CS 188 Summer 2019 — Introduction to Artificial Intelligence — Written HW 3

Due: Friday 7/12/2019 at 11:59pm (submit via Gradescope).

Policy: Can be solved in groups (acknowledge collaborators) but must be written up individually

Submission: Your submission should be a PDF that matches this template. Each page of the PDF should align with the corresponding page of the template (page 1 has name/collaborators, question 1 begins on page 2, etc.). **Do not reorder, split, combine, or add extra pages.** The intention is that you print out the template, write on the page in pen/pencil, and then scan or take pictures of the pages to make your submission. You may also fill out this template digitally (e.g. using a tablet.)

First name	
Last name	
SID	
Collaborators	

Q1. 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*
$V^\pi(s)$						

(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*					*Stop*

(c) Is $\pi(s)$ from part (a) optimal? Explain why or why not.

(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.

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

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

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

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

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

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

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

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

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

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

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

○ Other ____

Q2. Bellman Equations for the Post-Decision State

Consider an infinite-horizon, discounted MDP (S, A, T, R, γ). Suppose that the transition probabilities and the reward function have the following form:

$$T(s, a, s') = P(s'|f(s, a)), \qquad R(s, a, s') = R(s, a)$$

Here, f is some deterministic function mapping $S \times A \to Y$, where Y is a set of states called *post-decision states*. We will use the letter y to denote an element of Y, i.e., a post-decision state. In words, the state transitions consist of two steps: a deterministic step that depends on the action, and a stochastic step that does not depend on the action. The sequence of states (s_t), actions (a_t), post-decision-states (y_t), and rewards (r_t) is illustrated below.

$$\begin{tikzcd}
(s_0, a_0) \arrow[r, "f"] \arrow[dr, dotted] & y_0 \arrow[r, "P"] & (s_1, a_1) \arrow[r, "f"] \arrow[dr, dotted] & y_1 \arrow[r, "P"] & (s_2, a_2) \arrow[r, "f"] \arrow[dr, dotted] & \cdots \\
& r_0 & & r_1 & & r_2
\end{tikzcd}$$

You have learned about $V^\pi(s)$, which is the expected discounted sum of rewards, starting from state s, when acting according to policy π.

$$V^\pi(s_0) = E\left[R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots\right] \qquad \text{given } a_t = \pi(s_t) \text{ for } t = 0, 1, 2, \dots$$

$V^*(s)$ is the value function of the optimal policy, $V^*(s) = \max_\pi V^\pi(s)$.

This question will explore the concept of computing value functions on the post-decision-states y. [1]

$$W^\pi(y_0) = E\left[R(s_1, a_1) + \gamma R(s_2, a_2) + \gamma^2 R(s_3, a_3) + \dots\right]$$

We define $W^*(y) = \max_\pi W^\pi(y)$.

(a) Write W^* in terms of V^*.
$W^*(y) =$

- ○ $\sum_{s'} P(s' \mid y) V^*(s')$
- ○ $\sum_{s'} P(s' \mid y)[V^*(s') + \max_a R(s', a)]$
- ○ $\sum_{s'} P(s' \mid y)[V^*(s') + \gamma \max_a R(s', a)]$
- ○ $\sum_{s'} P(s' \mid y)[\gamma V^*(s') + \max_a R(s', a)]$
- ○ None of the above

(b) Write V^* in terms of W^*.
$V^*(s) =$

- ○ $\max_a [W^*(f(s, a))]$
- ○ $\max_a [R(s, a) + W^*(f(s, a))]$
- ○ $\max_a [R(s, a) + \gamma W^*(f(s, a))]$
- ○ $\max_a [\gamma R(s, a) + W^*(f(s, a))]$
- ○ None of the above

[1] In some applications, it is easier to learn an approximate W function than V or Q. For example, to use reinforcement learning to play Tetris, a natural approach is to learn the value of the block pile *after* you've placed your block, rather than the value of the pair (current block, block pile). TD-Gammon, a computer program developed in the early 90s, was trained by reinforcement learning to play backgammon as well as the top human experts. TD-Gammon learned an approximate W function.

(c) Recall that the optimal value function V^* satisfies the Bellman equation:

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left(R(s,a) + \gamma V^*(s')\right),$$

which can also be used as an update equation to compute V^*.

