1 RL: What's Changed Since MDPs?

1. Recall the Bellman Equation we used in MDPs to determine the value of a given state:

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

What information do we no longer have direct access to in the transition to RL?

In the switch from MDP to RL, we don't have access to some information on the environment model, namely the entire transition function: $T(s, a, s')$ and the reward function: $R(s, a, s')$. Instead, in RL, we receive episodes of information, or one sequence of states, actions, and rewards (one point in our T and R functions). RL still has an MDP as its backbone, but we don't have access to the complete transition and reward functions.

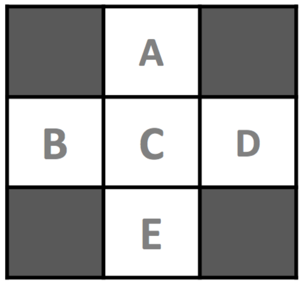
2. What is the difference between online and offline learning? Which type of learning does MDP use? How about RL?

Online learning agents learn values and policies by actually taking actions in the in environment, offline learners learn solely by analyzing the dynamics (transition and reward functions) of the environment model, without actually needing to take actions.

MDPs, having access to the dynamics and environment model, use offline learning methods like Q-value iteration or policy iteration which never need to take actions in the environment to learn an optimal value/policy. RL agents, on the other hand, must take actions in the environment in order to gain information on these dynamics to which MDPs have a priori access.

2 Temporal Difference Learning and Q-Learning

Consider the Gridworld example that we looked at in lecture. We would like to use TD learning to find the values of these states.



Suppose we use an ϵ-greedy policy and observe the following $(s, a, s', R(s, a, s'))$* transitions and rewards:

$(B$, East, C, 2), (C, South, E, 4), (C, East, A, 6), (B, East, C, 2)

**Note that the $R(s, a, s')$ in this notation refers to observed reward, not a reward value computed from a reward function (because we don't have access to the reward function).*

The initial value of each state is 0. Let $\gamma = 1$ and $\alpha = 0.5$.

1. What are the learned values for each state from TD learning after all four observations?

   For (B, East, C, 2), we update $V^\pi(B)$:
   $V^\pi(B) \leftarrow V^\pi(B) + \alpha(R(s, a, s') + \gamma V^\pi(C) - V^\pi(B)) = 0 + 0.5(2 + 1 * 0 - 0) = 1.$
   Following the same computation, we get final values: $V(B) = 3.5; V(C) = 4; V(s) = 0 \;\forall s \in \{A, D, E\}$
   Here are our intermediate computations - the values of each state after each transition are shown below:

Transitions	A	B	C	D	E
(initial)	0	0	0	0	0
$(B, East, C, 2)$	0	1	0	0	0
$(C, South, E, 4)$	0	1	2	0	0
$(C, East, A, 6)$	0	1	4	0	0
$(B, East, C, 2)$	0	3.5	4	0	0

2. In class, we presented the following two formulations for TD-learning:

$$V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)sample$$

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha(sample - V^\pi(s))$$

   Mathematically, these two equations are equivalent. However, they represent two conceptually different ways of understanding TD value updates. How could we intuitively explain each of these equations?

   The first equation takes a weighted average between our current values and our new sample. We can think of this as computing an expected value.

   The second equation updates our current values towards the new sample value, scaled by a factor of our learning rate, α. This is where the "temporal difference" term is motivated (for those of you familiar, this is gradient descent, where $(sample - V^\pi(s))$ is the gradient.).

3. What are the learned Q-values from Q-learning after all four observations? Use the same $\alpha = 0.5, \gamma = 1$ as before.

$Q(C, South) = 2; Q(C, East) = 3; Q(B, East) = 3;$ $Q(s, a) = 0$ for all other Q-states (s, a).

