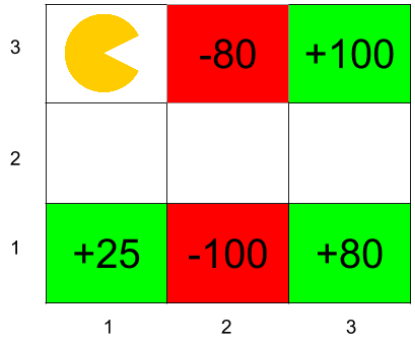
CS188: Exam Practice Session 5 Solutions

Q1. Q-Learning

Consider the grid-world given below and Pacman who is trying to learn the optimal policy. If an action results in landing into one of the shaded states, the corresponding reward is awarded during that transition. All shaded states are terminal states, i.e., the MDP terminates once arrived in a shaded state. The other states have the *North, East, South, West* actions available, which deterministically move Pacman to the corresponding neighboring state (or have Pacman stay in place if the action tries to move out of the grid). Assume the discount factor $\gamma = 0.5$ and the Q-learning rate $\alpha = 0.5$ for all calculations. Pacman starts in state (1, 3).



(a) What is the value of the optimal value function V^* at the following states:

$V^*(3,2) =$ 100 $V^*(2,2) =$ 50 $V^*(1,3) =$ 12.5

The optimal values for the states can be found by computing the expected reward for the agent acting optimally from that state onwards. Note that you get a reward when you transition *into* the shaded states and not *out* of them. So for example the optimal path starting from (2,2) is to go to the +100 square which has a discounted reward of $0 + \gamma * 100 = 50$. For (1,3), going to either of +25 or +100 has the same discounted reward of 12.5.

(b) The agent starts from the top left corner and you are given the following episodes from runs of the agent through this grid-world. Each line in an Episode is a tuple containing (s, a, s', r).

Episode 1	Episode 2	Episode 3
(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0
(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0
(2,2), S, (2,1), -100	(2,2), E, (3,2), 0	(2,2), E, (3,2), 0
	(3,2), N, (3,3), +100	(3,2), S, (3,1), +80

Using Q-Learning updates, what are the following Q-values after the above three episodes:

$Q((3,2),\text{N}) =$ 50 $Q((1,2),\text{S}) =$ 0 $Q((2,2),E) =$ 12.5

Q-values obtained by Q-learning updates - $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(R(s,a,s') + \gamma \max_{a'} Q(s',a'))$.

(c) Consider a feature based representation of the Q-value function:

$$Q_f(s,a) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(a)$$

$f_1(s)$: The x coordinate of the state $f_2(s)$: The y coordinate of the state

$$f_3(N) = 1,\ f_3(S) = 2,\ f_3(E) = 3,\ f_3(W) = 4$$

(i) Given that all w_i are initially 0, what are their values after the first episode:

$w_1 =$ -100 $w_2 =$ -100 $w_3 =$ -100

Using the approximate Q-learning weight updates: $w_i \leftarrow w_i + \alpha[(R(s,a,s') + \gamma \max_{a'} Q(s',a')) - Q(s,a)]f_i(s,a)$. The only time the reward is non zero in the first episode is when it transitions into the -100 state.

(ii) Assume the weight vector w is equal to $(1,1,1)$. What is the action prescribed by the Q-function in state $(2,2)$?

West

The action prescribed at (2,2) is $\max_a Q((2,2),a)$ where $Q(s,a)$ is computed using the feature representation. In this case, the Q-value for *West* is maximum $(2+2+4=8)$.

Q2. MDPs and RL for Solving Adversarial Games

Consider the problem of using an MDP to develop a strategy to maximize a player's reward in a game called **The Downward Spiral.** The game starts with both players having an "elevation" of zero. A game consists of N rounds, where each round consists of a turn by player 1, followed by a turn by player 2. After N rounds, the game is over. During each turn:

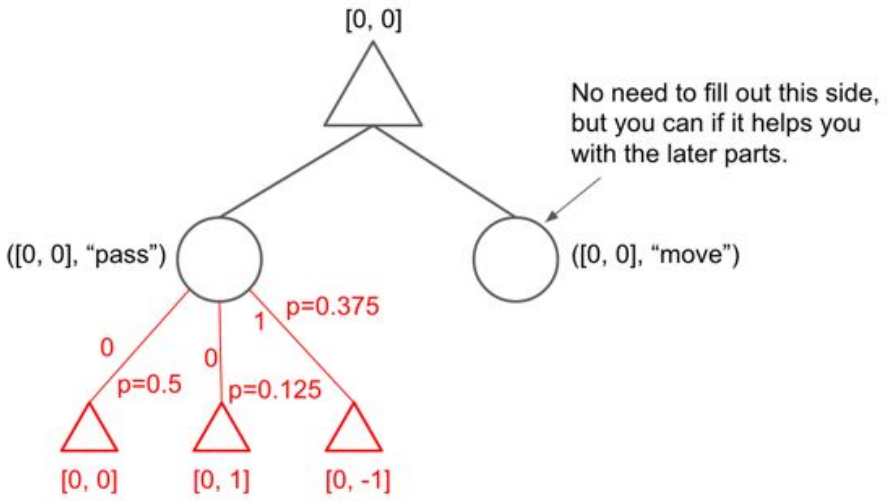
- A player may "pass" or "move".
- If a player "passes", their elevation is unchanged.
- If a player "moves", then their elevation ***increases*** by 1 with probability 0.25, and ***decreases*** by 1 with probability 0.75. Moving only changes the elevation of the active player.

Part 1: Suppose that player 1's utility is equal to the number of times that they end a round with a strictly higher elevation than player 2. One approach is to model the game's state as a pair of values representing the elevation of each player, i.e. S_0 is always equal to [0, 0]. After one round, suppose player 1 passes, and player 2 moves and goes down, then the new state is S_X = [0, -1]. To account for player 1's goal, our model has a reward function such that player 1 earns a reward of 1 point for ending the round with a higher elevation. So in the example above $R(S_0, \text{pass}, S_X) = 1$, since the round ended with player 1 having a higher elevation. Throughout the problem, let the discount factor be 1.

i. Player 1 will use an MDP model to decide what actions to take to maximize their rewards. Fill in the **MDP search tree** below for the game assuming that the chance of heads is 0.25, that the model predicts player 2 will randomly pick pass/flip with p=0.5 and the game lasts **only one round (N = 1)**. **Label transitions with their reward and probability**. Label states with the appropriate pair of values [x, y]. The starting state and Q-states are drawn for you. Only the edges and states below the left q-state ([0, 0], "pass") will be graded.

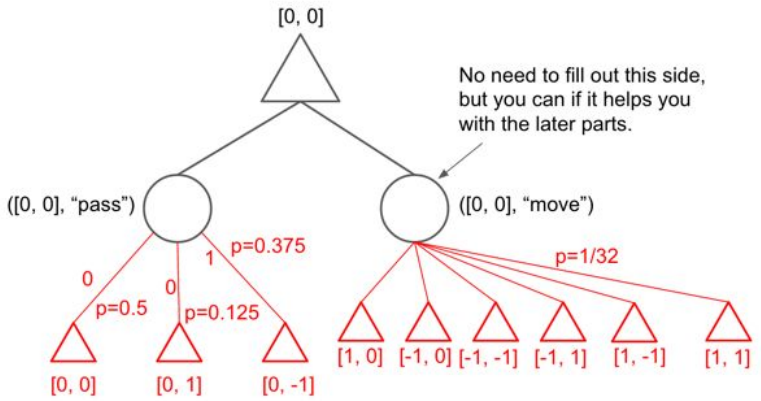
There are a few different reasonable interpretations for this problem. See the appendix to these solutions for alternate possibilities. An MDP search tree consists of decision nodes (which we denoted with upward triangles) and chance nodes (circles). The best answer is as shown below, where there are three possible outcomes:

- Outcome 1: Player 2 picks pass, no reward. Happens with chance 0.5.
- Outcome 2: Player 2 picks move, goes up. No reward. Happens with chance 0.5 * 0.25 = 0.125.
- Outcome 3: Player 2 picks move, goes down. Reward = 1. Happens with chance 0.5 * 0.75 = 0.375.



ii. In the **entire** MDP search tree **from part i** (including the side of the tree that you were not required to draw), what is the exact value of the smallest transition probability?

Solution: The least likely outcome is if both players chose to move and successfully increase their elevation; this happens with probability $1/4 * 1/2 * 1/4 = 1/32$. See below.



iii. **Let N = 2 rounds**. Let Z be the number of transitions in the entire MDP. How many total transitions are there? Give an exact answer. By transition, we mean an edge from a Q-state to a state.