Provide the equivalent of the Bellman equation for W^*.

$W^*(y) =$ __________________________

(d) Fill in the blanks to give a policy iteration algorithm, which is guaranteed return the optimal policy π^*.

- Initialize policy $\pi^{(1)}$ arbitrarily.
- For $i = 1, 2, 3, \ldots$
  - Compute $W^{\pi^{(i)}}(y)$ for all $y \in Y$.
  - Compute a new policy $\pi^{(i+1)}$, where $\pi^{(i+1)}(s) = \arg\max_a$ ____(1)____ for all $s \in S$.
  - If ____(2)____ for all $s \in S$, **return** $\pi^{(i)}$.

Fill in your answers for blanks (1) and (2) below.

(1)
- ○ $W^{\pi^{(i)}}(f(s,a))$
- ○ $R(s,a) + W^{\pi^{(i)}}(f(s,a))$
- ○ $R(s,a) + \gamma W^{\pi^{(i)}}(f(s,a))$
- ○ $\gamma R(s,a) + W^{\pi^{(i)}}(f(s,a))$
- ○ None of the above

(2) ____

Q3. Q-learning

Consider the following gridworld (rewards shown on left, state names shown on right).

Rewards

+10	+1

State names

A	B
G1	G2

From state A, the possible actions are right($\rightarrow$) and down($\downarrow$). From state B, the possible actions are left($\leftarrow$) and down($\downarrow$). For a numbered state (G1, G2), the only action is to exit. Upon exiting from a numbered square we collect the reward specified by the number on the square and enter the end-of-game absorbing state X. We also know that the discount factor $\gamma = 1$, and in this MDP all actions are **deterministic** and always succeed.

Consider the following episodes:

Episode 1 ($E1$)

s	a	s'	r
A	$\downarrow$	$G1$	0
$G1$	exit	X	10

Episode 2 ($E2$)

s	a	s'	r
B	$\downarrow$	$G2$	0
$G2$	exit	X	1

Episode 3 ($E3$)

s	a	s'	r
A	$\rightarrow$	B	0
B	$\downarrow$	$G2$	0
$G2$	exit	X	1

Episode 4 ($E4$)

s	a	s'	r
B	$\leftarrow$	A	0
A	$\downarrow$	$G1$	0
$G1$	exit	X	10

(a) Consider using temporal-difference learning to learn $V(s)$. When running TD-learning, all values are initialized to zero.
For which sequences of episodes, if repeated infinitely often, does $V(s)$ converge to $V^*(s)$ for all states s?

(Assume appropriate learning rates such that all values converge.)
Write the correct sequence under "Other" if no correct sequences of episodes are listed.

☐ $E1, E2, E3, E4$ ☐ $E1, E2, E1, E2$ ☐ $E1, E2, E3, E1$ ☐ $E4, E4, E4, E4$
☐ $E4, E3, E2, E1$ ☐ $E3, E4, E3, E4$ ☐ $E1, E2, E4, E1$

☐ Other ______________________________

(b) Consider using Q-learning to learn $Q(s, a)$. When running Q-learning, all values are initialized to zero.
For which sequences of episodes, if repeated infinitely often, does $Q(s, a)$ converge to $Q^*(s, a)$ for all state-action pairs (s, a)

(Assume appropriate learning rates such that all Q-values converge.)
Write the correct sequence under "Other" if no correct sequences of episodes are listed.

☐ $E1, E2, E3, E4$ ☐ $E1, E2, E1, E2$ ☐ $E1, E2, E3, E1$ ☐ $E4, E4, E4, E4$
☐ $E4, E3, E2, E1$ ☐ $E3, E4, E3, E4$ ☐ $E1, E2, E4, E1$

☐ Other ______________________________

Q4. Reinforcement Learning

Imagine an unknown game which has only two states $\{A, B\}$ and in each state the agent has two actions to choose from: {Up, Down}. Suppose a game agent chooses actions according to some policy π and generates the following sequence of actions and rewards in the unknown game:

t	s_t	a_t	s_{t+1}	r_t
0	A	Down	B	2
1	B	Down	B	-4
2	B	Up	B	0
3	B	Up	A	3
4	A	Up	A	-1

Unless specified otherwise, assume a discount factor $\gamma = 0.5$ and a learning rate $\alpha = 0.5$

(a) Recall the update function of Q-learning is:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'))$$

Assume that all Q-values initialized as 0. What are the following Q-values learned by running Q-learning with the above experience sequence?

$$Q(\text{A}, \text{Down}) = \_\_\_\_\_\_\_\_, \qquad Q(\text{B}, \text{Up}) = \_\_\_\_\_\_\_\_$$

(b) In model-based reinforcement learning, we first estimate the transition function $T(s, a, s')$ and the reward function $R(s, a, s')$. Fill in the following estimates of T and R, estimated from the experience above. Write "n/a" if not applicable or undefined.

