MI Sheet 7

Markov Decision Process

17.4 Sometimes MDPs are formulated with a reward function $R(s,a)$ that depends on the action taken or with a reward function $R(s,a,s')$ that also depends on the outcome state.

a. Write the Bellman equations for these formulations.

The most general formulations for the Bellman equation and its policy update assume the reward is given as $R(s,a,s')$ and are

$$U(s) = \max_{a\in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

$$\pi(s) = \operatorname{argmax}_{a\in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

If the reward is $R(s,a)$ then by observing that $\sum_{s'} P(s'|s,a)R(s,a) = R(s,a)$ as R no longer depends on s' we get the formulations

$$U(s) = \max_{a\in A(s)} R(s,a) + \sum_{s'} P(s'|s,a)\gamma U(s')$$

$$\pi(s) = \operatorname{argmax}_{a\in A(s)} R(s,a) + \sum_{s'} P(s'|s,a)\gamma U(s')$$

If the reward is $R(s)$ then by observing that $\operatorname{argmax}_{a\in A(s)} R(s) + \cdots = R(s) + \operatorname{argmax}_{a\in A(s)} \ldots$ as R no longer depends on a we get the formulations (notice we omit R from π as well)

$$U(s) = R(s) + \max_{a\in A(s)} \sum_{s'} P(s'|s,a)\gamma U(s')$$

$$\pi(s) = \operatorname{argmax}_{a\in A(s)} \sum_{s'} P(s'|s,a)\gamma U(s')$$

Although the first formulation will always work, its not simplified for the two special cases. To memorize them all, consider memorizing $\max_{a \in A(s)} \sum_{s'} P(s'|s,a)[\,\gamma U(s')\,]$ then when the reward is given consider if you need to add it inside the sum (square brackets) or add it to the sum or add it to max. In the last case, you can also ignore it in argmax.

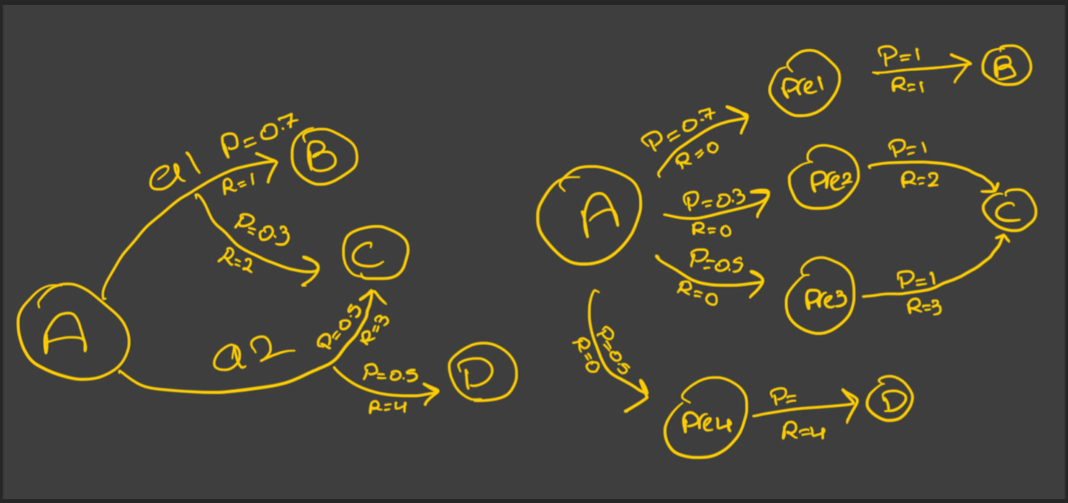
b. Show how an MDP with reward function $R(s,a,s')$ can be transformed into a different MDP with reward function $R(s,a)$, such that optimal policies in the new MDP correspond exactly to optimal policies in the original MDP.

Although using the formulation of $R(s,a,s')$ should work if $R(s,a)$, this unproductive question is asking what should we do to be able to do the opposite.

