CS 188
Summer 2023

Discussion 5C Solutions

Q1. How do you Value It(eration)?

(a) Fill out the following True/False questions.

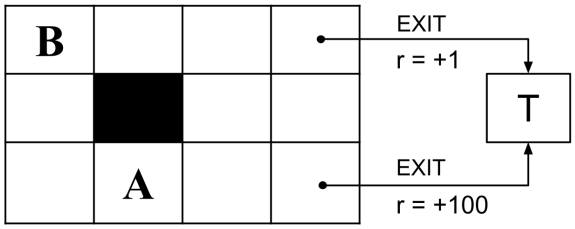
(i) ● True ○ False: Let A be the set of all actions and S the set of states for some MDP. Assuming that $|A| \ll |S|$, one iteration of value iteration is generally faster than one iteration of policy iteration that solves a linear system during policy evaluation. One iteration of value iteration is $O(|S|^2|A|)$, whereas one iteration of policy iteration is $O(|S|^3)$, so value iteration is generally faster when $|A| \ll |S|$

(ii) ○ True ● False: For any MDP, changing the discount factor does not affect the optimal policy for the MDP. Consider an infinite horizon setting where we have 2 states A, B, where we can alternate between A and B forever, gaining a reward of 1 each transition, or exit from B with a reward of 100. In the case that $\gamma = 1$, the optimal policy is to forever oscillate between A and B. If $\gamma = \frac{1}{2}$, then it is optimal to exit.

The following problem will take place in various instances of a grid world MDP. Shaded cells represent walls. In all states, the agent has available actions $\uparrow, \downarrow, \leftarrow, \rightarrow$. Performing an action that would transition to an invalid state (outside the grid or into a wall) results in the agent remaining in its original state. In states with an arrow coming out, the agent has an *additional* action $EXIT$. In the event that the $EXIT$ action is taken, the agent receives the labeled reward and ends the game in the terminal state T. Unless otherwise stated, all other transitions receive no reward, and all transitions are deterministic.

For all parts of the problem, assume that value iteration begins with all states initialized to zero, i.e., $V_0(s) = 0 \ \forall s$. **Let the discount factor be $\gamma = \frac{1}{2}$ for all following parts.**

(b) Suppose that we are performing value iteration on the grid world MDP below.



(i) Fill in the optimal values for A and B in the given boxes.

$V^*(A)$: 25 $V^*(B)$: $\frac{25}{8}$

(ii) After how many iterations k will we have $V_k(s) = V^*(s)$ for all states s? If it never occurs, write "never". Write your answer in the given box.

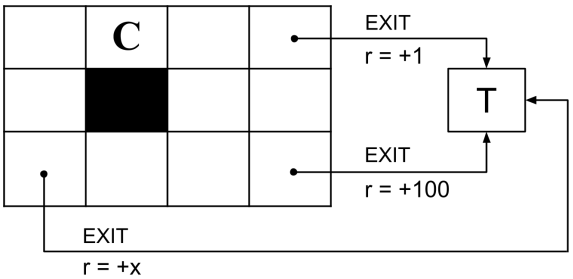
6

(iii) Suppose that we wanted to re-design the reward function. For which of the following new reward functions would the optimal policy **remain unchanged**? Let $R(s, a, s')$ be the original reward function.

- [x] $R_1(s, a, s') = 10R(s, a, s')$
- [x] $R_2(s, a, s') = 1 + R(s, a, s')$
- [x] $R_3(s, a, s') = R(s, a, s')^2$
- [ ] $R_4(s, a, s') = -1$
- [ ] None

R_1: Scaling the reward function does not affect the optimal policy, as it scales all Q-values by 10, which retains ordering
R_2: Since reward is discounted, the agent would get more reward exiting then infinitely cycling between states
R_3: The only positive reward remains to be from exiting state +100 and +1, so the optimal policy doesn't change
R_4: With negative reward at every step, the agent would want to exit as soon as possible, which means the agent would not always exit at the bottom-right square.

(c) For the following problem, we add a new state in which we can take the $EXIT$ action with a reward of $+x$.



(i) For what values of x is it *guaranteed* that our optimal policy π^* has $\pi^*(C) = \leftarrow$? Write ∞ and $-\infty$ if there is no upper or lower bound, respectively. Write the upper and lower bounds in each respective box.

50	$< x <$	∞

We go left if $Q(C, \leftarrow) > Q(C, \rightarrow)$. $Q(C, \leftarrow) = \frac{1}{8}x$, and $Q(C, \rightarrow) = \frac{100}{16}$. Solving for x, we get $x > 50$.

(ii) For what values of x does value iteration take the **minimum** number of iterations k to converge to V^* for all states? Write ∞ and $-\infty$ if there is no upper or lower bound, respectively. Write the upper and lower bounds in each respective box.

50	$\leq x \leq$	200

The two states that will take the longest for value iteration to become non-zero from either $+x$ or $+100$, are states C, and D, where D is defined as the state to the right of C. C will become nonzero at iteration 4 from $+x$, and D will become nonzero at iteration 4 from $+100$. We must bound x so that the optimal policy at D does not choose to go to $+x$, or else value iteration will take 5 iterations. Similar reasoning for D and $+x$. Then our inequalities are $\frac{1}{8}x \geq \frac{100}{16}$ and $\frac{1}{16}x \leq \frac{100}{8}$. Simplifying, we get the following bound on x: $50 \leq x \leq 200$

(iii) Fill the box with value k, the **minimum** number of iterations until V_k has converged to V^* for all states.