We use the following Q-value update rule to find what the new value should be (note what is inside of the brackets may also be referred to as *sample* in the slides):

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha)[R(s, a, s') + \gamma \max_{a'} Q(s'a')]$$

Here are our intermediate computations - the values of each Q-state after each transition are shown below (Q-states for which values did not change are omitted):

Transitions	$(B, East)$	$(C, South)$	$(C, East)$
(initial)	0	0	0
$(B, East, C, 2)$	1	0	0
$(C, South, E, 4)$	1	2	0
$(C, East, A, 6)$	1	2	3
$(B, East, C, 2)$	3	2	3

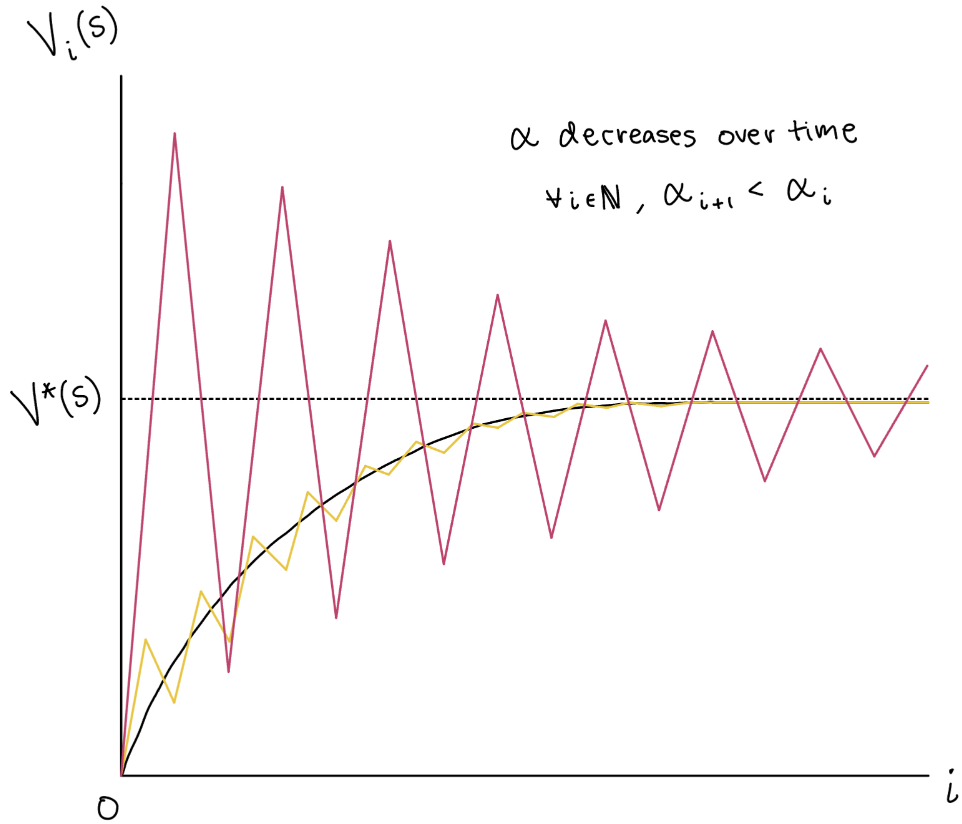
3 RL: Conceptual Questions

Recall that in Q-learning, we continually update the values of each Q-state by learning through a series of episodes, ultimately converging upon the optimal policy.

1. What's the main shortcoming of TD learning that Q-learning resolves?

   TD value learning provides a value for each state for a given policy π. It is impossible to get the optimal policy directly from the learned values because the state values are learned for the given policy π. And if we want to follow policy iteration to extract an improved policy from these values, we would need to use the R and T functions (which we don't have). With Q-learning, we can get values of Q-states (i.e., (state, action) pairs) of the optimal policy, from which we can extract an optimal policy simply by taking the action corresponding to the maximum Q-value from each state.

2. We are given two runs of TD-learning using the same sequence of samples but different α values depicted in the plot below. Assume the dashed horizontal line represents the optimal value for a specific state s and the black curve represents the smoothest transition to the optimal value given this sequence of samples. In both runs α decreases over time (or iterations), but one run has α values larger than the other run at any point in time. Which run (red or yellow) corresponds to the smaller values of α? How do the relative sizes of α affect the rate of convergence to the optimal value?



   The yellow run has smaller α values over time as the changes in $V^*(s)$ are smaller, showcasing the smaller weighting to new samples. The larger the α (or the longer α stays large compared to 0), generally the longer it takes for the run to reach convergence.