We want to replace all the $R(s,a,s')$ rewards with $R(s,a)$ rewards. For every (s,a,s') make a new state $pre(s,a,s')$ connected to s such that

- When a is executed at s it leads to $pre(s,a,s')$ and not s' with the same probability $P(s'|s,a)$
  $\Rightarrow$ The reward $R(s,a)$ the agent gets for this is 0
- There is only one action to execute at $pre(s,a,s')$ and it is to go to s' (i.e., $P(s'|pre(s,a,s'),\ go\ to\ s') = 1$)
  $\Rightarrow$ The reward $R(pre(s,a,s'),\ go\ to\ s')$ is $R(s,a,s')$

- If discounted rewards are used then we also/rather need
  $\Rightarrow$ $\gamma = \sqrt{\gamma_{old}}$ and $R(pre(s,a,s'),\ go\ to\ s') = R(s,a,s')/\sqrt{\gamma_{old}}$
  $\Rightarrow$ Makes sense as we a state transition takes place over two steps in our new model and the first reward should involve no γ



Notice that in the new state graph, all rewards are given as $R(s,a)$ without changing the utilities (and hence policies) of states.

c. Now do the same to convert MDPs with $R(s,a)$, into MDPs with $R(s)$.

Although using the formulation of $R(s,a)$ should work if $R(s)$, this unproductive question is asking what should we do to be able to do the opposite.

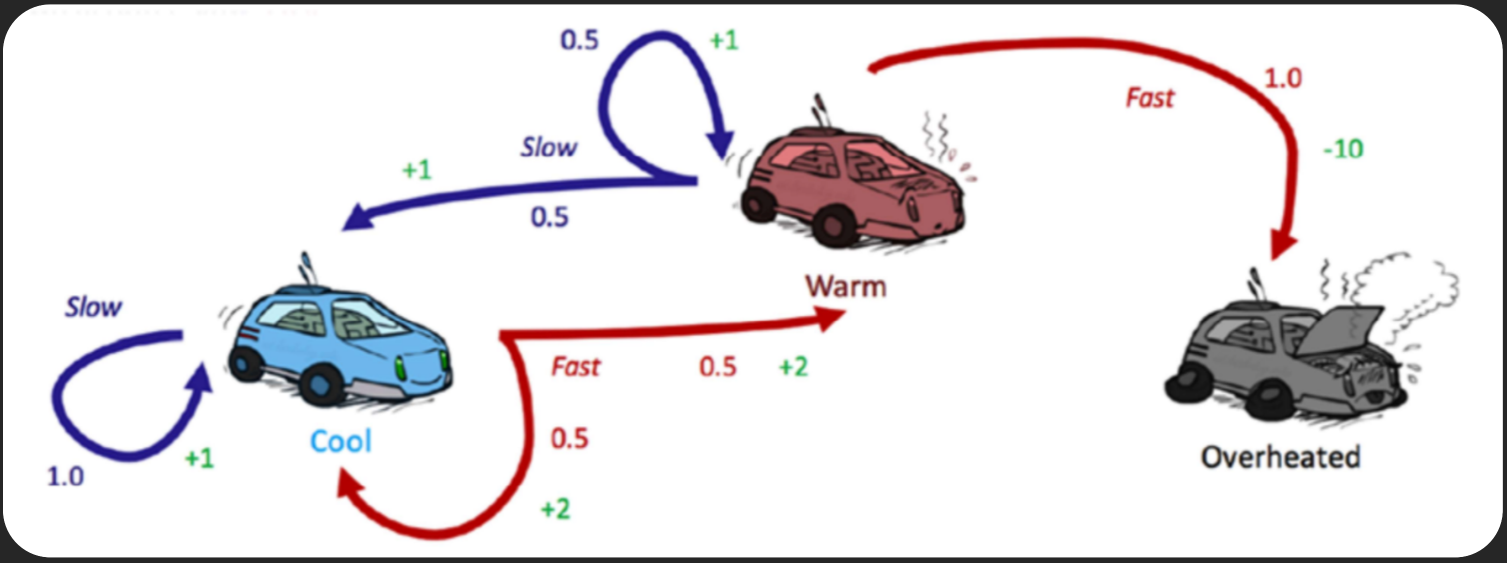
We want to replace all the $R(s,a)$ rewards with $R(s)$ rewards. For every (s,a) make a new state $post(s,a)$ connected to s such that

