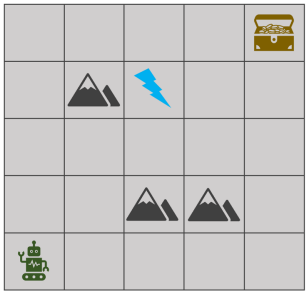
Homework Set 2

Problem 1: We want to compute a (near-optimal) policy for the same example in Problem 7 of Homework 1 in the infinite-horizon discounted setting. Consider an MDP over the grid-world illustrated below, where the goal is to take the agent to the treasure chest while avoiding the lightning.



- The state space contains the cells of the grid-world.
- The agent starts at the bottom left corner.
- The action space is $\{\text{up}, \text{down}, \text{left}, \text{right}\}$.
- The agent can only move to its adjacent cells, i.e., the cells that are above, below, to the left, or to the right of its current cell. If the agent is not at the boundary cells, each action will take the agent to the expected cell with probability 0.85 and to one of the remaining three cells, each with probability 0.05. If the agent is at the boundary, it will remain at its current cell with the sum of probabilities that would have taken it outside the grid-world. The cell with a mountain cannot be accessed, i.e., if adjacent to the mountain, the agent will remain at its current cell with the probability that would have taken it to the mountain. All actions in the cell with the lightning bolt, the one with the treasure chest, and the ones with a mountain will keep the agent in its current cell.
- The agent receives a reward of 0 in every cell for all actions except two cells. If at the cell with a lightning bolt, it will receive a reward of -1 for all actions, and if at the cell with the treasure chest, it will receive a reward of $+1$ for all actions.
- The discount factor is 0.95.

1. Implement and run a value iteration algorithm to compute an optimal policy in this MDP. Initialize the value function at zero. Pick the number of iterations in a way to ensure 0.01 accuracy in the final computed value function. Demonstrate the learned policy and report the value function at all states.
2. Implement and run a policy iteration algorithm to compute an optimal policy in this MDP. Initialize the policy randomly, by using a uniform distribution over the actions. Pick the number of iterations in a way to ensure 0.01 accuracy in the final computed value function. Demonstrate the learned policy and report the value function at all states.

[**Note:** For this problem, attach all your code in a programming language of your choice to the end of your submission.]

Problem 2: We would like to evaluate the result of the First-Visit Monte Carlo algorithm used for policy evaluation in the episodic, discounted setting. Recall that this algorithm estimates the value function under policy π in any state s, i.e.,

$$V^{\pi}(s) = \mathbb{E}_{\substack{A_t \sim \pi(.|S_t) \\ S_{t+1} \sim P(.|S_t, A_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) | S_0 = s \right]$$

by the empirical mean of a set of sample returns. Let $\mathcal{T}(s) = \{\tau_1^s, \tau_2^s, \ldots, \tau_{N^s}^s\}$ be the sample trajectories starting from s and ending in a terminal state obtained from different episodes of the algorithm. Let $\mathcal{G}(s) = \{G_1^s, G_2^s, \ldots, G_{N^s}^s\}$ be the sample returns corresponding to $\mathcal{T}(s)$, i.e.,

$$G_i^s = \sum_{t=0}^{|\tau_i^s|} \gamma^t R(S_{t,i}^s, A_{t,i}^s),$$

where $S_{t,i}^s$ and $A_{t,i}^s$ are the state and action observed at time t in trajectory τ_i^s, respectively. The Monte Carlo algorithm estimates $V^{\pi}(s)$ as follows:

$$\hat{V}^{\pi}(s) = \frac{1}{N^s} \sum_{i=1}^{N^s} G_i^s = \frac{1}{N^s} \sum_{\tau_i^s \in \mathcal{T}(s)} \sum_{t=0}^{|\tau_i^s|} \gamma^t R(S_{t,i}^s, A_{t,i}^s).$$

1. Assume that the reward function is bounded, $|R(s,a)| \leq R_{max}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Find an upper bound and lower bound on G_i^s, i.e., find α and β such that

