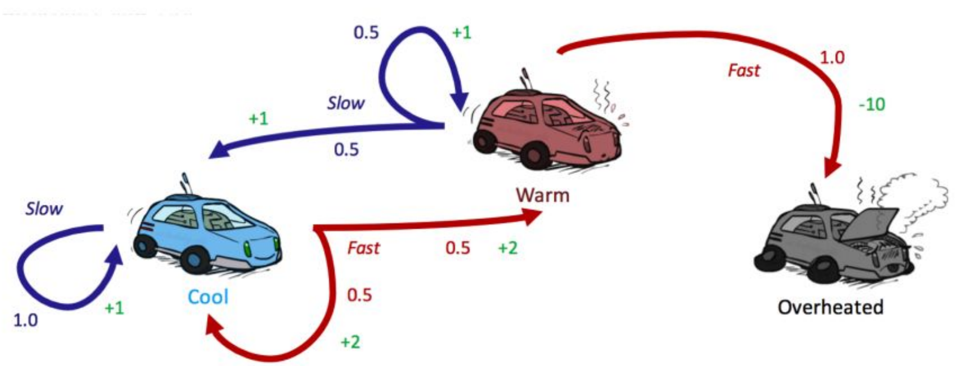
CS 188 Midterm Review 2

MDP, RL

What is an MDP?

- Set of states **S**
- Set of actions **A**
- Start state
- Terminal state(s)
- Discount factor **γ** [gamma]
  - Rewards decay as time passes, so we prefer sooner rewards
- Transition function **T(s, a, s’)**
  - Probability of ending up in state s’ by starting in s and taking action a
- Reward function **R(s, a, s’)**



Example:

- S = {cool, warm, overheated}
- A = {slow, fast}
- T(warm, slow, cool) = 0.5
- R(warm, slow, cool) = 1

Bellman Equations

- **Q*(s,a): the optimal value of (s, a)** [state, action pair]
  - The expected value of the utility an agent receives after starting in s, taking a, and acting optimally

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

- **V*(s) [aka U*(s)]: the optimal value of state s**
  - The expected value of the utility that an agent starting in s and acting optimally will receive

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

$$U^*(s) = \max_a Q^*(s,a)$$

Note: Generally, * means "optimal"

Value Iteration

Value Iteration

- A dynamic programming algorithm we use to compute values until convergence ($\forall s, U_{k+1}(s) = U_k(s)$)
- Algorithm
  - $\forall s \in S$, initialize $U_0(s) = 0$ [initial value estimate is 0]
  - Repeat rule until convergence (U-values stop changing)

$$\forall s \in S,\ U_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma U_k(s')]$$

- Convergence is when
  - $\forall s \in S, U_k(s) = U_{k+1}(s) = U^*(s)$

Remember: U(s) represents value of state, Q(s,a) represents value of state, action

Policy Iteration

Policy Iteration

- Issues with value iteration
  - $O(|S|^2|A|)$ runtime
  - Overcomputes, policy tends to converge faster than values
- Policy iteration: preserve the optimality from value iteration but with better performance by iterating until only the *policy* converges instead of the U-values

Policy Iteration

- Algorithm
  - Define initial policy $\boldsymbol{\pi_0}$ (can be arbitrary)
  - Repeat until convergence:
    - **Policy evaluation:** compute expected utility of starting in state s when following policy π, for all states s (basically, run modified value iteration until convergence)

$$U^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma U^{\pi}(s')]$$

    - **Policy improvement:** generate a better policy by picking best action at each state, given current values

$$\pi_{i+1}(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma U^{\pi_i}(s')]$$

  - Convergence when $\boldsymbol{\pi_{i+1} = \pi_i}$ (policy stops changing)

Summary

- **Value Iteration**
  - ∀s ∈ S, initialize $U_0(s) = 0$ [initial value estimate is 0]
  - Repeat rule until convergence (U-values stop changing) intuitively, pick best action based on current values

$$\forall s \in S,\ U_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma U_k(s')]$$