$\hat{T}(\text{A}, \text{Up}, \text{A}) = \_\_\_\_\_\_\_\_$, $\hat{T}(\text{A}, \text{Up}, \text{B}) = \_\_\_\_\_\_\_\_$, $\hat{T}(\text{B}, \text{Up}, \text{A}) = \_\_\_\_\_\_\_\_$, $\hat{T}(\text{B}, \text{Up}, \text{B}) = \_\_\_\_\_\_\_\_$

$\hat{R}(\text{A}, \text{Up}, \text{A}) = \_\_\_\_\_\_\_\_$, $\hat{R}(\text{A}, \text{Up}, \text{B}) = \_\_\_\_\_\_\_\_$, $\hat{R}(\text{B}, \text{Up}, \text{A}) = \_\_\_\_\_\_\_\_$, $\hat{R}(\text{B}, \text{Up}, \text{B}) = \_\_\_\_\_\_\_\_$

(c) To decouple this question from the previous one, assume we had **a different experience** and ended up with the following estimates of the transition and reward functions:

s	a	s'	$\hat{T}(s, a, s')$	$\hat{R}(s, a, s')$
A	Up	A	1	10
A	Down	A	0.5	2
A	Down	B	0.5	2
B	Up	A	1	-5
B	Down	B	1	8

(i) Give the optimal policy $\hat{\pi}^*(s)$ and $\hat{V}^*(s)$ for the MDP with transition function $\hat{T}$ and reward function $\hat{R}$. *Hint: for any $x \in \mathbb{R}$, $|x| < 1$, we have $1 + x + x^2 + x^3 + x^4 + \cdots = 1/(1-x)$.*

$\hat{\pi}^*(A) = \_\_\_\_\_\_\_\_$, $\hat{\pi}^*(B) = \_\_\_\_\_\_\_\_$, $\hat{V}^*(A) = \_\_\_\_\_\_\_\_$, $\hat{V}^*(B) = \_\_\_\_\_\_\_\_$.

(ii) If we repeatedly feed this new experience sequence through our Q-learning algorithm, what values will it converge to? Assume the learning rate α_t is properly chosen so that convergence is guaranteed.

- ○ the values found above, $\hat{V}^*$
- ○ the optimal values, V^*
- ○ neither $\hat{V}^*$ nor V^*
- ○ not enough information to determine

Q5. Policy Evaluation

In this question, you will be working in an MDP with states S, actions A, discount factor γ, transition function T, and reward function R.

We have some fixed policy $\pi : S \rightarrow A$, which returns an action $a = \pi(s)$ for each state $s \in S$. We want to learn the Q function $Q^\pi(s,a)$ for this policy: the expected discounted reward from taking action a in state s and then continuing to act according to π: $Q^\pi(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma Q^\pi(s', \pi(s'))]$. The policy π will not change while running any of the algorithms below.

(a) Can we guarantee anything about how the values Q^π compare to the values Q^* for an optimal policy π^*?

- ◯ $Q^\pi(s,a) \leq Q^*(s,a)$ for all s, a
- ◯ $Q^\pi(s,a) = Q^*(s,a)$ for all s, a
- ◯ $Q^\pi(s,a) \geq Q^*(s,a)$ for all s, a
- ◯ None of the above are guaranteed

(b) Suppose T and R are *unknown*. You will develop sample-based methods to estimate Q^π. You obtain a series of *samples* $(s_1, a_1, r_1), (s_2, a_2, r_2), \ldots (s_T, a_T, r_T)$ from acting according to this policy (where $a_t = \pi(s_t)$, for all t).