- When a is executed at s it leads to $post(s,a)$ and not s' with the same probability $P(s'|s,a)$
  $\Rightarrow$ The reward $R(s)$ the agent gets for this is 0
- There is only one action to execute at $post(s,a)$ and it is to go to s' (i.e., $P(s'|post(s,a),\ go\ to\ s') = 1$)
  $\Rightarrow$ The reward $R(post(s,a))$ is $R(s,a)$

- If discounted rewards are used then we also/rather need
  $\Rightarrow \gamma = \sqrt{\gamma_{old}}$ and $R(post(s,a)) = R(s,a)/\sqrt{\gamma_{old}}$
  $\Rightarrow$ Makes sense as we a state transition takes place over two steps in our new model and the first reward should involve no γ

Notice the general insight that in the formulation $R(s)$, we apply the reward upon taking an action at S and not upon descending at S (at the terminal state, an exit action gets automatically executed to give you its reward).

17.4 Run two iterations of value iteration on the following MDP



Use $\gamma = 0.5$ while solving

- Clearly, have three states: Cool, Warm and Overheated; $S \in \{C, W, O\}$
  - $\Rightarrow$ Implies we will solve a nonlinear system of three equations every iteration
  - $\Rightarrow$ The sum over the probabilities will at most have three terms
  - $\Rightarrow$ Our goal is to find the utility of each of these states
- Clearly, have two actions: Fast, Slow; $A \in \{F, S\}$
  - $\Rightarrow$ The max over the actions will at most have two terms
- $P(s^{'}|s,a)$ and $R(s,a,s^{'})$ seem to be given in the transition diagram
  - $\Rightarrow$ Due to $R(s,a,s^{'})$, must use the most general formulation

Assume the initial utilities are

$U(C)$	$U(W)$	$U(O)$
0	0	0

Recall,

$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

In value iteration, we write this equation for each state

$U(C)$	Has action slow with successor C only Has action fast with successors C or W	$\max(\ 1.0(1 + \gamma U(C)), 0.5(2 + \gamma U(C)) + 0.5(2 + \gamma U(W)))$
$U(W)$	Has action slow with successor C or W Has action fast with successors O	$\max(\ 0.5(1 + \gamma U(C)) + 0.5(1 + \gamma U(W)), 1.0(-10 + \gamma U(O)))$
$U(O)$	Has no actions or successors (never change)	$U(O)$

Then in each iteration, plug with the previous utility values to get new utility values

#Iteration 1

$U(C)$	$\max(\ 1.0(1 + \gamma.0), 0.5(2 + \gamma.0) + 0.5(2 + \gamma.0)) = 2$
$U(W)$	$\max(\ 0.5(1 + \gamma.0) + 0.5(1 + \gamma.0), 1.0(-10 + \gamma.0)) = 1$
$U(O)$	0

$U(C)$	$U(W)$	$U(O)$
2	1	0

#Iteration 2

$U(C)$	$\max\big(1.0(1+\gamma.2), 0.5(2+\gamma.2)+0.5(2+\gamma.1)\big) = \max(2, 1.5+1.25) = 2.75$
$U(W)$	$\max\big(0.5(1+\gamma.2)+0.5(1+\gamma.1), 1.0(-10+\gamma.0)\big) = \max(1+0.75, -10) = 1.75$
$U(O)$	0

Once we are done (like now), we find the underlying policy using the general equation (which is always just like making one more iteration with argmax instead of max)

$U(C)$	$U(W)$	$U(O)$
2.75	1.75	0

#Policy Extraction

//Recall, argmax returns the maximizer itself and not the maximum. In our notation above, the first operand was for slow (S) and the second operand was for fast (F)