4

See the explanation for the part above

Q2. MDPs: Dice Bonanza

A casino is considering adding a new game to their collection, but need to analyze it before releasing it on their floor. They have hired you to execute the analysis. On each round of the game, the player has the option of rolling a fair 6-sided die. That is, the die lands on values 1 through 6 with equal probability. Each roll costs 1 dollar, and the player **must** roll the very first round. Each time the player rolls the die, the player has two possible actions:

1. *Stop*: Stop playing by collecting the dollar value that the die lands on, or
2. *Roll*: Roll again, paying another 1 dollar.

Having taken CS 188, you decide to model this problem using an infinite horizon Markov Decision Process (MDP). The player initially starts in state *Start*, where the player only has one possible action: *Roll*. State s_i denotes the state where the die lands on i. Once a player decides to *Stop*, the game is over, transitioning the player to the *End* state.

(a) In solving this problem, you consider using policy iteration. Your initial policy π is in the table below. Evaluate the policy at each state, with $\gamma = 1$.

State	s_1	s_2	s_3	s_4	s_5	s_6
$\pi(s)$	*Roll*	*Roll*	*Stop*	*Stop*	*Stop*	*Stop*
$V^{\pi}(s)$	3	3	3	4	5	6

We have that $s_i = i$ for $i \in \{3, 4, 5, 6\}$, since the player will be awarded no further rewards according to the policy. From the Bellman equations, we have that $V(s_1) = -1 + \frac{1}{6}(V(s_1) + V(s_2) + 3 + 4 + 5 + 6)$ and that $V(s_2) = -1 + \frac{1}{6}(V(s_1) + V(s_2) + 3 + 4 + 5 + 6)$. Solving this linear system yields $V(s_1) = V(s_2) = 3$.

(b) Having determined the values, perform a policy update to find the new policy π'. The table below shows the old policy π and has filled in parts of the updated policy π' for you. If both *Roll* and *Stop* are viable new actions for a state, write down both *Roll*/*Stop*. In this part as well, we have $\gamma = 1$.

State	s_1	s_2	s_3	s_4	s_5	s_6
$\pi(s)$	*Roll*	*Roll*	*Stop*	*Stop*	*Stop*	*Stop*
$\pi'(s)$	*Roll*	*Roll*	*Roll*/*Stop*	*Stop*	*Stop*	*Stop*

For each s_i in part (a), we compare the values obtained via Rolling and Stopping. The value of Rolling for each state s_i is $-1 + \frac{1}{6}(3 + 3 + 3 + 4 + 5 + 6) = 3$. The value of Stopping for each state s_i is i. At each state s_i, we take the action that yields the largest value; so, for s_1 and s_2, we Roll, and for s_4 and s_5, we stop. For s_3, we Roll/Stop, since the values from Rolling and Stopping are equal.

(c) Is $\pi(s)$ from part (a) optimal? Explain why or why not.
Yes, the old policy is optimal. Looking at part (b), there is a tie between 2 equally good policies that policy iteration considers employing. One of these policies is the same as the old policy. This means that both new policies are as equally good as the old policy, and policy iteration has converged. Since policy iteration converges to the optimal policy, we can be sure that $\pi(s)$ from part (a) is optimal.

(d) Suppose that we were now working with some $\gamma \in [0, 1)$ and wanted to run **value iteration**. Select the **one** statement that would hold true at convergence, or write the correct answer next to Other if none of the options are correct.

○ $V^*(s_i) = \max\left\{-1 + \frac{i}{6} \ , \ \sum_j \gamma V^*(s_j)\right\}$

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

○ $V^*(s_i) = \max\left\{-\frac{1}{6} + i \ , \ \sum_j \gamma V^*(s_j)\right\}$

○ $V^*(s_i) = \max\left\{i \ , \ -\frac{1}{6} + \sum_j \gamma V^*(s_j)\right\}$

○ $V^*(s_i) = \frac{1}{6} \cdot \sum_j \max\{i \ , \ -1 + \gamma V^*(s_j)\}$

○ $V^*(s_i) = \frac{1}{6} \cdot \sum_j \max\left\{-1 + i \ , \ \sum_k V^*(s_j)\right\}$

○ $V^*(s_i) = \sum_j \max\left\{-1 + i \ , \ \frac{1}{6} \cdot \gamma V^*(s_j))\right\}$

○ $V^*(s_i) = \sum_j \max\left\{\frac{i}{6} \ , \ -1 + \gamma V^*(s_j)\right\}$

● $V^*(s_i) = \max\left\{i \ , \ -1 + \frac{\gamma}{6} \sum_j V^*(s_j)\right\}$

○ $V^*(s_i) = \sum_j \max\left\{i \ , \ -\frac{1}{6} + \gamma V^*(s_j)\right\}$

○ $V^*(s_i) = \sum_j \max\left\{\frac{-i}{6} \ , \ -1 + \gamma V^*(s_j)\right\}$

○ Other