# 10-703 Deep RL and Controls
# Homework 1
# Spring 2018

Released: Monday, Feb. 5, 2018

Due: Monday, Feb. 19, 2018 @ 11:59:59 PM EST

## Instructions

Submit the following to Gradescope by Monday, Feb. 19, 11:59:59 PM EST:

- Your write-up, written in LaTeX and submitted as a PDF file. Handwritten solutions will not be accepted.
- Your code for Problem 2

You are allowed 7 grace days in total for your homeworks. These do not need to be requested or mentioned in emails; we will automatically apply them to students who submit late. We will not give any further extensions so make sure you only use them when you are absolutely sure you need them. Students who submit late after their grace days have been used will be subject to 10% of their score for that assignment every additional day used.

You can discuss the homework problems with your classmates, but your code and your write-up must be your own. You are expected to comply with the University Policy on Academic Integrity and Plagiarism[1].

## Problem 1 (13+7=20 pts)

Consider an environment in which our agent requires caffeine to function[2]. Because caffeine is so important to our agent, we would like the agent to find a policy that will always lead it to the shortest path to coffee.

In order to apply optimal control techniques such as value iteration and policy iteration, we first need to model this scenario as a Markov decision process (MDP). Note that there may be many different ways to define each of the MDP components for a given problem. Recall that an MDP is defined as a tuple $(S, A, P, R, \gamma)$ where:

[1] `https://www.cmu.edu/policies/`

[2] If it helps, you can think of the agent as a graduate student.

$S$ is the (finite) set of all possible states.

$A$ is the (finite) set of all possible actions.

$P$ is the transition function $P : S \times S \times A \to [0, 1]$, which maps $(s', s, a)$ to $P(s'|s, a)$, i.e., the probability of transitioning to state $s' \in S$ when taking action $a \in A$ in state $s \in S$. Note that $\sum_{s' \in S} P(s'|s, a) = 1$ for all $s \in S, a \in A$.

$R$ is the reward function $R : S \times A \times S \to \mathbb{R}$, which maps $(s, a, s')$ to $R(s, a, s')$, i.e., the reward obtained when taking action $a \in A$ in state $s \in S$ and arriving at state $s' \in S$.

$\gamma$ is the discount factor, which controls the relative importance of short-term and long-term rewards. We generally have $\gamma \in [0, 1)$, where smaller values mean more discounting of rewards.

For this problem, we represent states by $(x, y, o)$ notation, where $x$ and $y$ are the horizontal and vertical coordinates of the agent's location, respectively, and $o$ is the agent's orientation which is one of $\{N, E, W, S\}$ (North, East, West, South). The agent's current orientation is the only direction it can move. The agent is able to move forward, turn right, or turn left (deterministically). All actions are available in all states. More specifically, the action space is $|A| = \{F, R, L\}$ where:

- F (move Forward): The agent moves forward along its current orientation, changing the agent's location but not its orientation.
- R (turn Right): The agent turns right, changing the agent's orientation but not its location.
- L (turn Left): The agent turns left, changing the agent's orientation but not its location.

Walls are represented by thick black lines. The agent cannot move through walls. If the agent attempts to move through a wall, it will remain in the same state.

When the agent reaches the coffee cup in any orientation, the episode ends. Another way to think of this is that every action in the coffee cup state keeps the agent in the coffee cup state.

## Problem 1, Part I: Markov Decision Process (13 pts)

For this section, consider the instance shown in Figure 1. The goal, displayed as a coffee cup, is located at $(3, 3)$. Using the above problem description answer the following questions:

a) How many states and actions are in this MDP (i.e., what are $|S|$ and $|A|$)? What is the dimensionality of the transition function $P$? (2 pts)

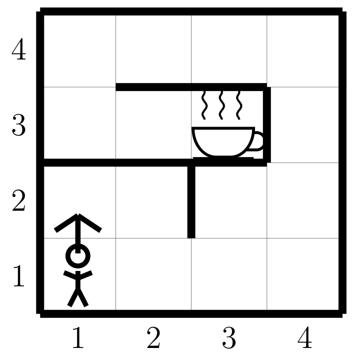


Figure 1: MDP for Problem 1, Part I. The goal is for the agent to have a policy that always leads it on the shortest path to the coffee at coordinate $(3, 3)$. In this instance, the agent's current state is $(1, 1, N)$

Solution

States $= 4 * 16 = 64$
Actions $= 3$
dimensionality of transition function = (# of states) $\times$ (# of actions) $= 64 \times 3$

b) Fill in the probabilities for the transition function $P$. (1 pt)

| | | $s'$ | | | |
|---|---|---|---|---|---|
| $s$ | $a$ | (1,2,N) | (1,1,S) | (1,4,N) | (1,4,S) |
| (1,1,S) | F | | | | |
| (1,1,N) | F | | | | |
| (1,4,E) | R | | | | |

Solution

| | | $s'$ | | | |
|---|---|---|---|---|---|
| $s$ | $a$ | (1,2,N) | (1,1,S) | (1,4,N) | (1,4,S) |
| (1,1,S) | F | 0 | 1 | 0 | 0 |
| (1,1,N) | F | 1 | 0 | 0 | 0 |
| (1,4,E) | R | 0 | 0 | 0 | 1 |

c) Describe a reward function $R : S \times A \times S$ and a discount factor $\gamma$ that will lead to an optimal policy giving the shortest path to the coffee cup from all states. Does the value of $\gamma \in (0, 1)$ affect the optimal policy in this case? Explain. (2 pts)

**Solution**

I will design the reward function such that the agent receives

(a) If the next state is not coffee cup, give reward of "-1"

(b) If the next step is coffee cup, give reward of "10"

I am choosing to give negative reward for each step not ending at coffee cup to prevent sub=optimal solutions as answer.

