Name: _____ SUID: _____

Stanford University

AA228/CS238: Decision Making under Uncertainty

Autumn 2020

Prof. Mykel J. Kochenderfer • Remote • email: *mykel@stanford.edu*

MIDTERM 2 **Due date: October 23, 2020 (5pm)**

You have **90 minutes** to complete this exam. This exam is electronically timed; you do not have to keep track of your own time. To accommodate those in other time-zones and complex working situations, you may choose any 90 minute window between 5pm PDT Oct 22nd, 2020 and 5pm PDT Oct 23rd, 2020 to take the exam. Answer all questions. You may consult any material (e.g., books, calculators, computer programs, and online resources), but you may not consult other people inside or outside of the class. If you need clarification on a question, please make a private post on Piazza. **Only what is submitted prior to the deadline will be graded.**

Question 1. (1 pt) I understand the above instructions and will adhere to them.

Question 2. (4 pts) We are implementing value iteration for an MDP with 10 possible states and 5 possible actions. Assume every state-action pair has a non-zero probability of transitioning to every state - in other words, $T(s' \mid s, a) > 0$ for all s and a.

a) (2 pts) For each iteration of value iteration, where a single iteration corresponds to all states being updated, how many times will we need to evaluate the transition function?

b) (2 pts) What is one advantage of *Gauss-Seidel value iteration* over standard value iteration? Explain.

Solution:

a) Recall that the update to the value function in value iteration is given by

$$U_{k+1}(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} T(s'|s,a) U_k(s') \right) \tag{1}$$

We will apply the given equation for every 10 states. In that equation, we take the argmax over a, so we will need to iterate over a for each state we are updating. For each value of a in the argmax, we iterate over all possible next states s' and access the transition function once for each s'. This gives us $|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}| = 10 \times 5 \times 10 = 500$ calls to the transition function for each iteration.

b) Any of the following (as well as any other valid answers) count for full credit:

- It can converge more quickly depending on the ordering chosen
- A second value function does not have to be constructed in memory in each iteration, which can lead to computational savings.

Question 3. (6 pts) We need to decide between action $A = 0$ or $A = 1$. We have a single variable C that affects our utility when we take this single action. The probability distribution over C and the utility as a function of C and the action A are given below. We would like to find the value of information for observing the value of variable C.

C	$P(C)$
0	9/10
1	1/10

A	C	$U(A,C)$
0	0	0
0	1	−20
1	0	−5
1	1	−2

a) (1 pt) Compute the optimal expected utility given C is 0.

b) (1 pt) Compute the optimal expected utility given C is 1.

c) (2 pts) What is the expected utility of taking the optimal action if we don't know the value of C?

d) (2 pts) Using your results from parts a, b, and c, compute the value of information for observing the value of the variable C.

Solution: We start with the equation for the value of information.

$$VOI(O' \mid o) = \left(\sum_{o'} P(o' \mid o) EU^*(o, o') \right) - EU^*(o)$$

Now, in this situation we have no previous observations. Our observation corresponds to knowing the state of C. As a result, we can rewrite the equation as

$$VOI(C) = \left(\sum_{c} P(c) EU^*(c) \right) - EU^*(\emptyset)$$

The $\emptyset$ corresponds to not having any observations. We can expand the sum, giving us:

$$VOI(C) = P(c^0) EU^*(c^0) + P(c^1) EU^*(c^1) - EU^*(\emptyset)$$

Each subpart of the question is asking for different terms from this equation. Part a is asking for $EU^*(c^0)$, part b is asking for $EU^*(c^1)$, part c is asking for $EU^*(\emptyset)$ and part d is asking for the full quantity.

a) From the table we observe that when c is 0, we can either take action 0 for 0 utility or action 1 for -5 utility. The optimal action will be taking action 0 for 0 utility. This gives us that

$$EU^*(c^0) = 0$$

b) From the table we observe that when c is 1, we can either take action 0 for -20 utility or action 1 for -2 utility. The optimal action will be taking action 1 for -2 utility. This gives us that

$$EU^*(c^1) = -2$$

c) We again will consider both actions. First we consider $a = 0$. This will give us an expected reward of $P(c^0) * U(c^0, a^0) + P(c^1) * U(c^1, a^0) = 9/10 * 0 + 1/10 * -20 = -2$. For $a = 1$, we get an expected reward of $P(c^0) * U(c^0, a^1) + P(c^1) * U(c^1, a^1) = 9/10 * -5 + 1/10 * -2 = -45/10 - 2/10 = -47/10$. Because -2 is larger than $-47/10$, we will choose action 0, and as a result

$$EU^*(\emptyset) = -2$$

d) Now we return to our expanded equation for the value of information to put this all together. We have

$$VOI(C) = P(c^0) EU^*(c^0) + P(c^1) EU^*(c^1) - EU^*(\emptyset)$$

Plugging in the values we've found in a, b, and c as well as $P(c^0) = 9/10$ and $P(c^1) = 1/10$ from the table we get:

$$9/10 \cdot 0 + -2 \cdot 1/10 - (-2) = -2/10 + 2 = 18/10 = 1.8$$

Question 4. (6 pts) Consider an MDP with a state space with two elements and an action space with two elements. We will consider an online policy which uses branch and bound to choose the action from a given state. Assume we perform a search with a depth $d = 3$.