  - Convergence is when ∀s ∈ S, $U_k(s) = U_{k+1}(s) = U^*(s)$

- **Policy Iteration**
  - Define initial policy π_0 (can be arbitrary)
  - Repeat until convergence:
    - **Policy evaluation:** compute expected utility of starting in state s when following policy π, for all states s. (value-iterate on this policy until convergence)

$$U^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma U^\pi(s')]$$

    - **Policy improvement:** generate a better policy

$$\pi_{i+1}(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma U^{\pi_i}(s')]$$

  - Convergence when $\pi_{i+1} = \pi_i$ (policy stops changing)

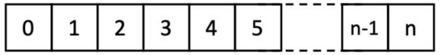
- **Q*(s,a): the optimal value of (s, a)** [state, action pair]

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

- **U*(s): the optimal value of state s** $U^*(s) = \max_a Q^*(s,a)$

Q5. MDPs: Flying Pacman

Pacman is in a 1-dimensional grid with squares labeled 0 through n, inclusive, as shown below:



Pacman's goal is to reach square n as cheaply as possible. From state n, there are no more actions or rewards available.

At any given state, if Pacman is not in n, Pacman has two actions to choose from:

- **Run**: Pacman deterministically advances to the next state (i.e. from state i to state $i + 1$). This action costs Pacman \$1.
- **Fly**: With probability p, Pacman directly reaches state n. With probability $1 - p$, Pacman is stuck in the same state. This action costs Pacman \$2.

(a) Fill in the blank boxes below to define the MDP. i represents an arbitrary state in the range $\{0, \dots, n - 1\}$.

s	a	s'	$T(s, a, s')$	$R(s, a, s')$
i	Run	$i + 1$		
i	Fly	i		
i	Fly			

For the next three subparts, assume that $\gamma = 1$.

Let π_R denote the policy of always selecting Run, and π_F denote the policy of always selecting Fly.

Compute the values of these two policies. Your answer should be an expression, possibly in terms of n, p, and/or i.

(b) What is $V^{\pi_R}(i)$?

(a) Fill in the blank boxes below to define the MDP. i represents an arbitrary state in the range $\{0, \dots, n-1\}$.

s	a	s'	$T(s, a, s')$	$R(s, a, s')$
i	Run	$i+1$	1.0	-1
i	Fly	i	$1-p$	-2
i	Fly	n	p	-2

s' is the resulting successor state after taking an action from a certain state. From state i, taking Fly either keeps you stuck in state i (row 2), or takes you directly to state n (the blank in row 3).

$T(s, a, s')$ represents the probability of starting in s, taking action a, and landing in s'.

Row 1: The action Run is deterministic, so the probability of being in i, taking action Run, and landing in $i+1$ is 100% or 1.0.

Rows 2-3: Being in i and taking action Fly lands you in state i with probability $1-p$, and lands you in state n with probability p.

Note that $R(s, a, s')$ is negative because we're trying to minimize costs. Representing costs as negative reward forces an MDP (which maximizes rewards) to minimize those costs.

As a concrete example: this MDP prefers reward -1 over reward -2 because $-1 > -2$. This is consistent with the problem statement, where Pacman would prefer to pay \$1 over paying \$2.

If we left the rewards positive, this MDP would prefer reward $+2$ over reward $+1$. This would be inconsistent with Pacman's preference of spending as little money as possible.

(b) What is $V^{\pi_R}(i)$?

$\boxed{i-n}$

This expression represents the expected reward of starting in state i and taking action Run indefinitely.

If we start in i and Run indefinitely, we will deterministically move to $i+1, i+2, \dots$ until we arrive in state n. This takes $n-i$ Run actions.

Each Run action incurs a reward of -1, and there's no discount ($\gamma = 1$), so the total reward of following this policy is $(-1) \cdot (n-i) = i-n$.

(c) What is $V^{\pi_F}(i)$?

Hint: Recall that the mean of a geometric distribution with success probability p is $\sum_{k=1}^{\infty} k(1-p)^{k-1}p = 1/p$.

(d) Given the results of the two previous subparts, we can now find the optimal policy for the MDP.

Which of the following are true? Select all that apply. (Hint: consider what value of i makes $V^{\pi_R}(i)$ and $V^{\pi_F}(i)$ equal.)

