CS 188 Spring 2019 | Introduction to Artificial Intelligence | Written HW 3

Due: Monday 2/18/2019 at 11:59pm (submit via Gradescope).

Leave self assessment boxes blank for this due date.

Self assessment due: Monday 2/25/2019 at 11:59pm (submit via Gradescope)

For the self assessment, **fill in the self assessment boxes in your original submission** (you can download a PDF copy of your submission from Gradescope). For each subpart where your original answer was correct, write "correct." Otherwise, write and explain the correct answer.

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	Gregory
Last name	Uezono
SID	3031967186
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*
$V^{\pi}(s)$	2.5	2.5	3	4	5	6

$$\frac{0+1+2+3+4+5}{6}$$

Self assessment

If your answer was correct, write "**correct**" above. Otherwise, **write and explain** the correct answer.

(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	stop	stop	stop	*Stop*

Self assessment

If your answer was correct, write **"correct"** above. Otherwise, **write and explain** the correct answer.

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

Yes b/c $\pi(s) = \pi'(s)$ for $\forall s \in S$

Self assessment

If your answer was correct, write **"correct"** above. Otherwise, **write and explain** the correct answer.

(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 $V^*(s_i) = \max\left\{i \ , \ \sum_k (k-1) + \gamma V^*(s_k)\right\}$

Self assessment
If your answer was correct, write "correct" above. Otherwise, write and explain the correct answer.

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.

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

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.

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

$V^*(s)$ 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]

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

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

$V^*(s) = \max_\pi V^\pi(s)$

$= \max\left(\sum T(s,a,s')[R(s,a,s') + \gamma V_k(s')\right)$

(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

Self assessment
If your answer was correct, write "**correct**" above. Otherwise, **write and explain** the correct answer.

[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.

(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

Self assessment
If your answer was correct, write **"correct"** above. Otherwise, **write and explain** the correct answer.

(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^*.

$f(s,a) \rightarrow y$

$W^*(y) = \max_a \sum_{y'} T(y)\,(R(y') + \gamma W^*(y))$

Self assessment
If your answer was correct, write **"correct"** above. Otherwise, **write and explain** the correct answer.

(...) 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) $\pi^{(i+1)} = \pi^{(i)}$

Self assessment
If your answer was correct, write "correct" above. Otherwise, **write and explain** the correct answer.