CS221 Problem Workout Solutions

Week 4

Key Takeaways from this Week

We formalize a **Markov Decision Process** or **MDP** for short as:

- **S**tates: States S with starting state $s_{\text{start}} \in S$.
- **T**ermination State: $\texttt{isEnd}(s)$.
- **A**ctions: $a \in A(s)$, all possible actions at state s.
- **R**ewards: $R(s, a, s')$, the reward of going from state s to s' by taking action a.
- **T**ransitions: $T(s, a, s')$, the probability of going from state s to s' by taking action a.
- **D**iscount: $0 \leq \gamma \leq 1$, the discount factor (default 1) for computing utility.

$V_\pi(s)$ is the expected utility received by following policy π from state s.
$Q_\pi(s, a)$ is the expected utility of taking action a from state s, and then following policy π.

The following are **off-policy** algorithms that output the optimal Q-value, Q_{opt}:

- **Value Iteration**: $V_{\text{opt}}^{(t)}(s) \leftarrow \max_{a \in A(s)} Q_{\text{opt}}^{(t-1)}(s, a)$.
- **Model-Based Value Iteration**: Estimate T and R using Monte Carlo, then run value iteration using estimates $\hat{T}$ and $\hat{R}$.
- **Q-Learning**: Estimate $\hat{Q}_{\text{opt}}(s, a)$ based on (i) the reward to state s' and (ii) the estimated optimal max value of s'.

The following are **on-policy** algorithms that output the Q-value, Q_π, of a specific policy:

- **Policy Iteration**: $V_\pi^{(t)}(s) \leftarrow Q_\pi^{(t-1)}(s, \pi(s))$.
- **Model-Free Monte Carlo**: Estimate $\hat{Q}_\pi(s, a)$ from the utility, u_t, along the path.
- **SARSA**: Estimate $\hat{Q}_\pi(s, a)$ based on (i) the update (s, a, r, s', a') and (ii) the estimated $\hat{Q}_\pi(s', a')$.

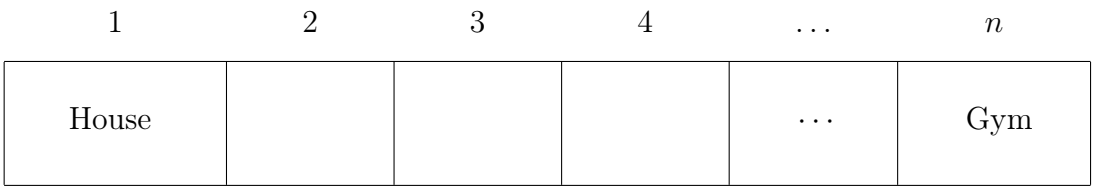
Algorithm	**Estimating**	**Based On**
Model-Based Monte Carlo	$\hat{Q}_{\text{opt}}$	$s_0, a_1, r_1, s_1, a_2, r_2, s_2, \ldots \implies \hat{T}, \hat{R}$
Q-Learning	$\hat{Q}_{\text{opt}}$	$(s, a, r, s'), \hat{V}_{\text{opt}}(s')$
Model-Free Monte Carlo	$\hat{Q}_\pi$	$u_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$
SARSA	$\hat{Q}_\pi$	$(s, a, r, s', a'), \hat{Q}_\pi(s', a')$

1) Problem 1: MDP for Riding the Bus

(a) Sabina wants to go from their house (located at location 1) to the gym (located at location n). At each location s, Sabina 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, it will take Sabina to the gym in $1+\alpha(n-s)$ units of time, where α is some parameter. If the bus doesn't come, then Sabina stays put waiting for nothing, 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.



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 closed form expressions for (i) the value of a policy where Sabina always walks at every location and (ii) the value of a policy where Sabina always waits for the bus at every location (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 Since walking is deterministic, the 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, but in this context, we want to minimize the time to takes to get to the gym. Hence, we make our rewards negative as maximizing over negative costs will give us our desired minimum time!

The 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 that 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) Unfortunately, Sabina's town is unable to provide transition probabilities or a reward function (i.e. a bus schedule), making the above MDP possibly (and likely) inaccurate. To get around this, Sabina decides to use reinforcement learning, specifically Q-learning to determine the best policy. Sabina 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:

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

2) Problem 2: Choosing an Algorithm

For each of the following questions, choose one or more of the algorithm(s) below to solve the given problem.

- Model-Based Monte Carlo.
- Q-Learning
- Model-Free Monte Carlo.
- SARSA

(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 chemical reaction transitions to a different state randomly. Your team would like to figure out the best course of action that'll most likely finish the experiments without fail, 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 data from many past experiments.

Solution 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) Your roommate seems to win a lot of money playing poker against their friends. Based on what you know about them, you've got some ideas on what strategies you can use to possibly beat them. But, you're concerned that if you tailor your strategy to your roommate's specific playstyle, then you'll lose to their friends in the crossfire. 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.

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

(c) You decide to foster a cat from the local Humane Society. Unfortunately, the cat 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 him out. 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.

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

3) [optional] Problem 3: Another MDP

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:

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

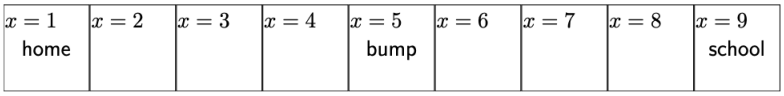


Figure 1: An example of a legal path that takes 6 time steps with $m = 3$ and $k = 1$. We show the position-velocity pairs (x, v) at each time step, and each number above an arrow is an acceleration (change in velocity).

(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)
- $\text{Reward}((x, v), a, (x', v')) = R \cdot \mathbf{1}[x' = n \wedge v' = 1] - 1$
- $\text{IsEnd}((x, v)) = \mathbf{1}[x \geq n]$

(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 Figure 2 below.

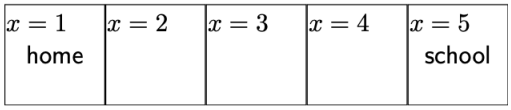


Figure 2: A small driving environment without speed bumps.

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.