3. We are given a pre-existing table of current estimate of Q-values (and its corresponding policy), and asked to perform ϵ-greedy Q-learning. Individually, what effect does setting each of the following constants to 0 have on this process?
Remember that in ϵ-greedy Q-learning, we follow the following formulation for choosing our action:

$$\text{action at time } t = \begin{cases} \arg\max_{Q(s,a)} & \text{with probability } 1-\epsilon \\ \text{any action } a & \text{with probability } \epsilon \end{cases}$$

TD value learning provides a value for each state for a given policy π. It is impossible to get the optimal policy directly from the learned values because the state values are learned for the given policy π. And if we want to follow policy iteration to extract an improved policy from these values, we would need to use the R and T functions (which we don't have). With Q-learning, we can get values of Q-states (i.e., (state, action) pairs) of the optimal policy, from which we can extract an optimal policy simply by taking the action corresponding to the maximum Q-value from each state.

(a) α

$Q(s,a) = Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$ becomes $Q(s,a)$.
We put 0 weight on newly observed samples, never updating the Q-values we already have.
Additional remarks about the value of α: α is the learning rate or step size determining to what extent newly acquired information overrides old information. When the environment is stochastic, the algorithm converges under some technical conditions on the learning rate that require it to decrease to zero. In practice, sometimes a constant learning rate is used, such as $\alpha_t = 0.1$ for all t. If you want to learn more about learning rate in Q-learning, you can search for research papers, e.g., Even-Dar and Mansour, JMLR 2005 (http://www.jmlr.org/papers/volume5/evendar03a/evendar03a.pdf).

(b) γ

$Q(s,a) = Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$ becomes $(1-\alpha)Q(s,a) + \alpha r$.
Our valuation of reward becomes short-sighted, as we weight Q-values of successor states with 0. Continue the Q-learning process with $\gamma = 0$ and gradually decreasing α will eventually lead to Q-values of $Q(s,a) = \sum_{s'} T(s,a,s')R(s,a,s')$ because we only care about immediate reward.

(c) ϵ

By definition of an ϵ-greedy policy, we randomly select actions with probability 0 and select our policy's recommended action with probability 1; we exclusively exploit the policy we already have.

4. Consider a variant of the ϵ-greedy Q-learning algorithm that is changed such that instead of using the policy extracted from our current Q-values, we use a fixed policy instead. We still perform exploration with probability ϵ. If this fixed policy happens to be optimal, how does the performance of this algorithm compare to normal ϵ-greedy Q-learning?

Both algorithms will result in finding the optimal Q-values eventually. However, normal ϵ-greedy Q-learning makes more mistakes along the way, racking up more *regret* (the difference between actual yielded rewards and the optimal expected rewards).
In practice, normal ϵ-greedy Q-learning with a small ϵ may lead to a policy that is "pretty good" but not necessarily optimal, thus making it very unlikely for it to change unless given an extremely high number of iterations to allow for random chance to find a better policy. This result is known as a local optimum. ϵ-greedy Q-learning is in spirit similar to the simulated annealing algorithm in local search.

5. Recall the count exploration function used in the modified Q-update:

$$f(u,n) = u + \frac{k}{n+1}$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \text{max}_{a'} f(Q(s',a'), N(s',a')) - Q(s,a)]$$

Remember that k is a hyperparameter that the designer chooses, and $N(s',a')$ is the number of times we've visited the (s',a') pair. What is the effect of increasing or decreasing k?

Count exploration Q-learning incentives agents to explore states it's seen less or hasn't seen by "weighting" Q-values of state-action pairs we don't visit often with $\frac{k}{n+1}$. This is because if we haven't seen a state-action pair often, our n will be small, so the amount we're adding to our updated Q-value will be greater. Thus if we increase k, we give more weight to lesser-seen states in our final policy, and if we decrease k, we give less weight to lesser-seen states.

6. Contrast the following pairs of reinforcement learning terms:

   (i) Off-policy vs. on-policy learning

   An off-policy learning algorithm learns the value of the optimal policy independently of the policy based on which the agent chooses actions. Q-learning is an off-policy learning algorithm. An on-policy learning algorithm learns the value of the policy being carried out by the agent.

   (ii) Model-based vs. model-free

   In model-based learning, we estimate the transition and reward functions by taking some actions, then solve the MDP using them. In model-free learning, we don't attempt to model the MDP, and instead just try to learn the values directly.

   (iii) Passive vs. active

   Passive learning involves using a fixed policy as we try to learn the values of our states, while active learning involves improving the policy as we learn.