(i) Recall the update equation for the Temporal Difference algorithm, performed on each sample in sequence:

$$V(s_t) \leftarrow (1-\alpha)V(s_t) + \alpha(r_t + \gamma V(s_{t+1}))$$

which approximates the expected discounted reward $V^\pi(s)$ for following policy π from each state s, for a learning rate α.

Fill in the blank below to create a similar update equation which will approximate Q^π using the samples. You can use any of the terms $Q, s_t, s_{t+1}, a_t, a_{t+1}, r_t, r_{t+1}, \gamma, \alpha, \pi$ in your equation, as well as $\sum$ and max with any index variables (i.e. you could write $\max_a$, or $\sum_a$ and then use a somewhere else), but no other terms.

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\,[\underline{\hspace{6cm}}]$$

(ii) Now, we will approximate Q^π using a linear function: $Q(s,a) = \sum_{i=1}^{d} w_i f_i(s,a)$ for weights $w_1, \ldots, w_d$ and feature functions $f_1(s,a), \ldots, f_d(s,a)$.

To decouple this part from the previous part, use Q_{samp} for the value in the blank in part (i) (i.e. $Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha Q_{samp}$).

Which of the following is the correct sample-based update for each w_i?

- ◯ $w_i \leftarrow w_i + \alpha[Q(s_t, a_t) - Q_{samp}]$
- ◯ $w_i \leftarrow w_i - \alpha[Q(s_t, a_t) - Q_{samp}]$
- ◯ $w_i \leftarrow w_i + \alpha[Q(s_t, a_t) - Q_{samp}]f_i(s_t, a_t)$
- ◯ $w_i \leftarrow w_i - \alpha[Q(s_t, a_t) - Q_{samp}]f_i(s_t, a_t)$
- ◯ $w_i \leftarrow w_i + \alpha[Q(s_t, a_t) - Q_{samp}]w_i$
- ◯ $w_i \leftarrow w_i - \alpha[Q(s_t, a_t) - Q_{samp}]w_i$

(iii) The algorithms in the previous parts (part i and ii) are:

☐ model-based ☐ model-free

Q6. Proofs: Admissibility, Consistency and Graph Search

Here, we will revisit and prove some of the properties of search mentioned in lectures in a more rigorous manner.

The central idea of consistency is that we enforce not only that a heuristic underestimates the total distance to a goal from any given node, but also the cost/weight of each edge in the graph. For graph search to be optimal, we have to make sure that every time we visit a node, it is already the most optimal way to reach that node, since we don't get another chance to expand it with our closed list in place. Hence, it is intuitive to see that, consistent heuristic, a function that underestimates distance of every intermediate "goal" just like how an admissible heuristic underestimates total distance to a goal, is likely to be sufficient for graph search to be optimal when run with A* search. We will prove that for a given search problem, if the consistency constraint is satisfied by a heuristic function h, using $A*$ graph search with h on that search problem will yield an optimal solution.

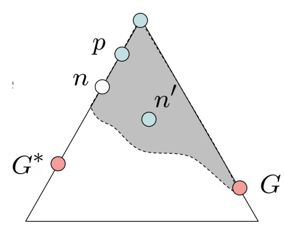
(a) Show that consistency implies admissibility.

(b) Construct a graph and a heuristic such that running A* tree search finds an optimal path while running A* graph search finds a suboptimal one.

(c) Recall the following notations:

- $g(n)$ - The function representing total backwards cost computed by UCS.
- $h(n)$ - The heuristic value function, or estimated forward cost, used by greedy search.
- $f(n)$ - The function representing estimated total cost, used by A* search. $f(n) = g(n) + h(n)$.

Show that the f value constructed with a consistent heuristic never decreases along a path. Specifically, consider a path $p = (s_1, s_2, ..., s_{t-1}, s_t)$, show that $f(s_{i+1}) \geq f(s_i)$. Also, check that this is indeed the case with your example in **(b)**. Hint: use the definition of consistent heuristic!



(d) Consider a scenario where some n on path to $G*$ isn't in queue when we need it, because some worse n' for the same state was dequeued and expanded first. Take the highest such n in tree and let p be the ancestor of n that was on the queue when n' was popped. Prove that p would have been expanded before n' and this scenario would never happen with a consistent heuristic.

(e) Finally, show that an optimal goal $G*$ will always be removed for expansion and returned before any suboptimal goal G with a consistent heuristic.