Z = 90

Solution: There are 9 transitions in round 1, as partially demonstrated in part (i). In round 2, there are 9 potential states where player 1 can make an action from (each player's score can range from -1 to 1), and each has 9 possible transitions (for each combination of player 1's and player 2's possible actions), for 81 transitions in round 2.

Some students observed that the transitions in round 1 are redundant with some of the transitions in round 2, and gave an answer of 81; this is the number of unique transitions (s, a, s'), and thus also correct, although the MDP model is no longer a tree, but a graph.

If you had the wrong model for part i, there are other possible answers.

iv. Suppose we want to use **model-based reinforcement learning** to approximate player 2's strategy rather than assuming that they choose randomly. Assume that they have some sort of consistent, but not necessarily deterministic strategy. Assuming there are Z transitions, how many T(s, a, s') values do we need to learn? Again assume that **N = 2**. You can do this problem even if you didn't do part 1-ii. If the problem seems ambiguous, state your assumptions.

Number of T(s, a, s') values to learn = Z

Explanation: We learn one transition probability per transition, which is an edge from a Q-state to a state. Some of you felt that reusing (0, 0) was redundant, and thus we also accepted Z - sqrt(Z) or 72.

Special note: If you used one of the incorrect models, the correct answer is zero. We defined Transitions as between Q-states and regular states, and in alternate models (#1 and #2), these transitions all have probability 0.75 or 0.25, defined by the rules of the game, and thus there is nothing to learn.

Special note #2: Some of you tried to be clever and count the number of states and possible transitions between them. Others tried to note that some transition probabilities can be computed from others. Both of these approaches are incompatible with how model-based RL works, where we don't get to pick which observations we make. While it is true that sometimes you can compute some transition probabilities from others,this is counterproductive given the learning process of counting + normalizing.

v. Same as part iii, but how many R(s, a, s') values do we need to learn? If ambiguous, state assumptions.

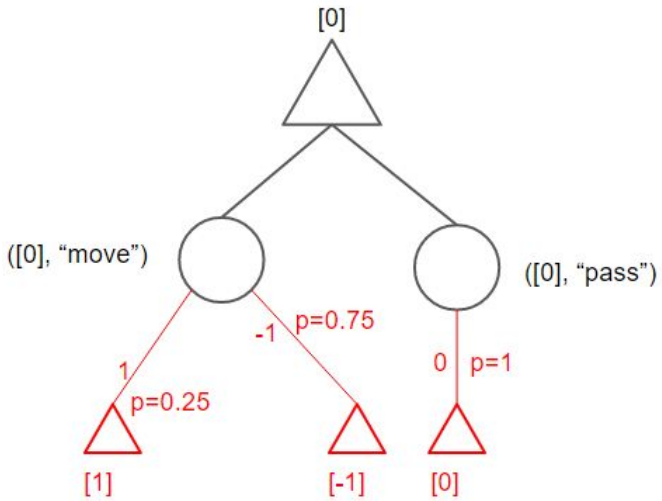
Number of R(s, a, s') values to learn = 0
Explanation: The rewards are all based on the rules of the game, and thus don't need to be learned.

Part 2) Suppose we keep the rules of The Downward Spiral the same, but change player 1's utility to **be equal to their elevation at the end of round N**, with no regard to player 2's elevation. This means adjusting our state space as well as reward function. Note: **You can do this problem without doing part 1.**

vi. Give a minimal representation of the state needed to achieve this new goal, and provide S_0. Hint: Player 2's actions no longer matter.

Solution 1: One representation is to store only the current elevation of player 1; the reward is given as +1 or -1 whenever we transition to a different state. The starting state would thus be S_0 = [0].
Solution 2: A better (and more minimal) representation is based off of the realization that the optimal action is independent of Player 1's elevation. Thus, our state only needs to be able to track whether Player 1 just moved up or down (in order to use the same reward function). Thus, the minimal state representation can be something like {"Up", "Down", "Same"}, with the reward for a transition $R(s,a,s')$ depending solely on s'. S_0 can be any of these states.

vii. Draw the entire MDP search tree for the game assuming N = 1. Make sure to label the probabilities and rewards for each transition for round 0, as well as the possible states for round 0 and round 1.



viii. If **N = 10,** give the expected utility $V^*(S_0)$ achieved under the optimal policy for this new MDP. Describe the optimal policy.

Solution: 0. We expect moving to move us lower, on average, and thus it is optimal to never move.