I will choose $\gamma = 1$. Infact, I can chose any $\gamma$ value here as it will not change the optimal policy.

d) How many possible deterministic policies are there, including both optimal and non-optimal policies? (2 pts)

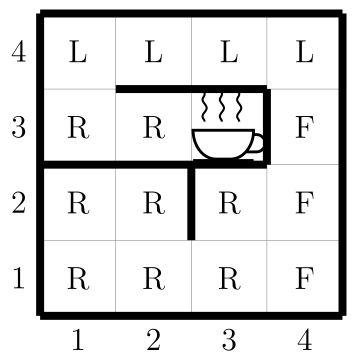
**Solution**

4.

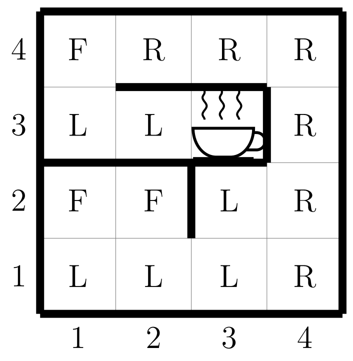
e) What is an optimal policy for this MDP? Draw the 4 grids (one for each orientation) and label each cell with one of {F,R,L}, for Forward, turn Right and turn Left, respectively. Is your policy deterministic or stochastic? (There may be multiple optimal policies. Pick one and show it.) (4 pts)

**Solution**

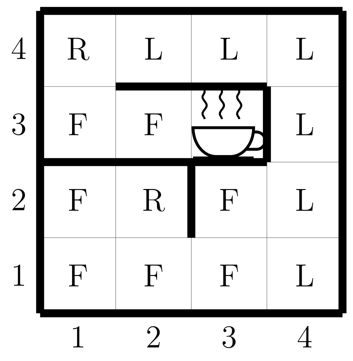
NORTH



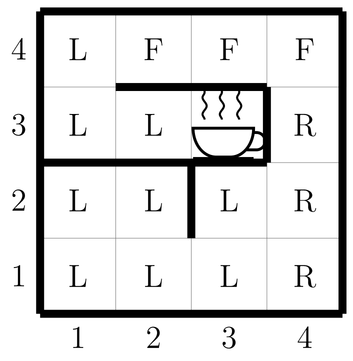
SOUTH



EAST



WEST



f) Is there any advantage to having a stochastic policy for this MDP? Explain. In general, what are the advantages of using a stochastic policy? (2 pts)

Solution

## Problem 1, Part II: Reward and Discount Factor (7 pts)

In this section, consider the instance in Fig. 3. This time, the agent has a choice between (1) a small coffee closer to the agent and (2) a large coffee further away from the agent. We will use this scenario to explore the interaction between optimal policy, discount factor, and reward function. Again, each coffee cup is still an absorbing state: when the agent reaches one of the coffee cups, the episode ends.

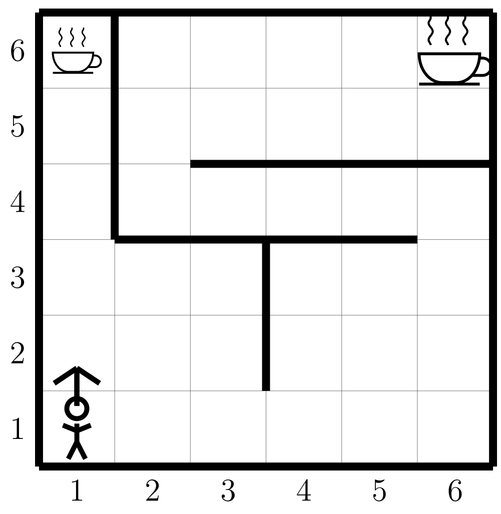


Figure 2: MDP for Problem 1, Part II. The agent starts in state $(1, 1, N)$. This time, the agent has a choice between a small coffee at $(1, 6)$ closer to the agent and a large coffee at $(6, 6)$ further away from the agent.

a) For a fixed reward function of +10 for the large coffee, +1 for the small coffee and 0 elsewhere, describe the optimal policy behavior for each value of the discount factor $\gamma$. (2 pts)

Solution

For the case when agent is starting at (1, 1, N) and there is no penalty on time (number of steps), the value of state is $max(\gamma^{n_{small}-1} \times 1, \gamma^{n_{large}-1} \times 10)$; where $n_{small} and n_{large}$ are the minimum number of steps that the agent can take to reach the small coffee and large coffee respectively

From start state to small coffee, the agent needs to take at least 5 steps. Similarly from start state to large coffee, the agent needs to take at least 23 steps.

So the agent will go to small coffee if:

$$\gamma^4 > \gamma^{22} \times 10 \tag{1}$$

$$\gamma < 0.1^{\frac{1}{18}} \tag{2}$$

$$\gamma < 0.8799 \tag{3}$$

(a) $\gamma < 0.8799$, the agent will follow this policy: $F \to F \to F \to F \to F$

(b) $\gamma > 0.8799$, the agent will follow this policy: $R \to F \to F \to F \to F \to F \to$

$L \to F \to F \to F \to L \to F \to F \to F \to R \to F \to F \to R \to F \to F \to F$

(c) $\gamma = 0.8799$, the agent is neutral towards goal position and will follow either of the above two policies with 50% probability.