4 Approximate Q-Learning

You decide to go to Kennywood this weekend. Our RL problem is based on choosing a ride from a set of many rides.

You start off feeling well, getting positive rewards from rides, some larger than others. However, there is some chance of each ride making you sick. If you continue going on rides while sick there is some chance of becoming well again, but you don't enjoy the rides as much, receiving lower rewards (possibly negative).

You have never been to an amusement park before, so you don't know how much reward you will get from each ride (while well or sick). You also don't know how likely you are to get sick on each ride, or how likely you are to become well again. What you do know about the rides is:

Actions / Rides	**Type**	**Wait**	**Speed**
Steel Curtain	Rollercoaster	Long	Fast
Lil' Phantom	Rollercoaster	Short	Slow
Cranky's Tower	Drop Tower	Short	Fast
Pirate	Pendulum	Short	Slow
Leave the Park	Leave	Short	Slow

We will formulate this as an RL problem with two states, **well** and **sick**. Each ride corresponds to an action. The 'Leave the Park' action ends the current run through the problem. Taking a ride will lead back to the same state with some probability or take you to the other state. We will use a feature-based approximation to the Q-values, defined by the following four features and associated weights:

Features	**Initial Weights**
$f_0 = f_{\text{sick}}(s,a) = \begin{cases} 0, s = \text{Well} \\ -5, s = \text{Sick} \end{cases}$	$w_0 = 1$
$f_1 = f_{\text{type}}(s,a) = \begin{cases} 1, \text{ if } action \text{ type is Rollercoaster} \\ 0, \text{otherwise} \end{cases}$	$w_1 = 2$
$f_2 = f_{\text{wait}}(s,a) = \begin{cases} 1, \text{ if } action \text{ wait is Short} \\ 0, \text{otherwise} \end{cases}$	$w_2 = 1$
$f_3 = f_{\text{speed}}(s,a) = \begin{cases} 1, \text{ if } action \text{ speed is Fast} \\ 0, \text{otherwise} \end{cases}$	$w_3 = 0.5$

1. Calculate Q-values for each action, given the state is $'Sick'$.

   (a) Q('Sick', 'Steel Curtain') = 1(-5) + 2(1) + 1(0) + 0.5(1) = -2.5

   (b) Q('Sick', 'Lil' Phantom') = 1(-5) + 2(1) + 1(1) + 0.5(0) = -2

   (c) Q('Sick', 'Cranky's Tower') = 1(-5) + 2(0) + 1(1) + 0.5(1) = -3.5

   (d) Q('Sick', 'Pirate') = 1(-5) + 2(0) + 1(1) + 0.5(0) = -4

   (e) Q('Sick', 'Leave the Park') = 1(-5) + 2(0) + 1(1) + 0.5(0) = -4

2. Apply a Q-learning update based on the sample ('Well', 'Steel Curtain', 'Sick', -10.5), using a learning rate of $\alpha = 0.5$ and discount of $\gamma = 0.5$. What are the new weights?

   Difference = -10.5 + 0.5*max(-2.5, -2, -3.5, -4, -4) - 2.5 = -14
   $w_0 = 1 + (0.5 * -14) * 0 = 1$
   $w_1 = 2 + (0.5 * -14) * 1 = -5$

$w_2 = 1 + (0.5 * -14) * 0 = 1$
$w_3 = 0.5 + (0.5 * -14) * 1 = -6.5$

3. Now we will consider the exploration exploitation tradeoff in this amusement park. Assume we have the initial weights from the table above. What action will an ϵ-greedy approach choose from the well state? If multiple actions could be chosen, give each action and its probability.

   With probability $(1 - \frac{4\epsilon}{5})$ we will choose Lil' Phantom. Each other action will be chosen with probability $\frac{\epsilon}{5}$.

4. When running Q-learning another approach to dealing with this tradeoff is using an exploration function:

$$f(u, n) = u + \frac{k}{n}$$

   (a) How is this function used in the Q-learning equations?

   The update replaces the max over Q values with a max over the exploration function (with Q and N passed in as arguments).

   (b) What does u represent in the exploration function?

   The utility, given by Q

   (c) What does n represent in the exploration function?

   The number of times this state has been visited.

   (d) What does k represent in the exploration function?

   A constant. Adjusting this constant allows control over how "optimistic" we are about states we haven't visited much.