Note: $\lceil x \rceil$ is the smallest integer greater than or equal to x.

- ☐ If $p < 2/n$, Fly is optimal for all states.
- ☐ If $p < 2/n$, Run is optimal for all states.
- ☐ If $p \geq 2/n$, Fly is optimal for all $i \geq \lceil n - 2/p \rceil$ and Run is optimal for all $i < \lceil n - 2/p \rceil$.
- ☐ If $p \geq 2/n$, Run is optimal for all $i \geq \lceil n - 2/p \rceil$ and Fly is optimal for all $i < \lceil n - 2/p \rceil$.
- ○ None of the above.

(c) What is $V^{\pi_F}(i)$?

Hint: Recall that the mean of a geometric distribution with success probability p is $\sum_{k=1}^{\infty} k(1-p)^{k-1}p = 1/p$.

$$\boxed{-2/p}$$

We're asked to calculate the expected total reward of starting in state i and taking action Fly indefinitely.

At each time step, with probability p, we land in state n, and with probability $1-p$, we stay in the same place. On average, using the formula given, it will take $1/p$ Fly actions before one succeeds and we reach state n. Each of those actions costs -2, and there's no discount, so the total reward of following this policy is $(-2)\cdot(1/p) = -2/p$.

Formally, to see how the formula results in $1/p$ Fly actions on average, we can write out all the possible outcomes of trying Fly indefinitely:

Case I: Our first Fly action succeeds (takes us to n). This happens with probability p, and results in a total of 1 action taken.

Case II: Our first Fly action fails (we're stuck), but our second Fly action succeeds. This happens with probability $(1-p)p$, where the $1-p$ comes from failing once and the p comes from succeeding. This results in a total of 2 actions taken.

Case III: We fail twice, and succeed on our third try. This happens with probability $(1-p)^2p$, and results in 3 actions taken.

Case IV: We fail three times, and succeed on our fourth try. This happens with probability $(1-p)^3p$, and results in 4 actions taken.

There are infinitely many cases, corresponding to failing more and more times before we succeed.

In total, the expected number of tries before we succeed comes from multiplying each possible number of actions with the probability of requiring that number of actions:

$[1\cdot p] + [2\cdot(1-p)p] + [3\cdot(1-p)^2p] + [4\cdot(1-p)^3p] + \ldots$

Writing this as an infinite sum actually yields the exact formula provided to us:

$\sum_{k=1}^{\infty} k(1-p)^{k-1}p = 1/p$

This tells us that on average, it takes $1/p$ Fly actions before one succeeds. This should match your intuition: for example, if Fly succeeds 10% of the time, we expect to try 10 times on average before succeeding once.

Notice that the total reward is independent of i (there's no i in the final expression). This should match your intuition that no matter which state you're in, the value of taking action Fly indefinitely is the same, because the Fly action behaves exactly the same regardless of which state you're in (either you're stuck in the same place, or you directly reach n).

(d) Given the results of the two previous subparts, we can now find the optimal policy for the MDP.

Which of the following are true? Select all that apply. (Hint: consider what value of i makes $V^{\pi_R}(i)$ and $V^{\pi_F}(i)$ equal.)

Note: $\lceil x \rceil$ is the smallest integer greater than or equal to x.

- ☐ If $p < 2/n$, Fly is optimal for all states.
- 🟥 If $p < 2/n$, Run is optimal for all states.
- ☐ If $p \geq 2/n$, Fly is optimal for all $i \geq \lceil n - 2/p \rceil$ and Run is optimal for all $i < \lceil n - 2/p \rceil$.
- 🟥 If $p \geq 2/n$, Run is optimal for all $i \geq \lceil n - 2/p \rceil$ and Fly is optimal for all $i < \lceil n - 2/p \rceil$.
- ○ None of the above.

Following the hint, we set $V^{\pi_R}(i) = V^{\pi_F}(i)$, and we get $i - n = -2/p$. Solving for i, we get $i = n - 2/p$.

i represents one of the states in the problem, so it should be an integer. We can use the ceiling function to round up i to the nearest integer: $i = \lceil n - 2/p \rceil$.

