =$$
2 Value Iteration

Consider the following transition diagram, transition function and reward function for an MDP.

<table>
<thead>
<tr>
<th>s</th>
<th>a</th>
<th>s'</th>
<th>T(s,a,s')</th>
<th>R(s,a,s')</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Clockwise</td>
<td>B</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>A</td>
<td>Counterclockwise</td>
<td>B</td>
<td>0.2</td>
<td>-1.0</td>
</tr>
<tr>
<td>A</td>
<td>Counterclockwise</td>
<td>C</td>
<td>0.8</td>
<td>-1.0</td>
</tr>
<tr>
<td>B</td>
<td>Clockwise</td>
<td>A</td>
<td>0.2</td>
<td>1.0</td>
</tr>
<tr>
<td>B</td>
<td>Clockwise</td>
<td>C</td>
<td>0.8</td>
<td>0.0</td>
</tr>
<tr>
<td>B</td>
<td>Counterclockwise</td>
<td>A</td>
<td>1.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>C</td>
<td>Clockwise</td>
<td>A</td>
<td>0.8</td>
<td>-1.0</td>
</tr>
<tr>
<td>C</td>
<td>Clockwise</td>
<td>B</td>
<td>0.2</td>
<td>2.0</td>
</tr>
<tr>
<td>C</td>
<td>Counterclockwise</td>
<td>A</td>
<td>0.2</td>
<td>2.0</td>
</tr>
<tr>
<td>C</td>
<td>Counterclockwise</td>
<td>B</td>
<td>0.8</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Suppose that after iteration $k$ of value iteration we end up with the following values for $V_k$:

<table>
<thead>
<tr>
<th>$V_k (A)$</th>
<th>$V_k (B)$</th>
<th>$V_k (C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.100</td>
<td>0.560</td>
<td>0.680</td>
</tr>
</tbody>
</table>

a) What is $V_{k+1}(A)$?

Now, suppose that we ran value iteration to completion and found the following value function, $V^*$:

<table>
<thead>
<tr>
<th>$V^* (A)$</th>
<th>$V^* (B)$</th>
<th>$V^* (C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.416</td>
<td>0.831</td>
<td>0.974</td>
</tr>
</tbody>
</table>

b) What is $Q^*(A, \text{clockwise})$?

c) What is $Q^*(A, \text{counterclockwise})$?

d) What is the optimal action from state A?
3 Properties

Assuming the MDP has a finite number of actions and states, and that the discount factor satisfies \(0 < \gamma < 1\),

a) True or False: Value iteration is guaranteed to converge.

b) True or False: Value iteration will converge to the same vector of values \((V^*)\) no matter what values we use to initialize \(V\).

4 Policy Evaluation

Consider the gridworld MDP for which Left and Right actions are 100% successful. Specifically, the available actions in each state are to move to the neighboring grid squares. From state \(a\), there is also an exit action available, which results in going to the terminal state and collecting a reward of 10. Similarly, in state \(e\), the reward for the exit action is 1. Exit actions are successful 100% of the time.

\[
\begin{array}{|c|c|c|c|c|}
\hline
10 & & & & 1 \\
\hline
a & b & c & d & e \\
\hline
\end{array}
\]

Let the discount factor \(\gamma = 1\).

a) Consider the policy \(\pi_1\) shown below, and evaluate the following quantities for this policy.

\[
\begin{array}{|c|c|c|c|c|}
\hline
 & & & & \text{exit} \\
\hline
a & b & c & d & e \\
\hline
\end{array}
\]

\(V^{\pi_1}(a) = \)

\(V^{\pi_1}(b) = \)

\(V^{\pi_1}(c) = \)

\(V^{\pi_1}(d) = \)

\(V^{\pi_1}(e) = \)

b) Consider the policy \(\pi_2\) shown below, and evaluate the following quantities for this policy.

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{exit} & & & \text{exit} \\
\hline
a & b & c & d & e \\
\hline
\end{array}
\]

\(V^{\pi_2}(a) = \)

\(V^{\pi_2}(b) = \)

\(V^{\pi_2}(c) = \)

\(V^{\pi_2}(d) = \)

\(V^{\pi_2}(e) = \)
5 Policy Iteration

Consider the gridworld MDP for which Left and Right actions are 100% successful. Specifically, the available actions in each state are to move to the neighboring grid squares. From state a, there is also an exit action available, which results in going to the terminal state and collecting a reward of 10. Similarly, in state e, the reward for the exit action is 1. Exit actions are successful 100% of the time.

