Qn T

\[ R = -1 \]

1

\[ R = 0 \]

2

\[ R = 1 \sim k \]

1.a

**Action a**

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>S2</td>
<td>0.8</td>
<td>0.2</td>
</tr>
</tbody>
</table>

1.b

**Action b**

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>S2</td>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>

We can do one of two actions in either of states 0, 1, 2. So there are 4 policies.

From 1, the optimal action is going to be b (since it goes to 3 which has 0 value with higher chance). From 2, the intuition is harder. We might think that b is the...
action for \( \pi = 5 \). However, it leaves us in \( 2 \) with higher chance, while \( 1 \) is going to take us out of \( 2 \) with higher chance (and we do want to get out of \( 2 \) fast since it has the worst reward).

\[
V_0 = \begin{bmatrix} -1 & -2 & 0 \end{bmatrix}
\]

\( V_1(3) = 0 \) (since it is in sink state)

\[
V_1(1) \leftarrow R(1) + \max \left\{ \begin{array}{l}
\text{doing a} \\
\quad \begin{bmatrix} 0.2 \times V_0(1) + 0.8 \times V_0(2) \\
\quad \quad 0.9 \times V_0(1) + 0.1 \times V_0(3) \end{bmatrix} \\
\text{doing b} \\
\quad \begin{bmatrix} 0.2 \times -1 + 0.8 \times -2 \\
\quad \quad 0.9 \times -1 + 0.1 \times 0 \end{bmatrix} \\
\end{array} \right\}
\]

\[
\leftarrow -1 + \max \left\{ \begin{bmatrix} -1.8 \\
-0.9 \end{bmatrix} \right\}
\]

\[
\leftarrow \begin{bmatrix} -1.9 \end{bmatrix}
\]

Similarly

\[
V_1(2) \leftarrow -2 + \max \left\{ \begin{array}{l}
\text{doing a} \\
\quad \begin{bmatrix} 0.2 \times -2 + 0.8 \times -1 \\
\quad \quad 0.9 \times -2 + 0.1 \times 0 \end{bmatrix} \\
\text{doing b} \\
\quad \begin{bmatrix} -1.2 \\
-1.8 \end{bmatrix} \\
\end{array} \right\}
\]

\[
\leftarrow -2 - 1.2 = \begin{bmatrix} -3.2 \end{bmatrix}
\]
\[ x_0 = [b \ b \ -1] \]

Computing \( V_0 \)

\[ V_0 (1) = R (1) + 0.9 \ast V_0 (1) + 0.1 \ast V_0 (3) \]

\[ = -1 + 0.9 \ast V_0 (1) + 0 \]

\[ V_0 (1) - 0.9 \ast V_0 (1) = -1 \]

\[ 0.1 \ast V_0 (1) = -1 \]

\[ \boxed{V_0 (1) = -10} \]

Similarly, we can calculate

\[ V_0 (2) = -2 + (0.1 \ast 0) + 0.9 \ast V_0 (2) \]

\[ \boxed{V_0 (2) = -20} \]

Now that we have the \( V_0 \) value vector, we will try to compute the Greedy policy w.r.t. this
for state 1

action a will give expected value
\[0.8x - 20 + 0.2x - 10 = -18\]

action b will give
\[0.1x0 + 0.9x - 10 = -9\]

So the greedy best action is b

\[\pi_1(1) = b\] (no change from \(\pi_0(1)\))

for state 2

action a → \[0.8x - 10 + 0.2x - 20 = -12\]

action b → \[0.1x0 + 0.9x - 20 = -18\]

So greedy best is a

\[\pi_1(2) = a\] (which changes from \(\pi_0(2) = b\))

Now we have to compute value vector corresponding to this policy \(\begin{bmatrix} b & a \end{bmatrix}\)

Evaluating

\[V_1(1) = -1 + (0.1x0) + 0.9xV_1(1)\]
\[V_1(2) = -2 + (0.8xV_1(1)) + 0.2xV_1(2)\]

\[\Rightarrow V_1(1) = -10\]
\[V_1(2) = -12.5\]
Policy improvement

for state 1

action \( a \rightarrow 0.8x - 12.5 + 0.2x - 10 \)

\[ = -12 \]

\( b \rightarrow 0.1x + 0.9x - 10 \)

Action \( b \) still in greedy best

for state 2

action \( a \rightarrow 0.8x - 10 + 0.2x - 12.5 \)

\[ = -10.5 \]

\( b \rightarrow 0.1x + 0.9x - 12.5 \)

Action \( a \) still in greedy best

\( \Pi_2 = [b \; a \; -1] \leftarrow \Pi^* \)

no change in policy. So policy iteration stops.

i.e. Value vector corresponding to \( \Pi^* \)

\[ V(1) = -1 + 0.1x + 0.9x - 10 \]

\[ = -10 \]

\[ V(2) = -2 + 0.8x - 10 + 0.2x - 12.5 \]

\[ = -2 + -10.5 = -12.5 \]

\[ V(3) = 0 \]