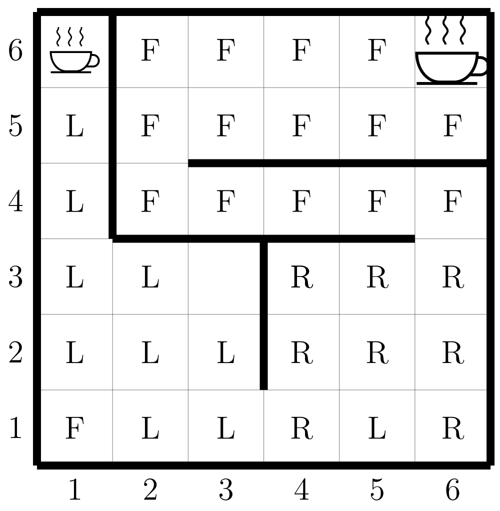


Figure 3: MDP for Problem 1, Part II. The agent starts in state $(1, 1, N)$. This time, the agent has a choice between a small coffee at $(1, 6)$ closer to the agent and a large coffee at $(6, 6)$ further away from the agent.

b) Suppose the discount factor is $\gamma = 1$ and the reward function gives +10 for the large coffee and +1 for the small coffee. Here we will add a negative reward $r < 0$ to every action the agent takes (think of it as a penalty for using energy). Describe the optimal policy behavior for each value of $r$. (2 pts)

**Solution**

Considering the case when agent is starting at (1, 1, N), there is negative reward to every action and $\gamma = 1$. The value of state is $max((n_{small} - 1) \times r + 1, (n_{large} - 1) \times r + 10)$; where $n_{small} and n_{large}$ are the minimum number of steps that the agent can take to reach the small coffee and large coffee respectively

From start state to small coffee, the agent needs to take at least 5 steps. Similarly from start state to large coffee, the agent needs to take at least 23 steps.

So the agent will go to small coffee if:

$$4 \times r + 1 > 22 \times r + 10 \quad (4)$$
$$18 \times r < -9 \quad (5)$$
$$r < -0.5 \quad (6)$$

(a) $r < -0.5$, the agent will follow this policy: $F \rightarrow F \rightarrow F \rightarrow F \rightarrow F$

(b) $r > -0.5 and r < 0$, the agent will follow this policy: $R \rightarrow F \rightarrow F \rightarrow F \rightarrow F \rightarrow F \rightarrow L \rightarrow F \rightarrow F \rightarrow F \rightarrow L \rightarrow F \rightarrow F \rightarrow F \rightarrow R \rightarrow F \rightarrow F \rightarrow R \rightarrow F \rightarrow F \rightarrow F$

(c) $r = -0.5$, the agent is neutral towards goal position and will follow either of the above two policies with 50% probability.

c) What would happen to the optimal policy if we chose $r$ to be a positive number in the previous question? (1 pts)

Solution

The problem will become infinite horizon and will not converge.

d) For a fixed discount factor of 0.9, find a reward function for which the optimal policy will prefer going to the small coffee from the starting state, and a reward function for which the optimal policy will prefer going to the large coffee from the starting state. (2 pts)

Solution

For the case when agent is starting at (1, 1, N) and let's assume there is no penalty on taking actions, the value of state is $max(\gamma^{n_{small}-1} \times r_{small}, \gamma^{n_{large}-1} \times r_{large})$; where $n_{small} and n_{large}$ are the minimum number of steps that the agent can take to reach the small coffee and large coffee respectively and $r_{small} and r_{large}$ are the end rewards for getting small coffee and large coffee respectively
From start state to small coffee, the agent needs to take at least 5 steps. Similarly from start state to large coffee, the agent needs to take at least 23 steps.
So the agent will go to small coffee if:

$$\gamma^4 \times r_{small} > \gamma^{22} \times r_{large} \tag{7}$$
$$Let r_{small} = 1 \tag{8}$$
$$\frac{1}{0.9^{18}} > r_{large} \tag{9}$$
$$r_{large} < 6.66 \tag{10}$$

Hence, to prefer small coffee, I will assign zero reward for taking action and +1 for getting small coffee and +6 for getting large coffee

To prefer large coffee, I will assign zero reward for taking action and +1 for small coffee and +10 for large coffee.

## Problem 2 (37+10+8=55 pts)

In this problem, you will implement value iteration and policy iteration. We will be working with different versions of the OpenAI Gym FrozenLake environment[3], defined in `deeprl_hw1/lake_envs.py`. Specific coding instructions are provided in the source code and `README.md`. Function templates are provided in `deeprl_hw1/rl.py` for you to fill in. You may use either Python 2.7 or Python 3.

In this domain , the agent starts at a fixed starting position, marked with "S". The agent can move up, down, left, and right.

- In the deterministic environments (`Deterministic-*-FrozenLake-v0`), the up action will always move the agent up, the left will always move left, etc.
- In the stochastic environments (`Stochastic-*-FrozenLake-v0`), the ice is slippery, so the agent won't always move in the direction it intends. Specifically, each action will move the agent in the intended direction with probability $\frac{1}{3}$, to the right of the intended direction with probability $\frac{1}{3}$, and to the left of the intended direction with probability $\frac{1}{3}$.