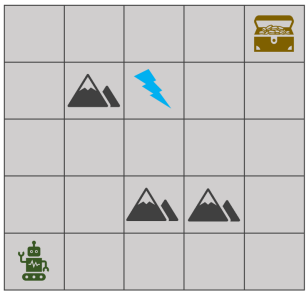
$$\alpha \leq G_i^s \leq \beta.$$

2. Let $E(s) = \sum_{i=1}^{N^s} G_i^s$ be the sum of all sample returns for state s. Recall that the expected value of each sample return is the true value function, i.e., $\mathbb{E}_{\substack{A_t \sim \pi(.|S_t) \\ S_{t+1} \sim P(.|S_t, A_t)}} [G_i^s] = V^{\pi}(s)$. Derive and express $\mathbb{E}_{\substack{A_t \sim \pi(.|S_t) \\ S_{t+1} \sim P(.|S_t, A_t)}} [E(s)]$ in terms of $V^{\pi}(s)$.

3. Apply Hoeffding's inequality to bound the probability that $E(s)$ deviates from its expected value obtained in the previous part, i.e., bound $\mathbb{P}\left(|E(s) - \mathbb{E}[E(s)]| \geq \epsilon\right)$ (the subscript for the expectation operator is omitted for simplifying the notation) for any $\epsilon > 0$. Notice that the samples G_i^s are independent random variables.

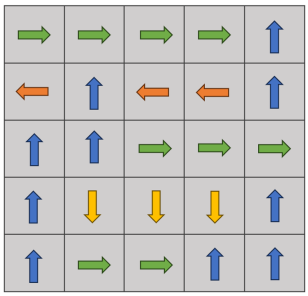
4. Now, considering $\hat{V}^{\pi}(s) = \frac{1}{N^s} E(s)$, bound the probability that $\hat{V}^{\pi}(s)$ deviates from $V^{\pi}(s)$, i.e., bound $\mathbb{P}\left(|\hat{V}^{\pi}(s) - V^{\pi}(s)| \geq \epsilon'\right)$ for any $\epsilon' > 0$.

Problem 3: We want to evaluate a policy for the same example in Problem 7 of Homework 1 in a model-free setting, i.e., when the environment model (the transition function and the reward function) is unknown to the agent. Consider an MDP over the grid-world illustrated below, where the goal is to take the agent to the treasure chest while avoiding the lightning.



- The state space contains the cells of the grid-world.
- The agent starts at the bottom left corner.
- The action space is $\{\text{up}, \text{down}, \text{left}, \text{right}\}$.
- The agent can only move to its adjacent cells, i.e., the cells that are above, below, to the left, or to the right of its current cell. If the agent is not at the boundary cells, each action will take the agent to the expected cell with probability 0.85 and to one of the remaining three cells, each with probability 0.05. If the agent is at the boundary, it will remain at its current cell with the sum of probabilities that would have taken it outside the grid-world. The cell with a mountain cannot be accessed, i.e., if adjacent to the mountain, the agent will remain at its current cell with the probability that would have taken it to the mountain. All actions in the cell with the lightning bolt, the one with the treasure chest, and the ones with a mountain will keep the agent in its current cell.
- The agent receives a reward of 0 in every cell for all actions except two cells. If at the cell with a lightning bolt, it will receive a reward of -1 for all actions, and if at the cell with the treasure chest, it will receive a reward of $+1$ for all actions.
- The discount factor is 0.95.
- Model the setting as an episodic one, where each episode ends once the agent is at one of the terminal states, i.e., the cell with the lightning bolt, the one with the treasure chest, and the ones with a mountain. Once an episode ends, the state resets to (agent respawns at) an initial state in the next episode.

Evaluate the deterministic, stationary policy shown in the figure below, where each arrow represents the prescribed action in each cell. This is the same policy evaluated in Homework 1; however, here we assume that the model is unknown to the agent. *Notice that for running the model-free algorithms, there should still be an environment model that can generate the samples. For instance, you may have an environment function in your code that takes a state, along with an action, and outputs a next state sampled according to the true (unknown to the agent) transition probabilities and a reward according to the true (unknown to the agent) reward function.*



1. Run a first-visit Monte Carlo algorithm to evaluate this policy. Describe how you ensure that each state is visited often so that its value is estimated accurately. Pick a termination criteria and justify it. Report the value function at all states after the algorithm terminates.