a) (2 pts) What is the *maximum* number of leaves that will be visited by the branch and bound algorithm? (A leaf in a tree is a node without any children.)

b) (2 pts) What is one benefit of using *branch and bound* over *forward search*? Explain.

c) (2 pts) What is one reason you would use an *online* planning method instead of *offline* planning method? Explain.

d) (BONUS: 2 pts) What is the *minimum* number of leaves that will be visited by the branch and bound algorithm? (A leaf in a tree is a node without any children.)

Solution:

a) In the worst case, like forward search, it will need to visit every leaf. We have a branching factor b of 2 actions * 2 states = 4. Because the depth d is 3 we have $b^d = 4^3 = 64$ leaves.

b) Potential answers:

- It may not need to explore the full tree. As a result it can be more space and time efficient.

c) Potential answers:

- If you have a large state space that would be prohibitive to iterate through. Online methods need only consider locally reachable states from your current state.

d) 1 leaf, if all of the rest can be pruned using the upper and lower bounds **and** the transitions are deterministic.

Question 5. (6 pts) You are a baby sea turtle trying to make it from your nest to the ocean. It's a challenging and dangerous journey of n steps, the last one being the ocean. You're new to this world so when you try taking a step there is a 50/50-chance you might slip and just stay where you are. You iteratively choose to move forward (a_1), stay put (a_2), or move backward (a_3) until you reach the ocean, where you are being rewarded with two sea turtle snacks. Each sea turtle snack corresponds to a unit of reward. You may model this as an infinite horizon problem with s_n being an absorbing state, meaning $T(s_n|s_n, a) = 1$ for all a; you cannot leave the ocean once you've entered it.

Your current policy is to stay put, unless you are one step away from the ocean s_{n-1}, where you would take action a_1 to move forward to reach the ocean (s_n). Let's perform one step of policy optimization:

a) (4 pts) With $\gamma = 0.5$ and $n = 10$, evaluate your current policy.
Hint: If you choose to invert a matrix, you can use the following link https://matrix.reshish.com/inverse.php or any other tool of your choice.

b) (2 pts) Compute your policy improvement: state your resulting action value function $Q(s, a)$ and new policy $\pi(s)$.

For ease of grading *please* put your $Q(s, a)$ in an $n \times 3$ matrix. The first row should correspond to state 1, the second to state 2, and so on. The first column should correspond to a_1, the second to a_2, and the third to a_3.

Solution:

a) We accept three different solutions that interpret the problem statement differently. The difference between these solutions is in the reward model and episode horizon.

Solution 1: We let the horizon be infinite and let $R(s_n, a) = 2$ for all a. That is, we reward any action that is taken while being in the ocean state, s_n. We start by setting up R and T^π matrices

$$R = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2 \end{bmatrix}, \quad T^\pi = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ & \ddots & & \ddots & \\ 0 & 0 & \dots & 0.5 & 0.5 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

We can evaluate the vectorized value function update and find,

$$U = (1-\gamma T)^{-1}R = \begin{bmatrix} 2 & 0 & \dots & 0 & 0 \\ 0 & 2 & \dots & 0 & 0 \\ & \ddots & & \ddots & \\ 0 & 0 & \dots & \frac{4}{3} & \frac{2}{3} \\ 0 & 0 & \dots & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \frac{4}{3} \\ 4 \end{bmatrix}$$

Solution 2: We let the horizon be infinite and let $R(s_{n-1}, a_1) = 1$. That is, we reward the successful transition from s_{n-1} to s_n (entering the ocean). With this, the expected reward for taking action a_1 in state s_{n-1} is 1 (with 0.5 probability we transition to s_{n-1} and get 0 reward and with 0.5 probability we transition to s_n and get 2 reward). Again, we start by setting up R and T^π matrices

$$R = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \quad T^\pi = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ & \ddots & & \ddots & \\ 0 & 0 & \dots & 0.5 & 0.5 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

We can evaluate the vectorized value function update and find,

$$U = (1-\gamma T)^{-1}R = \begin{bmatrix} 2 & 0 & \dots & 0 & 0 \\ 0 & 2 & \dots & 0 & 0 \\ & \ddots & & \ddots & \\ 0 & 0 & \dots & \frac{4}{3} & \frac{2}{3} \\ 0 & 0 & \dots & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \frac{4}{3} \\ 0 \end{bmatrix}$$