$\pi(C)$	$\text{argmax}\big(1.0(1+\gamma.2.75), 0.5(2+\gamma.2.75)+0.5(2+\gamma.1.75)\big) = \text{argmax}(2.375, 3.125) = F$
$\pi(W)$	$\text{argmax}\big(0.5(1+\gamma.2.75)+0.5(1+\gamma.1.75), 1.0(-10+\gamma.0)\big) = \text{argmax}(2.125, -10) = S$
$\pi(O)$	0

$\pi(C)$	$\pi(W)$	$\pi(O)$
F	S	$-$

Which makes sense, if the car is cool then the optimal action is to go fast and if its warm then try to slow down. If its overheated, can't do any action just call the mechanic.

17.8 Find the optimal policy for the following grid world by intuition for each of the following values of R: $100, 3, 0\ -3$. Take $\gamma = 0.99$ and let top-right state be the terminal state.

R	-1	+10
-1	-1	-1
-1	-1	-1

100 ←	-1 ←	+10
-1 ↑	-1 ←	-1 ↓
↑ -1	-1 ←	-1 ←

In this case, the agent can maximize their expected utility by avoiding the terminal state and reaching the top-left state. Once it's there it can keep doing actions that will likely not change its state (e.g., left or up) so it earns the 100 rewards as many times as possible. Notice how every single arrow points to the cell that would get us closer to 100, except for the one below +10 as its assumed here that actions can lead us to an orthogonal cell with some low probability and we never want to reach +10.

3 ←	-1 ←	+10
-1 ↑	-1 ←	-1 ↓
↑ -1	-1 ←	-1 ←

In this case, the agent can maximize their expected utility by avoiding the terminal state and reaching the top-left state. Once it's there it can keep doing actions that will likely not change its state (e.g., left or up) so it earns the 3 rewards as many times as possible (observe that γ is almost 1 which is an extremely slow decay; we can easily exceed +10 by repeatedly getting approximately 3 by doing useless actions in the top-left state). This is why the policy is just like the previous example.

0 →	-1 →	+10
-1 ↑	-1 ↑	-1 ↑
↑ -1	↑ -1	↑ -1

In this case, the agent can maximize their expected utility by seeking the terminal state by passing through the top-left state. All arrows point somewhere that gets us closer to the terminal state we can stop suffering -1. In the bottom-left 2x2 square, we always have an option to closer to the terminal state via right or up but in this case up is better because the orthogonal directions (which we may stochastically get to) are closer to the terminal state(right) or make it easier to pass by 0 (left). In the left-most column up is further favored due to the 0.

-3 →	-1 →	+10
-1 →	-1 →	-1 ↑
-1 →	-1 →	↑ -1

In this case, the agent can maximize their expected utility by seeking the terminal state by avoiding the top-left state. What's different this time is that in the bottom-left square everyone changed their opinion to going right. Recall that good options to get to the terminal state where up and right, up is bad here because it either directly goes to -3 or increases the chance of stochastically getting to it later (when top-left was 0, this was a good thing but its now a bad thing).

17.8. For the following 101x3 grid-world, suppose the agent can go up or down at START (S_o) but only right for every state afterwards. Using discounted rewards, find the γ that makes one of them better than the other.

+50	-1	-1	-1	…	-1	-1	-1	-1
S_0								
-50	+1	+1	+1	…	+1	+1	+1	+1

Recall, Geometric Sum Formula: $\sum_{i=1}^{n} a\, r^{i-1} = a(1-r^n)/(1-r)$

Let's consider U(S) in each of the case where the agent chooses up or down:

For Up: $\mathrm{U}(\mathrm{S_o}) = 50 + \sum_{i=1}^{100} \boldsymbol{\gamma^i} R(s_i) = 50 + \sum_{i=1}^{100} \boldsymbol{\gamma^i}. -1 = 5 - + \frac{1-\gamma^{100}}{1-\gamma}.\gamma$