We have provided two different maps, a $4 \times 4$ map and a $8 \times 8$ map:

```
GHFS
FHHF
FFHF
FFFF
```

```
FFFSFFFF
FFFFFFFF
HHHHFHFF
FFFFFFHF
FFFFFFFF
FHFFFFHF
FHFFHFHH
FGHFFFFF
```

There are four different tile types: Start (S), Frozen (F), Hole (H), and Goal (G).

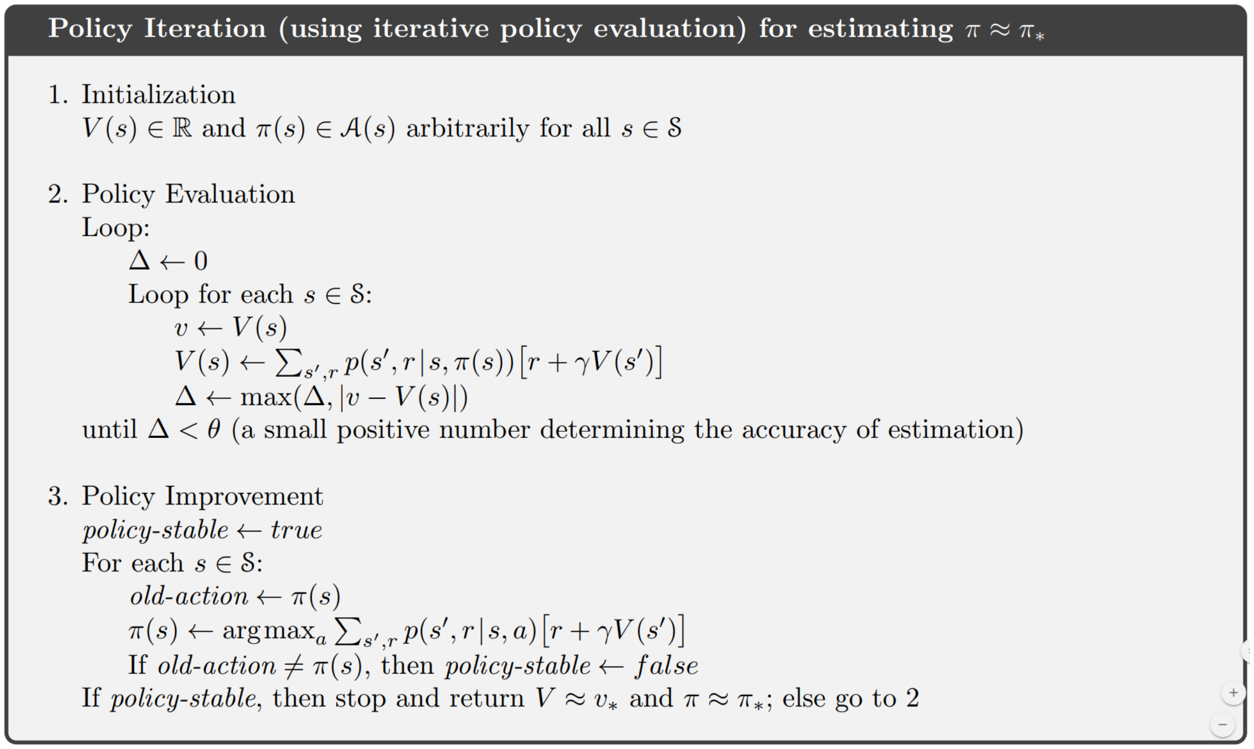
- The agent starts in the Start tile at the beginning of each episode.
- When the agent lands on a Frozen or Start tile, it receives 0 reward.
- When the agent lands on a Hole tile, it receives 0 reward and the episode ends.
- When the agent lands on the Goal tile, it receives +1 reward and the episode ends.

States are represented as integers numbered from left to right, top to bottom starting at zero. For example in a $4 \times 4$ map, the upper-left corner is state 0 and the bottom-right corner is state 15:

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

[3]`https://gym.openai.com/envs/FrozenLake-v0`

**Note:** Be careful when implementing value iteration and policy evaluation. Keep in mind that in this environment, the reward function depends on the current state, the current action, and the **next state**. Also, terminal states are slightly different. Think about the backup diagram for terminal states and how that will affect the Bellman equation.