Let the discount factor $\gamma = 0.9$. We will execute one round of policy iteration.

a) Step 1: Policy evaluation. Consider the policy $\pi_i$ shown below, and evaluate the following quantities for this policy.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$b) Step 2: Policy improvement. Perform a policy improvement step. The current policy’s values are the ones from Step 1 (so make sure you first correctly answer Step 1 before moving on to Step 2).

$\pi_{i+1}(a) =$

$\pi_{i+1}(b) =$

$\pi_{i+1}(c) =$

$\pi_{i+1}(d) =$

$\pi_{i+1}(e) =$
6 MDPs: Pick a card

You’re playing a game in which in each round the player has the option of drawing a card. In the game all cards have a value between 1 (inclusive) and 6 (inclusive). Each draw costs 1 dollar and the player must draw the very first round. Each time the player draws a card, the player has two possible actions:

1. **Stop**: Stop playing by collecting the dollar value of the card drawn, or
2. **Draw**: Draw again, paying another dollar

Having taken CS 4499/5599 at ISU, you decide to model this problem as an infinite horizon Markov Decision Process (MDP). The player initially starts in state **Start**, where the player only has one possible action: **Draw**. State $s_i$ denotes the state where the drawn card has value $i$. Once a player chooses to **Stop**, the game finishes, causing the player to transition to the **End** state.

a) To solve the problem, you decide to use policy iteration. Your initial policy $\pi$ is shown below. Evaluate the policy at each state, with discount $\gamma = 1$.

<table>
<thead>
<tr>
<th>State</th>
<th>$s_1$</th>
<th>$S2$</th>
<th>$S3$</th>
<th>$S4$</th>
<th>$S5$</th>
<th>$S6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi (s)$</td>
<td><strong>Draw</strong></td>
<td><strong>Draw</strong></td>
<td>Stop</td>
<td>Stop</td>
<td>Stop</td>
<td>Stop</td>
</tr>
<tr>
<td>$V^n (s)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) Having determined the values, perform a policy update to find the new policy $\pi'$. The table below shows the old policy $\pi$ and has filled in parts of the updated policy $\pi'$ for you. If both **Draw** and **Stop** are viable new actions for a state, write down both **Draw/Stop**. As previously, discount $\gamma = 1$.

<table>
<thead>
<tr>
<th>State</th>
<th>$s_1$</th>
<th>$S2$</th>
<th>$S3$</th>
<th>$S4$</th>
<th>$S5$</th>
<th>$S6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi (s)$</td>
<td><strong>Draw</strong></td>
<td><strong>Draw</strong></td>
<td>Stop</td>
<td>Stop</td>
<td>Stop</td>
<td>Stop</td>
</tr>
<tr>
<td>$\pi' (s)$</td>
<td><strong>Draw</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Stop</td>
</tr>
</tbody>
</table>

c) Is $\pi (s)$ from part (a) optimal? Justify your answer.
d) Suppose now that we are working with a discount \( \gamma \in [0, 1) \) and want to run value iteration. Which one of the following statements would hold true at convergence? If none of them are true, write the correct answer below next to “Other”.

- \( V^*(s_i) = \max \left\{ -1 + \frac{i}{6}, \sum_j \gamma V^*(s_j) \right\} \)
- \( V^*(s_i) = \max \left\{ i, \frac{1}{6} \left[-1 + \sum_j \gamma V^*(s_j) \right] \right\} \)
- \( V^*(s_i) = \max \left\{ -\frac{1}{6} + i, \sum_j \gamma V^*(s_j) \right\} \)
- \( V^*(s_i) = \max \left\{ i, -\frac{1}{6} + \sum_j \gamma V^*(s_j) \right\} \)
- \( V^*(s_i) = \frac{1}{6} \sum_j \max \{ i, -1 + \gamma V^*(s_j) \} \)
- \( V^*(s_i) = \sum_k \max \left\{ -1 + i, \sum_j V^*(s_j) \right\} \)
- \( V^*(s_i) = \max \left\{ -1 + i, \frac{1}{6} \gamma V^*(s_j) \right\} \)
- \( V^*(s_i) = \sum_j \max \left\{ i, -1 + \gamma V^*(s_j) \right\} \)
- \( V^*(s_i) = \max \left\{ i, -1 + \frac{2}{6} \sum_j V^*(s_j) \right\} \)
- \( V^*(s_i) = \max \left\{ i, -\frac{1}{6} + \gamma V^*(s_j) \right\} \)
- \( V^*(s_i) = \sum_j \max \left\{ \frac{-i}{6}, -1 + \gamma V^*(s_j) \right\} \)

Other: ________________________________