CS221 Problem Workout Solutions

Week 4

1) [CA session] Riding the Bus

(a) Norbert wants to go from their house (located at 1) to the gym (located at n). At each location s, they can either (i) deterministically walk forward to the next location $s+1$ (takes 1 unit of time) or (ii) wait for the bus. The bus comes with probability ϵ, in which case, they will reach the gym in $1+\alpha(n-s)$ units of time, where α is some parameter. If the bus doesn't come, well, they stays put, and that takes 1 unit of time. Let our reward be negative time, which is equivalent to minimizing the time it takes to get to the gym.

1	2	3	4	...	n
House				$\cdots$	Gym

We have formalized the problem as an MDP for you:

- State: $s \in \{1, 2, \ldots, n\}$ is Sabina's location
- Actions$(s) = \{\text{Walk}, \text{Bus}\}$
- $\text{Reward}(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s+1 \\ -\infty & \text{otherwise} \end{cases}$
- $\text{Reward}(s, \text{Bus}, s') = \begin{cases} -1-\alpha(n-s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$
- $T(s'|s, \text{Walk}) = \begin{cases} 1 & \text{if } s' = s+1 \\ 0 & \text{otherwise} \end{cases}$
- $T(s'|s, \text{Bus}) = \begin{cases} \epsilon & \text{if } s' = n \\ 1-\epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases}$
- IsEnd$(s) = \mathbf{1}[s = n]$

BEFORE YOU MOVE FORWARD: Make sure you understand *why* the MDP is formulated the way it is!

Compute a closed form expression for the value of the "always walk" policy and the "always wait for the bus" policy (using some or all of the variables ϵ, α, n). Assume a discount rate of $\gamma = 1$.

- $V_{\text{Walk}}(s) =$ _____

- $V_{\text{Bus}}(s) =$ _____

- For what values of ϵ (as a function of α and n) is it advantageous to walk rather than take the bus?

Solution Expected value for always walking is just the cost to get to the end n from a given state s:

$$V_{\text{Walk}} = -(n - s).$$

Don't forget the negative! In MDPs we maximize reward and we are considering how long it takes to get to the gym a cost (negative reward).

Expected value for always waiting for bus is trickier. Since the transition is no longer deterministic we need to split our value into the expected value of the successive state, weighted by the probability we end up in that successive state. Hence:

$$\begin{aligned} V_{\text{Bus}}(s) =& \mathbb{P}[\text{bus comes and we go to } n](R(s, \text{Bus}, n) + \gamma V_{\text{Bus}}(n)) \\ &+ \mathbb{P}[\text{bus doesn't come and we stay at } s](R(s, \text{Bus}, s) + \gamma V_{\text{Bus}}(s)) \end{aligned}$$

We only need to consider these two states since anything else happening has probability 0. Now we can just substitute in the correct values for these expressions and get:

$$V_{\text{Bus}}(s) = \epsilon(-1 - \alpha(n - s)) + (1 - \epsilon)(-1 + V_{\text{Bus}}(s)).$$

Simplifying, we get:

$$V_{\text{Bus}}(s) = -\alpha(n - s) - \frac{1}{\epsilon}.$$

For walking to be preferable, we need $V_{\text{Walk}}(s) \geq V_{\text{Bus}}(s)$, or equivalently:

$$n - s \leq \alpha(n - s) + \frac{1}{\epsilon} \Leftrightarrow (1 - \alpha)(n - s) \leq \frac{1}{\epsilon} \Leftrightarrow \begin{cases} \epsilon \leq \frac{1}{(1-\alpha)(n-s)} & , \alpha < 1 \\ \epsilon > 0 & , \alpha \geq 1. \end{cases}$$

(b) Due to bureaucracy, Norbert's town is unable to provide them with transition probabilities or a reward function (i.e. a bus schedule). To solve this, Norbert decides to use reinforcement learning, specifically Q-learning to determine the best policy. Norbert starts going around town both by bus and by walking, recording the following data:

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

Run the Q-learning algorithm once over the given data to compute an estimate of the optimal Q-value $Q_{\text{opt}}(s, a)$. Process the episodes from left to right. Assume all Q-values are initialized to zero, and use a learning rate of $\eta = 0.5$ and a discount of $\gamma = 1$.