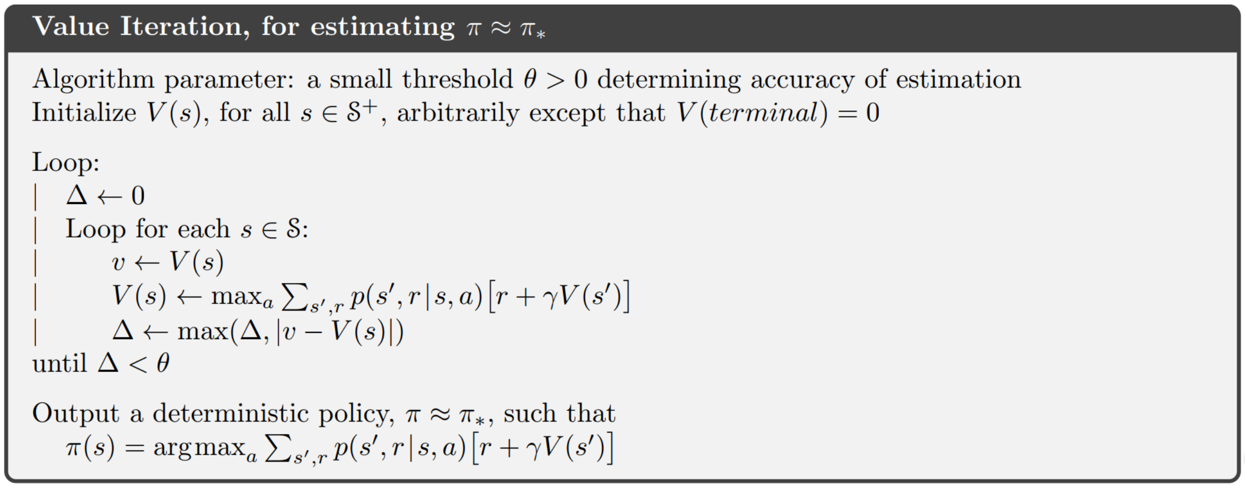
**Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$**

```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   $policy\text{-}stable \leftarrow true$
   For each $s \in \mathcal{S}$:
       $old\text{-}action \leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
       If $old\text{-}action \neq \pi(s)$, then $policy\text{-}stable \leftarrow false$
   If $policy\text{-}stable$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```

Figure 4: Policy iteration, taken from Section 4.3 of Sutton & Barto's RL book (2018).



**Value Iteration, for estimating $\pi \approx \pi_*$**

```
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
|   $\Delta \leftarrow 0$
|   Loop for each $s \in \mathcal{S}$:
|       $v \leftarrow V(s)$
|       $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
|       $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
    $\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
```

Figure 5: Value iteration, taken from Section 4.4 of Sutton & Barto's RL book (2018).

## Problem 2, Part I: Deterministic Environment (37 pts)

In this section, we will use the deterministic versions of the FrozenLake environment. Answer the following questions for the maps `Deterministic-4x4-FrozenLake-v0` and `Deterministic-8x8-FrozenLake-v0`.

a) Find the optimal policy using **synchronous policy iteration**. Specifically, you will implement `policy_iteration_sync()` in `deeprl_hw1/rl.py`, writing the policy evaluation and policy improvement steps in `evaluate_policy_sync()` and `improve_policy()`, respectively. Record (1) the time taken for execution, (2) the number of policy improvement steps, and (3) the total number of policy evaluation steps. Use a discount factor of $\gamma = 0.9$. Use a stopping tolerance of $\theta = 10^{-3}$ for the policy evaluation step. (5 pts)

| **Environment** | **Time (ms)** | **# Policy Improvement Steps** | **Total # Policy Evaluation Steps** |
|---|---|---|---|
| Deterministic-4x4 | | | |
| Deterministic-8x8 | | | |

Solution

| **Environment** | **Time (ms)** | **# Policy Improvement Steps** | **Total # Policy Evaluation Steps** |
|---|---|---|---|
| Deterministic-4x4 | 5.4 | 7 | 44 |
| Deterministic-8x8 | 29.8 | 12 | 132 |

b) What is the optimal policy for this map? Show as a grid of letters with "U", "D", "L", "R" representing the actions up, down, left, right respectively. See Figure 6 for an example of the expected output. (3 pts)