2. Run a one-step temporal-difference learning algorithm to evaluate this policy. Describe how you ensure that each state is visited often so that its value is estimated accurately. Pick a termination criteria and justify it. Report the value function at all states after the algorithm terminates.

3. [**Bonus**] Compute the true value function and report it at all states.
   [Hint: You can apply the analytical solution for policy evaluation but it requires some modifications since this is an episodic setting and not an infinite-horizon setting. To view the problem as an infinite-horizon one and apply the analytical solution, you may consider an extra (fictitious) state where any actions in the terminal states takes the state to this fictitious one. All actions in this new state will keep the agent in the same state and have a reward of 0.*]*

4. [**Bonus**] Plot the sequence of errors in the value function with respect to the number of episodes for both algorithms in the same figure. In particular, plot $\|V_n - V^\pi\|_2$ against $n \in \mathbb{N}_0$, where V_n is the value function after n episodes and V^π is the true value function computed in Part 3.

[**Note:** For this problem, attach all your code in a programming language of your choice to the end of your submission. You may get help from or use the existing implementations of the Monte Carlo and the temporal-difference learning algorithms.]

Problem 4: We would like to apply off-policy Monte Carlo evaluation to a policy in the infinite-horizon, discounted setting. Consider an MDP with three states $\mathcal{S} = \{1, 2, 3\}$, initial state 1, two actions $\mathcal{A} = \{g, h\}$, transition function $P(s'|s, a)$ defined as

$$\begin{aligned}
P(1|1,g) = 0.1\,, \quad P(2|1,g) = 0.8\,, \quad P(3|1,g) = 0.1\,,\\
P(1|1,h) = 0.8\,, \quad P(2|1,h) = 0.1\,, \quad P(3|1,h) = 0.1\,,\\
P(1|2,g) = 0.1\,, \quad P(2|2,g) = 0.1\,, \quad P(3|2,g) = 0.8\,,\\
P(1|2,h) = 0.1\,, \quad P(2|2,h) = 0.8\,, \quad P(3|2,h) = 0.1\,,\\
P(1|3,g) = 0.8\,, \quad P(2|3,g) = 0.1\,, \quad P(3|3,g) = 0.1\,,\\
P(1|3,h) = 0.1\,, \quad P(2|3,h) = 0.1\,, \quad P(3|3,h) = 0.8\,,
\end{aligned}$$

reward function $R(s, a)$ outputting 1 for $(3, h)$ and 0 otherwise, and discount factor 0.95.

1. Consider the target policy π^t

$$\begin{aligned}
\pi^t(g|1) = 0.9\,, \quad \pi^t(h|1) = 0.1\,,\\
\pi^t(g|2) = 0.9\,, \quad \pi^t(h|2) = 0.1\,,\\
\pi^t(g|3) = 0.1\,, \quad \pi^t(h|3) = 0.9\,,
\end{aligned}$$

   and the behavior policy π^b

$$\begin{aligned}
\pi^b(g|1) = 0.85\,, \quad \pi^b(h|1) = 0.15\,,\\
\pi^b(g|2) = 0.88\,, \quad \pi^b(h|2) = 0.12\,,\\
\pi^b(g|3) = 0.1\,, \quad \pi^b(h|3) = 0.9\,.
\end{aligned}$$

   Evaluate the infinite-horizon performance of these two policies, i.e., compute the value functions V^{π^t} and V^{π^b}, assuming access to the model. Report the values for both policies.

2. In order to limit the sample trajectories to finite ones in the Monte Carlo algorithm, compute an effective horizon such that the errors in returns are less than $\epsilon = 0.1$.

3. Use off-policy Monte Carlo evaluation for the target policy to estimate the value function in the initial state $s = 1$. In particular, generate 50 trajectories of length equal to the effective horizon (computed in the previous part), starting from $s = 1$ following the behavior policy π^b in the underlying model. Then, use importance sampling to calculate the estimated value function $\hat{V}^{\pi^t}(1)$ in the initial state.

4. Compute the error between the estimated value function $\hat{V}^{\pi^t}$ from the sample finite trajectories (computed in Part 3) and the true infinite-horizon value function V^{π^t} (computed in Part 1) at state $s = 1$.

[**Note:** For this problem, attach all your code in a programming language of your choice to the end of your submission. You may get help from or use the existing implementations of the Monte Carlo algorithms.]