CMPSCI 687 Homework 3

Due October 29, 2019, 11:55pm Eastern Time

Instructions: Collaboration is not allowed on any part of this assignment. Submissions must be typed (hand written and scanned submissions will not be accepted). You must use LaTeX. The assignment should be submitted as two documents: a .pdf with your written answers and a single .cpp file as described in the programming portion.

Part One: Written (50 Points Total)

1. (2 Points) One day while working in the engineering department of the Starship Enterprise, your friend Geordi comes to you with an idea. He points out that the warp core (engine) uses a reinforcement learning algorithm to regulate its temperature. He hypothesizes the the value function that it uses would be easier to represent and/or faster to approximate in two distinct parts: one that estimates the value of a state given that the next state is safe (within desirable thresholds), and another that estimates the value of a state given that the next state is not safe. Working with Geordi, who of course uses the notation from this class, you decide to define $\mathcal{X}$ to be the set of safe states, and $\mathcal{X}^\complement$ to be the set of unsafe states, i.e., $\mathcal{X}^\complement = \mathcal{S} \setminus \mathcal{X}$.[1] In order to continue, you and Geordi decide to establish some notation. Specifically, you want to define $v^\pi_\mathcal{Y}(s)$ to be the expected discounted return given that the agent begins in state s, follows policy π, and the next state (but not necessarily the states after the next state) happens to be in $\mathcal{Y}$. Give a mathematical definition for $v^\pi_\mathcal{Y}$ like our definition for v^π:
$$v^\pi_\mathcal{Y}(s) \coloneqq \tag{1}$$

   Answer:
$$v^\pi_\mathcal{Y}(s) \coloneqq \sum_{a\in\mathcal{A}} \pi(s,a) \sum_{s'\in\mathcal{X}} P(s,a,s')R(s,a) + \mathbf{E}[\sum_{k=1}^{\infty} \gamma^k R_{t+k} | S_t = s', \pi] \tag{2}$$

2. (5 Points) Having defined $v^\pi_\mathcal{Y}$, you decide to relate your new value functions, $v^\pi_\mathcal{X}$ and $v^\pi_{\mathcal{X}^\complement}$, to the standard value function, v^π. Derive an expression for $v^\pi(s)$ that *only* uses the following terms: $\pi, P, \mathcal{A}, \mathcal{S}, \mathcal{X}, v^\pi_\mathcal{X}$ and $v^\pi_{\mathcal{X}^\complement}$. Note: You may introduce variables when summing over sets, e.g., x in $\sum_{x\in\mathcal{X}}$. Your final answer should not include expectations or any random variables like S_t or R_t. You should begin with the definition of $v^\pi(s)$ and end with an expression that only contains the allowed terms. Show your work (show

[1] In latex, here we are using the symbols \complement and \setminus for $\complement$ and $\setminus$ respectively.

the steps, don't just jump to your final answer). You may want to derive some properties before proceeding with the derivation for $v^\pi(s)$—that is allowed.

$$v^\pi(s) = \tag{3}$$

3. (13 Points) Having related your new value functions to the standard value function, you now talk to Geordi about what to do next to design a reinforcement learning algorithm using your new value functions. Another friend named Data loads the course notes from CMPSCI 687 in Fall 2019. He finds that the next step towards developing an algorithm with this value function may be to write out a new Bellman equation for $v^\pi_{\mathcal{X}}$. Derive a Bellman-like equation for this new value function. You should begin with the definition of $v^\pi_{\mathcal{X}}$ according to your answer to the first question, and should end with a recursive expression for $v^\pi_{\mathcal{X}}$ that is written only in terms of $\mathcal{S}, \mathcal{A}, \mathcal{P}, R, d_0, \gamma, \pi, \mathcal{X}$, and $\mathcal{X}^{\complement}$. For this problem, use an alternate definition of R: $R(s, a, s') := \mathbf{E}[R_t | S_t = s, A_t = a, S_{t+1} = s']$. (Hint: Using font size "tiny", our answer spans two lines—do not expect a short answer).

$$v^\pi_{\mathcal{X}}(s) = \tag{4}$$
$$\gamma \tag{5}$$

Answer:

4. (5 Points) Consider the following definition of an optimal policy:

   > For any finite MDP with $\gamma < 1$ and precisely two actions, a_1 and a_2, for any two policies π and π', $\pi \geq \pi'$ iff $\forall s \in \mathcal{S}$, $q^\pi(s, a_1) \geq q^{\pi'}(s, a_1)$. A policy π is optimal iff $\pi \geq \pi'$ for all policies π'.