```
LLLL
DDDD
UUUU
RRRR
```

Figure 6: An example (deterministic) policy for a 4 × 4 map of the `FrozenLake-v0` environment. L, D, U, R represent the actions up, down, left, right respectively.

Solution

For 4x4 map:
LLRD
ULLD
ULLD
ULLL

For 8x8 map:
DDDDDLLL
RRRRDLLL
LLLLDLRD
DLLLLLLD
DLLLLLLL
DLULLLLU
DLULLULL
RLLULLLL

c) Find the value function of this policy. Plot it as a color image, where each square shows its value as a color. See Figure 7 for an example. (3 pts)

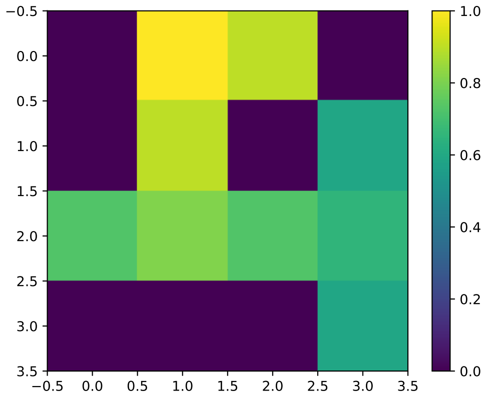


Figure 7: Example of value function color plot for a 4 × 4 map of the `FrozenLake-v0` environment. Make sure you include the color bar or some kind of key.

Solution

4X4: Figure 8
8X8: Figure 9

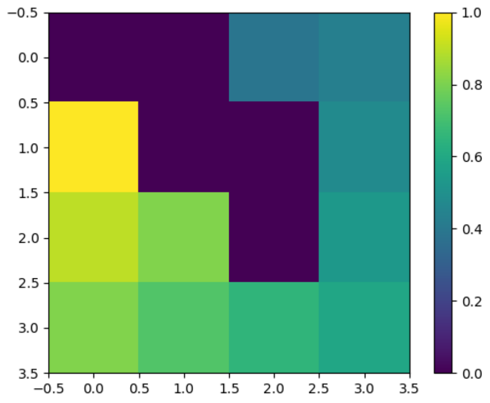


Figure 8: Solution of value function color plot for a $4 \times 4$ map of the `FrozenLake-v0` environment.

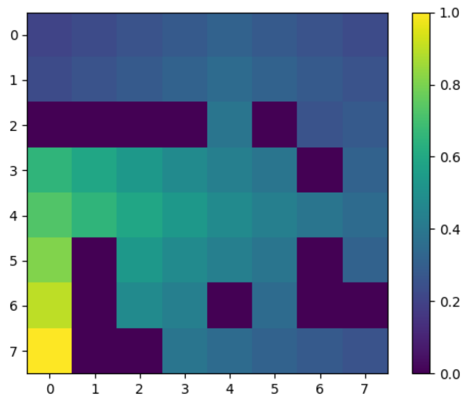


Figure 9: Solution of value function color plot for a $8 \times 8$ map of the `FrozenLake-v0` environment.

d) Find the optimal value function directly using **synchronous value iteration**. Specifically, you will implement `value_iteration_sync()` in `deeprl_hw1/rl.py`. Record the time taken for execution, and the number of iterations it took to converge. Use $\gamma = 0.9$ Use a stopping tolerance of $10^{-3}$. (5 pts)

| **Environment** | **Time (ms)** | **# Iterations** |
|---|---|---|
| Deterministic-4x4 | | |
| Deterministic-8x8 | | |

Solution

| **Environment** | **Time (ms)** | **# Iterations** |
|---|---|---|
| Deterministic-4x4 | 1.31 | 10 |
| Deterministic-8x8 | 7.92 | 16 |

e) Plot this value function as a color image, where each square shows its value as a color. See Figure 7 for an example. (3 pts)

Solution

4X4: Figure 10
8X8: Figure 11

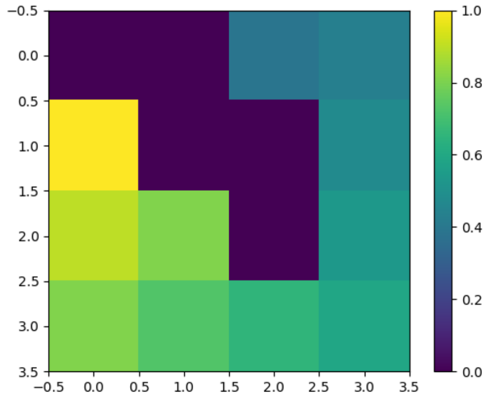


Figure 10: Solution of value function color plot for a $4 \times 4$ map of the `FrozenLake-v0` environment.

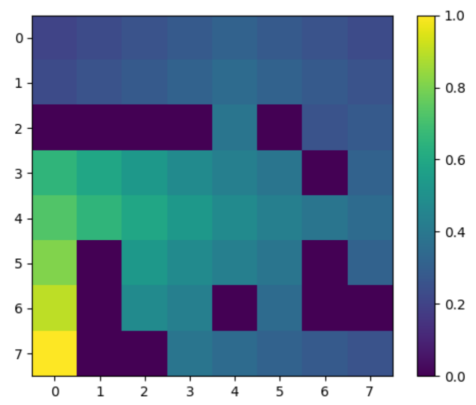


Figure 11: Solution of value function color plot for a $8 \times 8$ map of the `FrozenLake-v0` environment.

f) Which algorithm was faster, policy iteration or value iteration? Which took fewer iterations? Are there any differences in the value function? (2 pts)

Solution

Value-iteration was faster than policy iteration.
Value-iteration took fewer iterations than policy-iteration
There is no difference between the value function derived from two approaches: policy-iteration versus value-iteration

g) Convert the optimal value function to the optimal policy. Show the policy a grid of letters with "U", "D", "L", "R" representing the actions up, down, left, right respectively. See Figure 6 for an example of the expected output. (3 pts)

Solution

For 4x4 map:
LLRD
ULLD
ULLD
ULLL

For 8x8 map:
DDDDDLLL
RRRRDLLL
LLLLDLRD
DLLLLLLD
DLLLLLLL

DLULLLLU
DLULLULL
RLLULLLL

h) Implement **asynchronous policy iteration** using two heuristics:

1. The first heuristic is to sweep through the states in the order they are defined in the gym environment. Specifically, you will implement `policy_iteration_async_ordered()` in `deeprl_hw1/rl.py`, writing the policy evaluation step in `evaluate_policy_async_ordered()`.
2. The second heuristic is to choose a random permutation of the states at each iteration and sweep through all of them. Specifically, you will implement `policy_iteration_async_randperm()` in `deeprl_hw1/rl.py`, writing the policy evaluation step in `evaluate_policy_async_randperm()`.

Fill in the table below with the results for Deterministic-8x8-FrozenLake-v0: (5 pts)

| **Heuristic** | **Time (ms)** | **Policy Improvement Steps** | **Total Policy Evaluation Steps** |
|---|---|---|---|
| Ordered | | | |
| Randperm | | | |

Solution

| **Heuristic** | **Time (ms)** | **Policy Improvement Steps** | **Total Policy Evaluation Steps** |
|---|---|---|---|
| Ordered | 23.29 | 12 | 78 |
| Randperm | 27.77 | 12 | 95 |

i) Implement **asynchronous value iteration** using two heuristics:

1. The first heuristic is to sweep through the states in the order they are defined in the gym environment. Specifically, you will implement `value_iteration_async_ordered()` in `deeprl_hw1/rl.py`.
2. The second heuristic is to choose a random permutation of the states at each iteration and sweep through all of them. Specifically, you will implement `value_iteration_async_randperm()` in `deeprl_hw1/rl.py`.

Fill in the table below with the results for Deterministic-8x8-FrozenLake-v0: (5 pts)