What does this value of i represent? At state i, $V^{\pi_R}(i) \approx V^{\pi_F}(i)$, so this i tells us approximately where it is equally good to always Run or always Fly.

—

At this point, we have to figure out whether it's better to Run or Fly for the other values of i. The answer choices suggest that for all $i \geq \lceil n - 2/p \rceil$, it's always better to Fly or Run. Similarly, for all $i < \lceil n - 2/p \rceil$, it's always better to Fly or Run.

Without looking at the answer choices, you can also use intuition to realize the same point. For low values of i, the value of Run is low (because you're far from the end), and for higher values of i, the value of Run gets higher (because you're closer to the end). However, no matter which i you're at, the value of Fly is the same (as we discussed in the previous subpart).

—

Mathematically, $V^{\pi_R}(i) = i - n$ as a function of i is increasing linearly; for higher values of i, $V^{\pi_R}(i)$ increases as well. You could draw this as an upward-sloping line on a graph of state i (x-axis) vs. value (y-axis).

However, $V^{\pi_F}(i) = -2/p$ as a function of i is a constant (it doesn't depend on i). On the same graph of state vs. value, you'd have a flat line.

The upwards-sloping line crosses the flat line at $i = n - 2/p$.

In summary: for higher states $i \geq \lceil n - 2/p \rceil$, we have $V^{\pi_R}(i) > V^{\pi_F}(i)$, which means that it's better to Run in these states.

For lower states $i < \lceil n - 2/p \rceil$, we have $V^{\pi_R}(i) < V^{\pi_F}(i)$, which means that it's better to Fly in these states.

This should match your intuition: if you're close to the end, Run is better. If you're very far away, Fly is better.

—

The final thing we have to check is the $p < 2/n$ condition that the answer choices suggest.

If we plug in $p = 2/n$, we get $i = \lceil n - 2/(2/n) \rceil = \lceil n - n \rceil = 0$.

If we plug in some lower value like $p = 1/n$, we get $i = \lceil n - 2/(1/n) \rceil = \lceil n - 2n \rceil = -2n$.

What does this i value represent? It represents the first state where Run becomes better than Fly. But if $p < 2/n$, we start getting negative values of i! For example, when $p = 1/n$, the first state where Run is better than Fly is $-2n$, which isn't a valid state! In other words, for all states greater than $-2n$, it's better to Run than Fly. This actually means that for all states in the MDP (0 through $n - 1$, which are all non-negative and thus greater than $-2n$), Run is better than Fly.

This should match your intuition: if the probability of succeeding with Fly is way too small, then it's better to always Run, no matter where you're at.

Regardless of your answers to the previous parts, consider the following modified transition and reward functions (which may not correspond to the original problem). As before, once Pacman reaches state n, no further actions or rewards are available.

For each modified MDP and discount factor, select whether value iteration will converge to a finite set of values.

(e) $\gamma = 1$

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i+1$	1.0	+5

○ Value iteration converges
○ Value iteration does not converge
○ Not enough information to decide

(f) $\gamma = 1$

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i-1$	1.0	+5

○ Value iteration converges
○ Value iteration does not converge
○ Not enough information to decide

(g) $\gamma < 1$

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i-1$	1.0	+5

○ Value iteration converges
○ Value iteration does not converge
○ Not enough information to decide

Regardless of your answers to the previous parts, consider the following modified transition and reward functions (which may not correspond to the original problem). As before, once Pacman reaches state n, no further actions or rewards are available.

For each modified MDP and discount factor, select whether value iteration will converge to a finite set of values.

(e) $\gamma = 1$

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i+1$	1.0	+5

[✓ Value iteration converges]
Value iteration does not converge
Not enough information to decide

In these questions, it helps to recall that the value of a state is the expected, discounted sum of rewards for starting in that state and acting optimally. Value iteration is trying to compute a value for every state in the MDP.

Value iteration converges if the value at every state is finite. If the value at some states is infinite, the value iteration will never converge.