For Down: $\mathrm{U}(\mathrm{S_o}) = -50 + \sum_{i=1}^{100} \boldsymbol{\gamma^i} R(s_i) = -50 + \sum_{i=1}^{100} \boldsymbol{\gamma^i}.1 = -50 + \frac{1-\gamma^{100}}{1-\gamma}.\gamma$

To decide which is better, let's see for what γ is up better than down, i.e., solve:

$$50 + \frac{1-\gamma^{100}}{1-\gamma}\gamma > -50 - \frac{1-\gamma^{100}}{1-\gamma}\gamma$$

Which by solving numerically yields

$$\gamma \ < \ 0.9843976692$$

Hence, up is better for $0 \leq \gamma \ < \ 0.9843976692$ and down is better for $0.9843976692 < \gamma \leq 1$

17.10. Consider the following undiscounted MDP $(\gamma = 1)$

States	S_1	S_2	$S_3(Terminal)$
Reward	-1	-2	0
Action A	Go to S_2 or S_1 with Prob. 0.8 and 0.2 respectively	Go to S_1 or S_2 with Prob. 0.8 and 0.2 respectively	-
Action B	Go to S_3 or S_1 with Prob. 0.1 and 0.9 respectively	Go to S_3 or S_2 with Prob. 0.1 and 0.9 respectively	-

1. Can you qualitatively conclude the optimal policy?

Yes, the policy here corresponds to choosing the optimal actions for S_1 and S_2.

⇒ For S_1 action B is clearly better since we want to reach the terminal state and we want to maximize utility so we we don't want to pass by S_2 (being stuck in S_1 is better than being stuck in S_2)

⇒ For S_2 action A is better since we are very likely to get to S_1 and getting stuck there (-1 each time) is better than getting stuck in S_2

⇒ In summary, in both cases we want to reach the terminal state and in both cases, we have to apply an action (B) that will keep up stuck for some time until we go to the terminal state. In both cases, having this scenario happen in S_1 is better.

2. Solve it via Value Iteration (3 Iterations)

Assume the initial utilities are

$U(S_1)$	$U(S_2)$	$U(S_3)$
0	0	0

Notice that for terminal states, the Bellman equation reduces to "just the reward" so the 0 in $U(S_3)$ is also the final utility (and if the reward at the terminal state was any other x then it would be x). We only need to write the utility equations for nonterminal states.

Recall, the Bellman equation we used for the last problem (most general), the max is over the different actions that could be done in the state and each term in it is a sum for each possible successors if that action is done.

$$U(s) = \max(P(s_1'|s,a_1)[R(s,a_1,s_1') + \gamma U(s_1')] + \cdots, P(s_1'|s,a_2)[R(s,a_2,s_1') + \gamma U(s_1')] + \cdots, \ldots)$$

In our case, the reward is not a function of s' so a so we can rewrite that as

$$U(s) = \mathrm{R(s)} + \max(P(s_1'|s,a_1)[\gamma U(s_1')] + \cdots, P(s_1'|s,a_2)[\gamma U(s_1')] + \cdots, \ldots)$$

Now let's do so for each state

$U(S_1)$	$-1 + \max(\, 0.8(\gamma U(S_2)) + 0.2(\gamma U(S_1)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_1)))$
$U(S_2)$	$-2 + \max(\, 0.8(\gamma U(S_1)) + 0.2(\gamma U(S_2)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_2)))$
$U(S_3)$	0

#Iteration 1

$U(S_1)$	$-1 + \max\big(\, 0.8(\gamma.0) + 0.2(\gamma.0), 0.1(\gamma.0) + 0.9(\gamma.0)\big) = -1$
$U(S_2)$	$-2 + \max\big(\, 0.8(\gamma.0) + 0.2(\gamma.0), 0.1(\gamma.0) + 0.9(\gamma.0)\big) = -2$
$U(S_3)$	0