| **Heuristic** | **Time(s)** | **# Iterations** |
|---|---|---|
| Ordered | | |
| Randperm | | |

Solution

| **Heuristic** | **Time(s)** | **# Iterations** |
|---|---|---|
| Ordered | 6.4 | 12 |
| Randperm | 6.74 | 8 |

j) Write an agent that executes the optimal policy. Record the total cumulative discounted reward. Does this value match the value computed for the starting state? If not, explain why. (3 pts)

| **Env** | $V(s_0)$ **Computed** | $V(s_0)$ **Simulated** |
|---|---|---|
| 4x4 | | |
| 8x8 | | |

Solution

| **Env** | $V(s_0)$ **Computed** | $V(s_0)$ **Simulated** |
|---|---|---|
| 4x4 | 0.43046 | 0.43046 |
| 8x8 | 0.28242 | 0.28242 |

## Problem 2, Part II: Stochastic Environment (10 pt)

In this section, we will use the stochastic versions of the FrozenLake environment. Answer the following questions for the maps `Stochastic-4x4-FrozenLake-v0` and `Stochastic-8x8-FrozenLake-v0`.

a) Using **synchronous value iteration**, find the optimal value function. Record the time taken for execution and the number of iterations required. Use a stopping tolerance of $10^{-3}$. Use $\gamma = 0.9$. (2 pts)

| **Environment** | **Time (ms)** | **# Iterations** |
|---|---|---|
| Stochastic-4x4 | | |
| Stochastic-8x8 | | |

Solution

| Environment | Time (ms) | # Iterations |
|---|---|---|
| Stochastic-4x4 | 4.05 | 28 |
| Stochastic-8x8 | 18.67 | 25 |

b) Plot the value function as a color map like in Part I. Is the value function different compared to the deterministic versions of the maps? (2 pts)

Solution

4X4: Figure 12
8X8: Figure 13

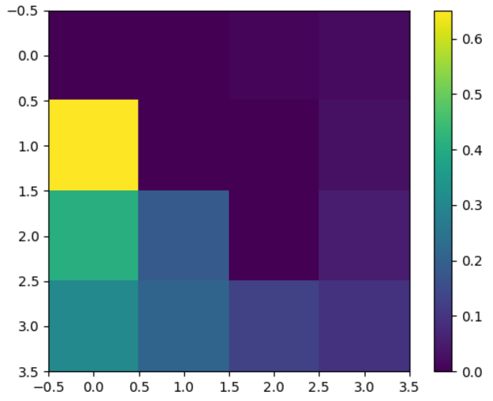


Figure 12: Solution of value function color plot for a $4 \times 4$ map of the `FrozenLake-v0` environment.

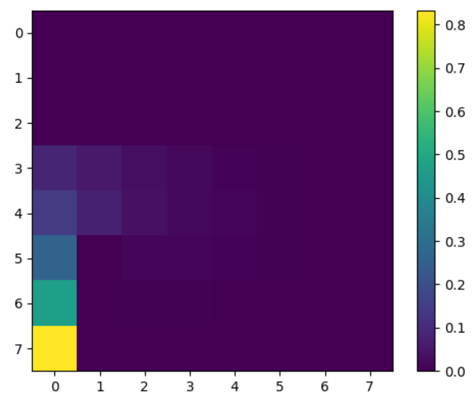


Figure 13: Solution of value function color plot for a $8 \times 8$ map of the `FrozenLake-v0` environment.

c) Convert this value function to the optimal policy and include it in the writeup. Does this optimal policy differ from the optimal policy in the deterministic map? If so, pick a state where the policy differs and explain why the action is different. (3 pts)

**Solution**

For 4x4 map:
LLRR
LLLR
LLLR
LLDD

For 8x8 map:
URRRRLLL
UUUURUUD
LLLLLLRR
DDDDDLLR
LUULLLDU
LLRUULLR
LLRLLLLL
DLLRDLDD

The optimal policy in stochastic map is different compared to deterministic map. For example: let us consider the state 3. In deterministic map, the optimal action is "DOWN" as it is the upper right corner and there is hole on the left and down is the only safe tile. Whereas, in stochastic map, the optimal action is "RIGHT". This ensures that the agent will move either down to only safe position with 0.33 probability and or stay in its location with 0.66 probability.

d) Write an agent that executes the optimal policy. Run this agent 100 times on the map and record the total cumulative discounted reward. Average the results. Does this value match the value computed for the starting state? If not explain why. (3 pts)

| **Env** | $V(s_0)$ **Computed** | $V(s_0)$ **Simulated** |
|---|---|---|
| 4x4 | | |
| 8x8 | | |

Solution

| **Env** | $V(s_0)$ **Computed** | $V(s_0)$ **Simulated** |
|---|---|---|
| 4x4 | 0.0197 | 0.026 |
| 8x8 | 0.00032 | 0.00091 |

## Problem 2, Part III: Custom Heuristic for Asynchronous DP (8 pt)

In this part, you will design a custom heuristic for asynchronous value iteration (`value_iteration_async_custom()`) to try to beat the heuristics previously defined (although this is not a requirement for full points). A potential idea for the heuristic might be to sweep through the entire state space ordered by e.g. distance to goal. You do not need to systemically sweep through the entire state space in your design (but you can do so if you so choose).

a) Fill in the table below with the results for asynchronous value iteration using your custom heuristic. Compare against the previous heuristics. Since the previous heuristics did do a systemic sweep through the entire state space, you can compare against them by measuring the number of individual state updates. You can calculate this for the old heuristics by taking # Iterations and multiplying it by the size of the state space. (5 pts)

| **Env** | **Time (s)** | **# Individual State Updates** |
|---|---|---|
| Deterministic-4x4 | | |
| Deterministic-8x8 | | |
| Stochastic-4x4 | | |
| Stochastic-8x8 | | |

Solution