—

In this subpart, n is an absorbing state: no matter how you act in this MDP, you eventually have to reach n in a finite number of time steps. Therefore, you only have a limited number of time steps to accumulate reward in this MDP, so the expected discounted sum of rewards in this MDP is finite.

Intuitively, Pacman is only allowed to move forward, so Pacman will inevitably reach n and run out of rewards to collect.

(f) $\gamma = 1$

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i-1$	1.0	+5

Value iteration converges
[✓ Value iteration does not converge]
Not enough information to decide

The expected discounted sum of rewards in this MDP could be infinite. Intuitively, Pacman could use Run and Fly to move between states i and $i+1$ indefinitely, collecting infinite reward.

(g) $\gamma < 1$

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i-1$	1.0	+5

[✓ Value iteration converges]
Value iteration does not converge
Not enough information to decide

The only difference between this subpart and the previous subpart is the discount factor, which is now less than 1.

Pacman can still use Run and Fly to move between states i and $i+1$ indefinitely, but future rewards are discounted. This causes the expected discounted sum of rewards to form an infinite geometric series with common ratio $\gamma < 1$, which converges.

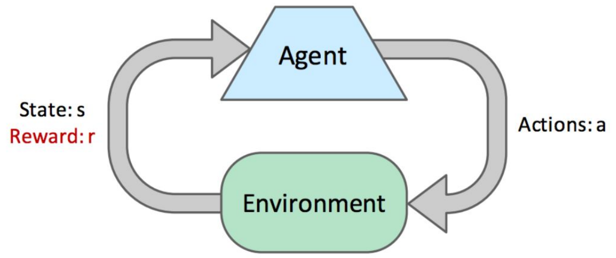
Concretely, the expected discounted sum of rewards for starting somewhere and alternating Run and Fly to move back and forth forever is: $5 + 5\gamma + 5\gamma^2 + \ldots = 5/(1-\gamma)$, which is a finite value.

Today’s Topics

- RL (Reinforcement Learning)
  - Model-based Learning
  - TD-learning (Temporal Difference)
  - Q-learning

Reinforcement Learning Overview

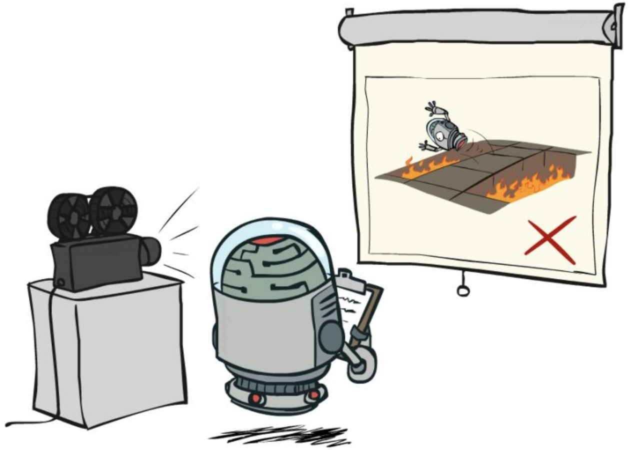
- We solved MDPs using **offline planning**
  - Knew exact transition and reward functions, could precompute optimal actions without trying anything
- Now we move to **online planning**
  - Agent has no prior knowledge of transitions and rewards
  - Try **exploration** to receive feedback
  - Estimate an optimal policy to use for **exploitation** (maximizing rewards)