Solution 3: We also accepted solutions for a finite horizon formulation where the episode ends once any action is taken in state s_n. For this, we have $R(s_n, a) = 2$ for all a. However, because the episode ends once we take an action in s_n, we do not transition to any states after s_n. In order to solve using the same formula as the previous solutions, we can signify the end of the episode in the finite horizon problem as transitioning to an absorbing state s_{n+1} in an infinite horizon problem where $R(s_{n+1}, a) = 0$ for all a. Again, we start by setting up R and T^π matrices.

$$R = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2 \\ 0 \end{bmatrix}, \quad T^\pi = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ & \ddots & & \ddots & & \\ 0 & 0 & \dots & 0.5 & 0.5 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix}$$

We can evaluate the vectorized value function update and find,

$$U = (1-\gamma T)^{-1}R = \begin{bmatrix} 2 & 0 & \dots & 0 & 0 & 0 \\ 0 & 2 & \dots & 0 & 0 & 0 \\ & \ddots & & \ddots & & \\ 0 & 0 & \dots & \frac{4}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \dots & 0 & 1 & 1 \\ 0 & 0 & \dots & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \frac{2}{3} \\ 2 \\ 0 \end{bmatrix}$$

Here, the last element of U is the value of the absorbing state, s_{n+1} that we added. The final solution can exclude this value.

b) Our solution here depends on part a),

Solution 1:

$$Q(s,a) = R(s,a) + \gamma \sum^{s'} T(s' \mid s,a)U(s')$$
$$= \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ \gamma\frac{1}{2}\frac{4}{3} & 0 & 0 \\ \gamma\frac{1}{2}\frac{4}{3} + \gamma\frac{1}{2}4 & \gamma\frac{4}{3} & \gamma\frac{1}{2}\frac{4}{3} \\ 2+4\gamma & 2+4\gamma & 2+4\gamma \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ \frac{1}{3} & 0 & 0 \\ \frac{4}{3} & \frac{2}{3} & \frac{1}{3} \\ 4 & 4 & 4 \end{bmatrix}$$

Solution 2:

$$Q(s,a) = R(s,a) + \gamma \sum^{s'} T(s' \mid s,a)U(s')$$
$$= \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ \gamma\frac{1}{2}\frac{4}{3} & 0 & 0 \\ 1+\gamma\frac{1}{2}\frac{4}{3} & \gamma\frac{4}{3} & \gamma\frac{1}{2}\frac{4}{3} \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ \frac{1}{3} & 0 & 0 \\ \frac{4}{3} & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix}$$

Solution 3:

$$Q(s,a) = R(s,a) + \gamma \sum^{s'} T(s' \mid s,a)U(s')$$
$$= \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ \gamma\frac{1}{2}\frac{2}{3} & 0 & 0 \\ \gamma\frac{1}{2}\frac{2}{3} + \gamma\frac{1}{2}2 & \gamma\frac{2}{3} & \gamma\frac{1}{2}\frac{2}{3} \\ 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ \frac{1}{6} & 0 & 0 \\ \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ 2 & 2 & 2 \end{bmatrix}$$

In all cases, our new policy re-defines $\pi(s_{n-2}) = a_1$.

Question 6. (3 pts) What is one advantage of the cross entropy method over Hooke-Jeeves for policy optimization? Explain.

Solution:

- The cross entropy method is less likely to reach local optima given its stochastic and EM nature.
- The cross entropy method takes steps in arbitrary directions within the convex span of the samples drawn, whereas Hooke-Jeeves is constrained to searching in the N canonical directions (for N dimensions).
- The cross entropy method can represent multimodal distributions, representing multiple policies in different regions of the parameter space.

Question 7. (BONUS: 2 pts) Suppose we are using policy gradient optimization. If all trajectories used have a reward of 0, what effect will the gradient ascent step have on the policy? Explain.

Solution: Recall that in policy gradient methods we have that

$$\begin{aligned}\nabla_\theta U(\theta) &= \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) R_t \right] \\ &= \mathbb{E}_\tau \left[\nabla_\theta \log \pi_\theta(\tau) R(\tau) \right]\end{aligned}$$

Which allows us to estimate the gradient by computing

$$\nabla_\theta \log p_\theta(\tau^{(i)}) R(\tau^{(i)})$$

for each sampled trajectory $\tau^{(i)}$. If all $R(\tau^{(i)}) = 0$ then clearly

$$\nabla_\theta U(\theta) = 0$$

And the gradient ascent step becomes

$$\begin{aligned}\theta' &\leftarrow \theta + \alpha \nabla_\theta U(\theta) \\ \theta' &\leftarrow \theta + 0 \\ \theta' &\leftarrow \theta\end{aligned}$$

So the policy parameterization will not change.