| Env | Time (ms) (ms) | # Ind State Updates Updates | # Ind State Updates Ordered | # Ind State Updates RandPerm |
|---|---|---|---|---|
| D-4x4 | 0.92 | 57 | 60 | 96 |
| D-8x8 | 7.23 | 603 | 192 | 192 |
| S-4x4 | 7.5 | 420 | 336 | 400 |
| S-8x8 | 27.27 | 1575 | 336 | 336 |

b) Provide an intuitive description of your heuristic (max 250 word). (3 pts)

Solution

I have implemented a priority queue (decreasing order) for states. Before constructing the queue, I check if the state is terminal and add the state to queue (with zero key value) only if the state is not terminal. In each loop, I pop the first element from queue, update its state value using bellman equation and push the state back in the queue with key value as:

(a) $\Delta = -|old\_value(state) - new\_value(state)|$ if $\Delta > tolerance$

(b) zero if $\Delta < tolerance$. In this case, I also increase the counter (think of it as number of times the state got visited)and use it in priority queue. For states with same priority value get arranged with decreasing counter values.

My stopping criteria is that all the states in the queue have values converged. I am maintaining a counter to check if convergence is achieved.

## Problem 3 (25 pts)

For this problem, assume the MDP has finite state and action spaces. Let $V^\pi(s)$ be the value of policy $\pi$ when starting at state $s$. That is:

$$V^\pi(s) := \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \mid s_t = s\right]$$

A policy $\pi$ is called *deterministic* if for all states $s \in S$, there exists an action $a \in A$ such that $\pi(a|s) = 1$.

(a) Assume $\gamma < 1$. Show that for all policies $\pi$ and states $s \in S$, the value function $V^\pi(s)$ is well-defined, i.e., (i) the infinite sum $\sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$ converges even if the episode never terminates and (ii) the outer expectation $\mathbb{E}_\pi[\cdot]$ is well-defined. (5 pts)

Solution

(i) As this is an example of finite MDP, the number of states transactions $\tau \in T$ are finite. Also let $r_\tau$ represents the reward corresponding to $\tau$ transaction. Hence we can split the infinite sum equation as:

$$\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \leq \sum_{k=0}^{\infty} \gamma^k \times \sum_{\tau \in T} r_\tau \tag{11}$$

Now, $\sum_{k=0}^{\infty} \gamma^k$ is finite as $\gamma < 1$.

Also, $\sum_{\tau \in T} r_\tau$ finite because the number of terms is finite and the value of each term is also finite.

$\implies \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$ converges even if the episode never terminates.

(ii) $\mathbb{E}_\pi[\cdot] = \sum_{a \in A} p(a|s) \times [\cdot]$

(b) Show that for every state $s \in S$, there exists a deterministic policy $\pi_s$ that is optimal for that state $s$ amongst deterministic policies (i.e., $V^{\pi_s}(s) \geq V^\pi(s)$ for every deterministic policy $\pi$). (3 pts)

Solution

For every state $s \in S$, we can select a deterministic policy $\pi$ that gives the maximum value for that state $s$ and call it the optimal policy for that state $s$. Hence,
$\pi_s(s) = argmax_\pi V^\pi(s)$
$\pi_s(s) = argmax_\pi \ \max_{a \in A}(\pi(a|s) \times (r(s'|a,s) + \gamma V^\pi(s')))$

(c) Given state-optimal deterministic policies $\pi_s$ as defined in part (b), write down a deterministic policy $\pi^*$ that is optimal for all states among deterministic policies (i.e., $V^{\pi^*}(s) \geq V^\pi(s)$ for all states $s$ and deterministic policies $\pi$). (3 pts)

Solution

$\pi^*(s) = \pi_s(s)$
$V^{\pi^*}(s) = V^{\pi_s}(s) \geq V^\pi(s)$

(d) Assume that the discount factor is 1. Give an example MDP and policy where the value function is infinite, i.e., the value of the infinite sum diverges. (2 pt)

Solution

(e) Specify a reasonable criterion which can be used for deciding whether one infinite-valued policy is superior to another. (3 pts)

Solution

(f) Assume that the discount factor is 1. Specify an MDP with multiple states and actions that has an infinite-valued optimal policy according to this criterion. Specify the policy and show that it is optimal. (4 pts)

Solution

(g) Assume that the discount factor is 1. Specify an MDP which has multiple infinite-valued policies for which there exists no clear criterion for deciding which is superior. Specify those policies and explain your reasoning. (4 pts)

Solution

(h) Give a class of infinite-horizon MDPs where $\gamma = 1$ but every stationary policy has finite value. (1 pts)

Solution