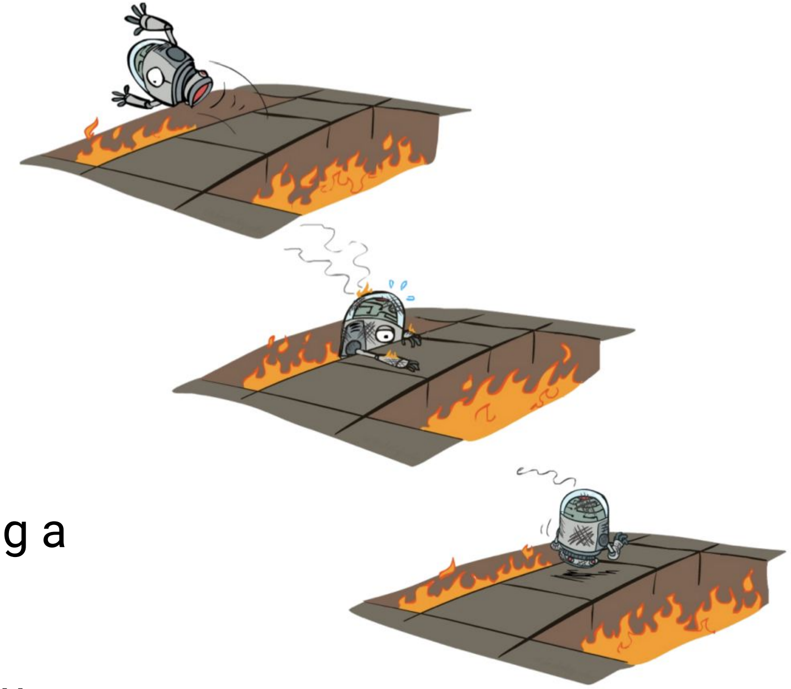


Types of Reinforcement Learning

- **Passive RL:** learn values from experience using a given policy, then use that to inform better policies
  - **Model-based:** learn the MDP model (transitions and rewards) from experiences, then solve the MDP
  - **Model-free:** forego learning the MDP model, directly learn V(s) or Q(s, a)
    - **Direct Evaluation**
    - **TD-learning (Temporal Difference)**
- **Active RL:** learn policy from our experiences directly
  - **Q-learning:** learns Q(state, action) values of the optimal policy
  - **Approximate Q-learning**

Passive vs Active RL





- **Passive RL:** learn values by watching a given policy play out
- **Active RL**: can generate and test better policies while training

Model-Based Learning, Direct Evaluation

- **Model-based learning:** observe a bunch of actions, then estimate transition probabilities T(s, a, s') and rewards R(s, a, s') by averaging.
    - T(s, a, s') is the number of times you end up in s' after being in state s and taking action a, divided by the total number of times you took a from s.
    - R(s, a, s') is the average reward you got from starting in s, taking action a, and ending up in s'.
- **Direct Evaluation:** instead of averaging to estimate transition probs and rewards, use averaging to estimate value (sum of discounted future reward) of a state $V^{\pi}(s)$ directly

TD-Learning (model-free, passive RL)

- Idea: learn values of states, given that you're following some policy
- **Algorithm**
  - Initialize value estimates for policy π $\boxed{\forall s,\ V^{\pi}(s) = 0}$
  - Each timestep:
    - Take action π(s) and receive reward R(s, π(s), s')
    - Obtain sample $\boxed{\text{sample} = R(s, \pi(s), s') + \gamma V^{\pi}(s')}$
    - Update value estimate with **exponential moving average**

$$\boxed{V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha \cdot \text{sample}}$$

    - Learning rate α, 0 ≤ α ≤ 1

Only learns values for a **given policy**. Doesn't learn optimal values/policy! Instead, results of TD-learning can be used to improve on given policy.

To make it converge, reduce α with time!

Q-Learning (active RL)

- Idea: learn Q(s, a) values - which gives you the *optimal policy* (maximize Q)
- **Algorithm**
  - Initialize Q(state, action) estimates to 0
  - Each timestep:
    - Take some action, any action*! It doesn't need to be optimal
    - Obtain sample $$\text{sample} = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$
    - Update Q-value estimate with exponential moving average

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \cdot \text{sample}$$

  - If we explore enough and decrease α appropriately, learn optimal Q-values

Remember Q(s, a) from MDP's? Represents value of being at state s and taking action a

Q6. RL: Rest and ReLaxation

Consider the grid world MDP below, with unknown transition and reward functions.

A	B	C
D	E	F
G	H	I

The agent observes the following samples in this grid world:

s	a	s'	$R(s,a,s')$
E	East	F	-1
E	East	H	-1
E	South	H	-1
E	South	H	-1
E	South	D	-1

Reminder: In grid world, each non-exit action succeeds with some probability. If an action (e.g. North) fails, the agent moves in one of the cardinally adjacent directions (e.g. East or West) with equal probability, but will not move in the opposite direction (e.g. South).