$U(S_1)$	$U(S_2)$	$U(S_3)$
-1	-2	0

#Iteration 2

$U(S_1)$	$-1 + \max\big(0.8(\gamma.-2) + 0.2(\gamma.-1), 0.1(\gamma.0) + 0.9(\gamma.-1)\big) = -1 + \max(-1.8, -0.9) = -1.9$
$U(S_2)$	$-2 + \max\big(0.8(\gamma.-1) + 0.2(\gamma.-2), 0.1(\gamma.0) + 0.9(\gamma.-2)\big) = -2 + \max(-1.2, -1.9) = -3.2$
$U(S_3)$	0

$U(S_1)$	$U(S_2)$	$U(S_3)$
-1.9	-3.2	0

#Iteration 3

$U(S_1)$	$-1 + \max\big(0.8(\gamma.-3.2) + 0.2(\gamma.-1.9), 0.1(\gamma.0) + 0.9(\gamma.-1.9)\big) = -1 + \max(-2.94, -1.71) = -2.71$
$U(S_2)$	$-2 + \max\big(0.8(\gamma.-1.9) + 0.2(\gamma.-3.2), 0.1(\gamma.0) + 0.9(\gamma.-3.2)\big) = -2 + \max(-2.16, -2.88) = -4.16$
$U(S_3)$	0

$U(S_1)$	$U(S_2)$	$U(S_3)$
-2.71	-4.16	0

#Policy Extraction

$\pi(S_1)$	$\operatorname{argmax}\big(0.8(\gamma.-4.16) + 0.2(\gamma.-2.71), 0.1(\gamma.0) + 0.9(\gamma.-2.71)\big) = \operatorname{argmax}(-3.87, -2.44) = B$
$\pi(S_2)$	$\operatorname{argmax}\big(0.8(\gamma.-2.71) + 0.2(\gamma.-4.16), 0.1(\gamma.0) + 0.9(\gamma.-4.16)\big) = \operatorname{argmax}(-2.99, -3.74) = A$
$\pi(S_3)$	0

$\pi(S_1)$	$\pi(S_2)$	$\pi(S_3)$
B	A	$-$

Which indeed is the optimal policy.

2. Solve it via Policy Iteration (Until convergence)

Recall, the difference between policy and value iteration is

$\Rightarrow$ When we write the utility equations we don't max over actions, we only write the term decided by the current policy and we don't plug old utilities, we solve the system of equations. (Evaluation)

$\Rightarrow$ We update the policy with every iteration (Improvement)

Given is that the initial policy is b for all states (if not given, its known to just start randomly).

$\pi(S_1)$	$\pi(S_2)$	$\pi(S_3)$
B	B	$-$

In each iteration, we will choose one of the terms in max only and write it with U as an unknown then solve the resulting system of equations

$U(S_1)$	$-1 + \max(0.8(\gamma U(S_2)) + 0.2(\gamma U(S_1)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_1)))$
$U(S_2)$	$-2 + \max(0.8(\gamma U(S_1)) + 0.2(\gamma U(S_2)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_2)))$
$U(S_3)$	0

Then we will apply policy improvement to get a new policy

$\pi(S_1)$	$\text{argmax}(0.8(\gamma U(S_2)) + 0.2(\gamma U(S_1)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_1)))$
$\pi(S_2)$	$\text{argmax}(0.8(\gamma U(S_1)) + 0.2(\gamma U(S_2)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_2)))$
$\pi(S_3)$	$-$

#Iteration 1

Since, the policy is B for both we write the second term in each only (let's also plug for $U(S_3) = 0$)

$U(S_1)$	$U(S_1) = -1 + 0.9(\gamma U(S_1))$	-10
$U(S_2)$	$U(S_2) = -2 + 0.9(\gamma U(S_2))$	-20
$U(S_3)$	0	0

$\pi(S_1)$	$\text{argmax}(0.8(\gamma U(S_2)) + 0.2(\gamma U(S_1)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_1))) = \text{argmax}(-18, -9) = B$
$\pi(S_2)$	$\text{argmax}(0.8(\gamma U(S_1)) + 0.2(\gamma U(S_2)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_2))) = \text{argmax}(-12, -18) = A$
$\pi(S_3)$	$-$

#Iteration 1

Since, the policy is B and A for S_1 and S_2 respectively

$U(S_1)$	$U(S_1) = -1 + 0.9(\gamma U(S_1))$	-10
$U(S_2)$	$U(S_2) = -2 + 0.8(\gamma U(S_1)) + 0.2(\gamma U(S_2))$	-12.5
$U(S_3)$	0	0

Note that generally, you'd have to solve the equations simultaneously but we luckily got again one of the equations in one variable only (so we solved it, got -10, plugged in the next to solve the next variable).