- $\hat{Q}(1, \text{Walk}) =$ _____
- $\hat{Q}(1, \text{Bus}) =$ _____
- $\hat{Q}(3, \text{Walk}) =$ _____
- $\hat{Q}(3, \text{Bus}) =$ _____
- $\hat{Q}(4, \text{Walk}) =$ _____
- $\hat{Q}(4, \text{Bus}) =$ _____

Solution On each (s, a, r, s'), recall the Q-learning updates:

$$\hat{Q}_{opt}(s,a) \leftarrow (1-\eta)\hat{Q}_{opt}(s,a) + \eta(r + \gamma \max_{a' \in \text{Actions}(s')} \hat{Q}_{opt}(s', a')). \tag{1}$$

Now the concrete updates:

- On $(1, \text{Bus}, -1, 1)$: $\hat{Q}(1, \text{Bus}) = 0.5(0) + 0.5(-1 + 1(\max(0, 0))) = -0.5$
- On $(1, \text{Bus}, -1, 1)$: $\hat{Q}(1, \text{Bus}) = 0.5(-0.5) + 0.5(-1 + 1(\max(0, -0.5))) = -0.75$
- On $(1, \text{Bus}, 3, 3)$: $\hat{Q}(1, \text{Bus}) = 0.5(-0.75) + 0.5(3 + 1(\max(0, 0))) = 1.125$
- On $(3, \text{Walk}, 1, 4)$: $\hat{Q}(3, \text{Walk}) = 0.5(0) + 0.5(1 + 1(\max(0, 0))) = 0.5$
- On $(4, \text{Walk}, 1, 5)$: $\hat{Q}(4, \text{Walk}) = 0.5(0) + 0.5(1 + 1(\max(0, 0))) = 0.5$

The final values:

- $\hat{Q}(1, \text{Walk}) = 0$
- $\hat{Q}(1, \text{Bus}) = 1.125$
- $\hat{Q}(3, \text{Walk}) = 0.5$
- $\hat{Q}(3, \text{Bus}) = 0$
- $\hat{Q}(4, \text{Walk}) = 0.5$
- $\hat{Q}(4, \text{Bus}) = 0$

(c) After using Q-learning, Norbert wants to try a different algorithm to compute $\hat{Q}$. They consider SARSA, Model-Free Monte Carlo, and Model-Based Monte Carlo with Value Iteration. Which of these can Norbert use to compute the *optimal* policy? Explain why.

(d) Norbert's job decided to start subsidizing their bus pass. Norbert decides that for states $s < \frac{3}{4}n$ they will take the bus and for states $s \geq \frac{3}{4}n$ they will walk. Write down a policy function $\pi(s)$ to represent this.

Norbert wants to know how long their expected commute will be. Should they use Q-Learning, SARSA, or Model-Free Monte Carlo to compute that, and why? Will the algorithm converge to the correct $\hat{Q}_\pi(s, a)$ *for all states* after going to work enough times? (recall that the starting state is $s = 1$ and assume $n > 1$).

(e) Why is there (potentially) a convergence problem in this situation? How can Norbert change their policy to π' so that $\hat{V}_{\pi'}(s)$ converges for *all* states?

(f) Now that Norbert understands on vs off policy learning, they consider their attempt at Q-learning from part (b). They'd like to generate more data, but still get to the gym in a reasonable amount of time. What kind of policy would do this while guaranteeing convergence?

Solution

(c) Both SARSA and Model-Free Monte Carlo are on-policy, meaning they output $\hat{Q}_\pi$ for some (fixed) policy π used to explore the space. Thus only Model-Based Monte Carlo with Value Iteration can be used, estimating $\hat{T}$ and $\hat{R}$ before running value iteration with our estimates.

(d) The policy is:

$$\pi(s) = \begin{cases} \text{Bus} & \text{if } s < \frac{3}{4}n \\ \text{Walk} & \text{if } s \geq \frac{3}{4}n \end{cases}$$

Q-Learning will not work in this case since it computes the optimal policy. You can choose between Model-Free Monte Carlo and SARSA, however they will both only update $\hat{Q}_\pi(s = 1, a = \text{Bus})$, since Norbert starts at $s = 1$ and $\pi(1) = \text{Bus}$. Eventually this one $\hat{Q}$ value will converge but no other states will.

(e) The convergence problem is as stated in the solution to (d). The only data generated will be (1,Bus,1,...,Bus,n) with arbitrary number of times they sit at state 1 waiting. Any policy that can visit all of the states will work, say walking with probability p and taking the bus with probability $1 - p$.