Let p denote the probability that an action succeeds.

In this question, we will consider 3 strategies for estimating the transition function in this MDP.

Strategy 1: The agent does not know the rules of grid world, and runs model-based learning to directly estimate the transition function.

(a) From the samples provided, what is $\hat{T}$(E, South, H)?

○ 0　○ 3/5　○ 2/3　○ Not enough information
○ 1/5　○ 4/5　○ 1/2
○ 2/5　○ 1/3　○ 1

(b) From the samples provided, what is $\hat{T}$(E, West, D)?

○ 0　○ 3/5　○ 2/3　○ Not enough information
○ 1/5　○ 4/5　○ 1/2
○ 2/5　○ 1/3　○ 1

Strategy 2: The agent knows the rules of grid world, and runs model-based learning to estimate p. Then, the agent uses the estimated $\hat{p}$ to estimate the transition function.

(c) From the samples provided, what is $\hat{p}$, the estimated probability of an action succeeding?

○ 0　○ 3/5　○ 2/3　○ Not enough information
○ 1/5　○ 4/5　○ 1/2
○ 2/5　○ 1/3　○ 1

(a) From the samples provided, what is $\hat{T}$(E, South, H)?

○ 0	○ 3/5	[✓ 2/3]	○ Not enough information
○ 1/5	○ 4/5	○ 1/2	
○ 2/5	○ 1/3	○ 1	

There are 3 samples that move South from E, and 2 of them result in successor state H.

(b) From the samples provided, what is $\hat{T}$(E, West, D)?

○ 0	○ 3/5	○ 2/3	[✓ Not enough information]
○ 1/5	○ 4/5	○ 1/2	
○ 2/5	○ 1/3	○ 1	

We have no samples that move West from E, so there is not enough information to empirically estimate this transition probability.

Strategy 2: The agent knows the rules of grid world, and runs model-based learning to estimate p. Then, the agent uses the estimated $\hat{p}$ to estimate the transition function.

(c) From the samples provided, what is $\hat{p}$, the estimated probability of an action succeeding?

○ 0	[✓ 3/5]	○ 2/3	○ Not enough information
○ 1/5	○ 4/5	○ 1/2	
○ 2/5	○ 1/3	○ 1	

There are 3 samples of successful actions: 1 sample of E East F, and 2 samples of E South H.

The other 2 samples are unsuccessful actions: E East H, and E South D.

(d) Based on $\hat{p}$, what is $\hat{T}$(E, West, D)?

○ 0	○ 3/5	○ 2/3	○ Not enough information
○ 1/5	○ 4/5	○ 1/2	
○ 2/5	○ 1/3	○ 1	

(e) Select all true statements about comparing Strategy 1 and Strategy 2.

☐ Strategy 1 will usually require fewer samples to estimate the transition function to the same accuracy threshold.

☐ There are fewer unknown parameters to learn in Strategy 1.

☐ Strategy 1 is more prone to overfitting on samples.

○ None of the above

(d) Based on $\hat{p}$, what is $\hat{T}$(E, West, D)?

○ 0
○ 1/5
○ 2/5
[✓ 3/5]
○ 4/5
○ 1/3
○ 2/3
○ 1/2
○ 1
○ Not enough information

Even though we have never seen a sample that moves West from E, we can still use the estimated parameters in the grid world to provide an estimate. Our estimate is that actions succeed with probability 3/5, so moving West from E will succeed (and land in D) with probability 3/5.

(e) Select all true statements about comparing Strategy 1 and Strategy 2.

☐ Strategy 1 will usually require fewer samples to estimate the transition function to the same accuracy threshold.
☐ There are fewer unknown parameters to learn in Strategy 1.
[✓ Strategy 1 is more prone to overfitting on samples.]
○ None of the above

Option 1: False. Estimating the transition function directly would require collecting samples for every state-action pair. By contrast, estimating the grid world parameters can be done even if you don't see every state-action pair.

Option 2: False. The transition function has probabilities for every (s, a, s') transition. By contrast, there is only one grid world parameter to estimate, the probability of an action succeeding.