   Is this definition equivalent to the definition from Section 4.5 in the course notes? Prove that your answer is correct.
   Answer:

   The definition here is equivalent to the definition from Section 4.5. From the definition here we have

$$q^\pi(s, a_1) = \mathbf{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a_1, \pi] \tag{6}$$
$$= \mathbf{E}[R_t + \sum_{k=1}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a_1, \pi] \tag{7}$$
$$= \mathbf{E}[R_t | S_t = s, A_t = a_1] + \mathbf{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, \pi] \tag{8}$$

and

$$v^\pi(s) = \mathbf{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, \pi] \tag{9}$$

From these two definitions we can find that if $q^\pi(s, a_1) \geq q^{\pi'}(s, a_1)$, we can derive that

$$\mathbf{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, \pi] \geq \mathbf{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, \pi'] \tag{10}$$

5. (5 Points) Consider a different definition of $\geq$ for policies: $\pi \geq \pi'$ iff $\sum_{s \in \mathcal{S}} d_0(s) v^\pi(s) \geq \sum_{s \in \mathcal{S}} d_0(s) v^{\pi'}(s)$. Using this modified version of $\geq$, we can still define an optimal policy to be any policy π such that $\pi \geq \pi'$ for all π'. Prove that using this definition of an optimal policy is equivalent to using our first definition:

$$\pi^* \in \arg\max_{\pi \in \Pi} J(\pi). \tag{11}$$

Answer:
Here, $\sum_{s \in \mathcal{S}} d_0(s) v^\pi(s)$ equals to

$$\sum_{s \in \mathcal{S}} Pr(S_0 = s) \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_t | S_9 = s, \pi] \tag{12}$$

while $J(\pi) = \mathbf{E}[\sum_{t=0}^{\infty} R_t | \pi]$, we can find that if $\sum_{s \in \mathcal{S}} d_0(s) v^\pi(s) \geq \sum_{s \in \mathcal{S}} d_0(s) v^{\pi'}(s)$ is equivalent to

$$\pi^* \in \arg\max_{\pi \in \Pi} J(\pi). \tag{13}$$

6. (20 Points) In class we proved that the Bellman operator is a contraction, and used this to show that value iteration converges to a unique fixed point. In this problem you will prove that the dynamic programming policy evaluation operator is a contraction, and so the policy evaluation algorithm converges to a unique fixed-point. (From the Bellman equation, it should then be clear that this fixed point is v^π, establishing that our dynamic programming policy evaluation algorithm converges to v^π.) Let f denote the dynamic programming policy evaluation operator (this is currently equation (200) in the course notes, viewed as an operator on value function approximations):

$$fv(s) = \sum_{a \in \mathcal{A}} \pi(s, a) \sum_{s' \in \mathcal{S}} P(s, a, s') \big(R(s, a) + \gamma v(s')\big). \tag{14}$$

Notice that the definition of f relies on a specific policy π—this is the policy being evaluated by the policy evaluation algorithm. Prove that f is a contraction under the L^∞ norm (the same max norm used in our proof that the Bellman operator is a contraction).

Part Two: Programming (25 Points Total)

For this part of the assignment, you will implement value iteration (modified to terminate when the value function estimate has not changed significantly between two iterations). Your program will read an MDP from a file, run value iteration on the MDP, and output the final estimate of the optimal value function and the policies that are greedy with respect to this value function. As a soft introduction to C++, we are providing you with most of the code here: your job is to fill in the missing lines in the function `valueIteration`, marked with a comment saying "TODO". Do not change the code logic outside of the valueIteration function (you may add new functions if you like, but do not modify any of the other functions in your final submission or it may fail to run as expected in our auto-grader).

You are free to use any IDE or toolchain you would like to program in C++. If you are not familiar with C++, we have provided two different systems for opening and working with this C++ code. If you are using Windows, you should download Microsoft Visual Studio. The community version is perfectly sufficient, and is free online (in my opinion, this is the best C++ experience out there). Clicking on the .sln file in HW3/build/VisualStudio will open the project. On the left you should see main.cpp—open this file to see all of the code for this assignment. If you are using Mac or Linux, we have provided a CLion project. CLion is free for students. To open this project, select "Open" when launching CLion. Select the file HW3/build/CLion/CMakeLists.txt. When prompted, select "Open as Project". If main.cpp does not immediately open, on the left click on HW3/main.cpp.

This assignment is your chance to begin to familiarize yourself with C++. Please look over all of the provided code, and feel free to ask if you have questions about what some portion of the code is doing. Also, take this opportunity to familiarize yourself with the debugger in your IDE—developing simple programs in C++ is a breeze when you are familiar with how to use the different capabilities of your debugger.

We have provided you (within the provided code) with 687Gridworld.txt, a text file containing the MDP we have been using in class. We will evaluate your program on other MDPs that we are not providing to you. You are welcome to create your own test MDPs, but do not share these with others.

You must submit your main.cpp file. A correct implementation is worth 20 points. *Any* incorrect output (beyond numerical issues) will result in 0/20 points. In the .pdf that you submit, answer the following questions.

1. (2 Points) Did your final code compile on your machine? (Yes or no).
   Answer:
   Yes

2. (3 Points) Comment on your experience with this problem. Did your first implementation work, or did you introduce a bug at first? Was there anything we could do to smooth your introduction to C++? Did you

implement any additional test MDPs (you do not have to in order to get full credit). Did the number of iterations required by value iteration surprise you? Do you have any other comments on this problem?
Answer:
Yes, my first implementation works well. To smooth the introduction to C++, maybe you should introduce more relevant APIs, which may help us familiar with usable libraries.