(f) It sounds like Norbert wants an ϵ-greedy policy that would balance exploration (random actions) with exploitation (doing the optimal action at a state) by a stochastic policy. As Norbert gets more confident about their $\hat{Q}$ values, they can decrease ϵ to take the optimal action more often (and get to the gym faster!).

2) [CA session] Choosing the Algorithm

For each of the following scenarios, determine whether Model-Based Monte Carlo, Model-Free Monte Carlo, SARSA, or Q-Learning should be used and **why**. Note there can be more than one good answer.

(a) You work in a chemistry lab that is conducting some *extremely* expensive experiments. Unfortunately, sometimes the actions you take cause non-deterministic outcomes (due to unobservable factors), and your reaction transitions to a different state randomly. Your team has several different strategies they'd like to test for running these experiments, but you don't have enough budget for lots of trials. Fortunately, the number of states and possible actions are relatively small, and you have detailed notes on many past experiments run. What should you do and why?

(b) Your roommate Valentin seems to win a lot of money playing poker against his friends. Based on what you know about him, you've got some ideas for strategies against him, but you're concerned you'll lose to his friends by tailoring your strategy to him. You decide to build a poker bot to test your strategies against random players online. You choose your reward to be how much you win at the end of a round. Which algorithm should you use and why?

(c) You decide to foster a cat, Monte, from the local Humane Society. Unfortunately, Monte is quite skittish and really likes the dark. As such, he won't get out from under your bed. You've assembled an arsenal of treats, toys, and trinkets to try and lure out Monte. Some things seem to pique his interest, but he won't seem to come out. You're pretty sure that presenting him with the right order of items at the right time of day might convince him to come out. Which algorithm could you use?

(d) For your CS 221 project you decide to build a bot to play Monopoly. After coding up all of the states, (pieces on the board, how much money each player has, etc.) you decide to train with Model-Free Monte Carlo. After using all of your compute credits, you realize that nearly every entry in your table that stores $\hat{Q}(s, a)$ is empty (how you made a table that big is besides the point). How can you fix this?

Solution

(a) Model-Based Monte Carlo would let us simulate many strategies (policies) using old and new data by generating $\hat{T}$ and $\hat{R}$ before using Policy Iteration.

(b) Model-Free Monte Carlo is a good choice since we only get a reward at the terminal state, meaning that SARSA updates would be slower. Since we are testing specific policies we need an on-policy algorithm.

(c) Q-Learning would work well here, since we'd like to learn an optimal policy (get Monte out from under the bed), and have no idea how he reacts to things. Alternatively, *Monte* Carlo (:

(d) Q-Learning with function approximation would help here. Two states that differ by only a small amount (say how much money you have, \$100 vs \$110) probably have similar $\hat{Q}$ values. Using a function approximation would allow for generalization to unseen state action pairs.

3) [Extra Practice] Problem 3

You're programming a self-driving car that can take you from home (position 1) to school (position n). At each time step, the car has a current position $x \in \{1, \ldots, n\}$ and a current velocity $v \in \{0, \ldots, m\}$. The car starts with $v = 0$, and at each time step, the car can either increase the velocity by 1, decrease it by 1, or keep it the same; this new velocity is used to advance x to the new position. The velocity is not allowed to exceed the speed limit m nor return to 0.

In addition, to prevent people from recklessly cruising down Serra Mall, the university has installed speed bumps at a subset of the n locations. The speed bumps are located at $B \subseteq \{1, \ldots, n\}$. The car is not allowed to enter, leave, or pass over a speed bump with velocity more than $k \in \{1, \ldots, m\}$. **Your goal is to arrive at position n with velocity 1 in the smallest number of time steps.**

Figure 1 shows an example with $n = 9$ positions and one speed bump $B = \{5\}$. If the maximum speed is $m = 3$ and $k = 1$ for a speed bump, then an example of a legal path is the following:

$$(1,0) \xrightarrow{+1} (2,1) \xrightarrow{+1} (4,2) \xrightarrow{-1} (5,1) \xrightarrow{0} (6,1) \xrightarrow{+1} (8,2) \xrightarrow{-1} (9,1)$$

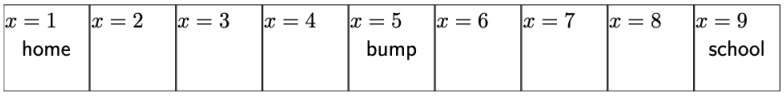

Figure 1: An example of a legal path that takes 6 time steps with $m = 3$ and $k = 1$. We show the position-velocity pairs (x, v) at each time step, and each number above an arrow is an acceleration (change in velocity).