Option 3: True. The transition function for some (s, a, s') transition is only estimated using the samples that start in s and take action a. If the samples for that particular state/action pair are biased, then the transition function for that value will also be biased.

The grid world and samples, repeated for your convenience:

A	B	C
D	E	F
G	H	I

s	a	s'	$R(s, a, s')$
E	East	F	-1
E	East	H	-1
E	South	H	-1
E	South	H	-1
E	South	D	-1

Strategy 3: The agent knows the rules of grid world, and uses an exponential moving average to estimate p. Then, the agent uses the estimated $\hat{p}$ to estimate the transition function.

(f) Consider this update equation: $\hat{p} \leftarrow (1-\alpha)\hat{p} + (\alpha)x$

Given a sample (s, a, s'), what value of x should be used in the corresponding update?

○ $R(s, a, s')$
○ 1.0 if the action succeeded, and 0.0 otherwise
○ 1.0 if the action failed, and 0.0 otherwise
○ $V(s)$
○ $V(s')$

(g) Select all true statements about comparing Strategy 2 and Strategy 3.

☐ Strategy 2 gives a more accurate estimate, because it is the maximum likelihood estimate.
☐ Strategy 3 gives a more accurate estimate, because it gives more weight to more recent samples.
☐ Strategy 3 can be run with samples streaming in one at a time.
○ None of the above

Strategy 3: The agent knows the rules of grid world, and uses an exponential moving average to estimate p. Then, the agent uses the estimated $\hat{p}$ to estimate the transition function.

(f) Consider this update equation: $\hat{p} \leftarrow (1-\alpha)\hat{p} + (\alpha)x$

Given a sample (s, a, s'), what value of x should be used in the corresponding update?

- ○ $R(s, a, s')$
- ● 1.0 if the action succeeded, and 0.0 otherwise
- ○ 1.0 if the action failed, and 0.0 otherwise
- ○ $V(s)$
- ○ $V(s')$

We're trying to estimate p, the probability of an action succeeding.

Consider a sample where the action succeeds. The estimated probability of success from that one sample is 1.0. Similarly, the estimated probability of success from a sample where the action fails is 0.0.

Note that the reward and value are not needed here, because we are not trying to estimate the action of states; instead, we are trying to estimate the probability of success.

(g) Select all true statements about comparing Strategy 2 and Strategy 3.

- ■ Strategy 2 gives a more accurate estimate, because it is the maximum likelihood estimate.
- □ Strategy 3 gives a more accurate estimate, because it gives more weight to more recent samples.
- ■ Strategy 3 can be run with samples streaming in one at a time.
- ○ None of the above

Option 1: True. The maximum likelihood estimate comes from the count estimate.

Option 2: False. In TD learning, we want to give weight to more recent samples because they use more accurate values in their calculation. However, when we're just estimating a probability from independent samples (whose values don't depend on the estimated values of other states), then there's no reason to give more weight to recent samples.

Option 3: True. To compute a count estimate, we need to be able to count up all the samples. The exponential moving average can be computed with each sample streaming in one at a time.

The rest of the question is independent from the previous subparts.

Suppose the agent runs Q-learning in this grid world, with learning rate $0 < \alpha < 1$, and discount factor $\gamma = 1$.

(h) After iterating through the samples once, how many learned Q-values will be nonzero?

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ > 4

(i) After iterating through the samples repeatedly until convergence, how many learned Q-values will be nonzero?

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ > 4

The rest of the question is independent from the previous subparts.

Suppose the agent runs Q-learning in this grid world, with learning rate $0 < \alpha < 1$, and discount factor $\gamma = 1$.

(h) After iterating through the samples once, how many learned Q-values will be nonzero?

○ 0 ○ 1 [✓ 2] ○ 3 ○ 4 ○ > 4

Q(E, East) and Q(E, South) will be nonzero.

(i) After iterating through the samples repeatedly until convergence, how many learned Q-values will be nonzero?

○ 0 ○ 1 [✓ 2] ○ 3 ○ 4 ○ > 4

Still the same two nonzero values. In order to update a Q-state, we have to see a sample with that Q-state, and we only ever see two Q-states.