$\pi(S_1)$	$\text{argmax}(\ 0.8(\gamma U(S_2)) + 0.2(\gamma U(S_1)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_1))) = \text{argmax}(-12, -9) = B$
$\pi(S_2)$	$\text{argmax}(\ 0.8(\gamma U(S_1)) + 0.2(\gamma U(S_2)), 0.1(\gamma U(S_3)) + 0.9(\gamma U(S_2))) = \text{argmax}(-10.5, -11.25) = A$
$\pi(S_3)$	$-$

Notice that the policy didn't change compared to last iteration which means we have converged (no further iteration will change anything at this point).

2. What happens to policy iteration if we let the initial policy be A for both states? Does discounting help and does the optimal policy depend on it?

#Iteration 1

Since, the policy is A for both we write the first term only in each

$U(S_1)$	$U(S_1) = -1 + 0.8(\gamma U(S_2)) + 0.2(\gamma U(S_1)),$	...
$U(S_2)$	$U(S_2) = -2 + 0.8(\gamma U(S_1)) + 0.2(\gamma U(S_2)),$	...
$U(S_3)$	0	0

To find the utilities, we need to solve the system

$$\begin{aligned} x &= -1 + 0.8y + 0.2x \Rightarrow -0.8x + 0.8y = 1 \\ y &= -2 + 0.8x + 0.2y \Rightarrow 0.8x - 0.8y = 2 \end{aligned}$$

But the system has no solution. These are two parallel straight lines that don't intersect and this is the case due to the symmetry in the transition model when the action is A.

In particular, we have the following matrix equation

$$\begin{pmatrix} -0.8 & 0.8 \\ 0.8 & -0.8 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

And the determinant must be nonzero for it to have a solution (so far its $0.8^2 - 0.8^2$). Although choosing a different initialization to eradicate the symmetry works (just like we did in the previous) however, we can also see if changing γ from 1 to something else would destroy the symmetry and bring the system back to a solvable one.

Rewriting the system like we did before but this time not setting $\gamma = 1$ yields

$$\begin{aligned} x &= -1 + 0.8\gamma y + 0.2\gamma x \Rightarrow (0.2\gamma - 1)x + 0.8\gamma y = 1 \\ y &= -2 + 0.8\gamma x + 0.2\gamma y \Rightarrow 0.8\gamma x + (0.2\gamma - 1)y = 2 \end{aligned}$$

Which is equivalent in the matrix form to

$$\begin{pmatrix} (0.2\gamma - 1) & 0.8\gamma \\ 0.8\gamma & (0.2\gamma - 1) \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

This has a solution if the determinant is nonzero,

$$(0.2\gamma - 1)^2 - (0.8\gamma)^2 \neq 0 \Rightarrow 0.2\gamma - 1 \neq \pm 0.8\gamma \;\Rightarrow\; \gamma \neq -1.67 \; and \;\; \gamma \neq 1$$

The former is impossible anyway as $0 \leq \gamma \leq 1$ and the latter means that any $0 \leq \gamma < 1$ would work. That is, if we set γ as any other value we can solve the system

So in essence,

- ⇒ If the initial policy is initialized that way, we can't use policy iteration as we won't be able to get by the first iteration.
- ⇒ Yes, discounting helps. Any other value would have been fine
- ⇒ Yes, the optimal policy in general changes if γ is changed. The lower γ is, the more the agent will favor nearby than farther states even if they have less reward.

Thank you <3

Thanks for all those that commented on errata in the previous version, all of it has hopefully been fixed