(a) It turns out that you were so excited about the AI algorithms that you didn't really pay much attention to the brakes of the car. As a result, when you try to decrease the velocity by 1, with some failure probability α, the velocity actually stays the same. To simplify our lives, assume there are no speed bumps. Assume a reward of R if we get to school (at a velocity of 1) but -1 if we pass the school, with a cost of 1 per time step. Let us formulate the resulting problem as an MDP:

- $s_{\text{start}} = (1, 0)$
- $\text{Actions}((x, v)) = \{a \in \{+1, 0, -1\} : x + v + a \leq n \wedge v + a \leq m \wedge (v + a \leq k \vee \{x, \ldots, x + v + a\} \cap B = \emptyset)\}$. Suppose we want to apply acceleration a. First, we want to make sure we don't exceed the school ($x + v + a \leq n$) or go out of the velocity range ($v + a \leq m$). Next, we want to make sure that we're not entering, passing through, or leaving any speed bumps at a velocity greater than k. This is captured logically by ensuring a safe speed ($v + a \leq k$) or checking that there are no speed bumps between x and the new location $x + v + a$.

- $T((x', v')|(x, v), a) =$ (to be filled out by you below)
- $\text{Reward}((x, v), a, (x', v')) = R \cdot \mathbb{1}[x' = n \wedge v' = 1] - 1$
- $\text{IsEnd}((x, v)) = \mathbb{1}[x \geq n]$

(i) Fill out the definition of the transition probabilities T:

$T((x', v')|(x, v), a) =$

Solution

$$T((x', v')|(x, v), a) = \begin{cases} \alpha & \text{if } x' = x + v' \text{ and } v' = v \text{ and } a = -1 \\ 1 - \alpha & \text{if } x' = x + v' \text{ and } v' = v + a \text{ and } a = -1 \\ 1 & \text{if } x' = x + v' \text{ and } v' = v + a \text{ and } a \neq -1 \\ 0 & \text{otherwise.} \end{cases}$$

(ii) Let us explore the effect of unreliable brakes. Consider the example in Figure2.

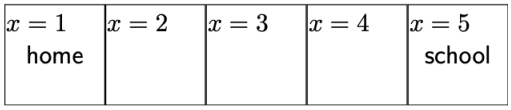

Figure 2: An small driving environment without speed bumps.

Consider two policies:

- π_1: always move with velocity 1:

$$\pi_1((1,0)) = +1 \quad \pi_1((2,1)) = 0 \quad \pi_1((3,1)) = 0 \quad \pi_1((4,1)) = 0.$$

- π_2: speed up and slow down:

$$\pi_2((1,0)) = +1 \quad \pi_2((2,1)) = +1 \quad \pi_2((4,2)) = -1.$$

Compute the expected utility of π_1 as a function of α and R (with discount $\gamma = 1$).

Solution The policy π_1 deterministically obtains reward $\boxed{R-4}$. Using $Reward(x', v')$ we have $Reward(2,1) + Reward(3,1) + Reward(4,1) + Reward(5,1) = -1 - 1 - 1 - 1 + R$

Compute the expected utility of π_2 as a function of α and R (with discount $\gamma = 1$).

Solution The policy π_2 obtains reward $\boxed{(1-\alpha)R - 3}$. We only get reward R if we are able to break at the end so: $(1-\alpha)(Reward(2,1) + Reward(4,2) + Reward(5,1)) + \alpha$
$(Reward(2,1) + Reward(4,2) + Reward(6,2)) = (1-\alpha)(R-3) + \alpha(-3) = (1-\alpha)R - 3$

For what values of α and R does π_2 obtain higher expected reward than π_1? Your answer should be an expression relating α and R.

Solution Therefore, π_2 is better when $(1-\alpha)R - 3 > R - 4$, which is precisely when $\boxed{\alpha < 1/R}$.

(b) Bad news: you realize that your brakes are not only faulty, but that you don't know how often they fail (α is unknown). Circle all of the following algorithms that can be used to compute the optimal policy in this setting:

model-based value iteration model-free Monte Carlo SARSA Q-learning

Solution **Model-based value iteration** would estimate the transition probabilities, which can be used to compute the optimal policy. **Q-learning** can be used to directly estimate the value of the optimal policy. Model-free Monte Carlo and SARSA can only be used to compute the value of a fixed policy.