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, γ) 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 $s' \in S$.

γ 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)

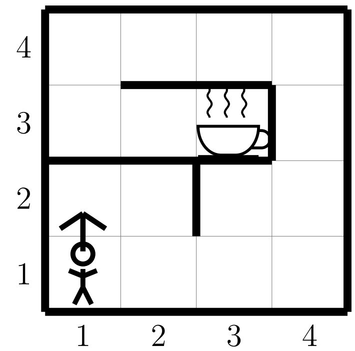


Figure 1: MDP for Problem 1, Part I. The goal is for the agent to have a policy that always leads it on the shortest path to the coffee at coordinate $(3,3)$. In this instance, the agent's current state is $(1,1,N)$

Solution

$|S| = 4 * 4 * 4 = 64$
$|A| = 3$
The dimensionality of the transition function P is $|S| * |S| * |A| = 12288$

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

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 γ 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

$R(s,a,s') = -1$ for all $s \in S, a \in A, s' \in S$, $\gamma = 0.9$.
The value of $\gamma \in (0,1)$ doesn't affect the optimal policy in this case. Because there is only one goal in this MDP.

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

Solution
There are infinite possible deterministic policies.

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

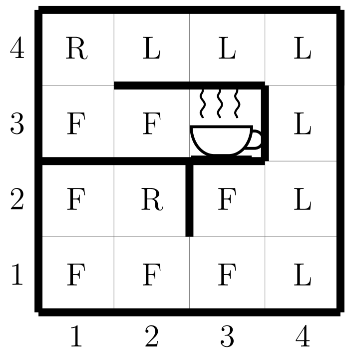
North:

4	L	L	L	L
3	R	R		F
2	R	R	R	F
1	R	R	R	F
	1	2	3	4

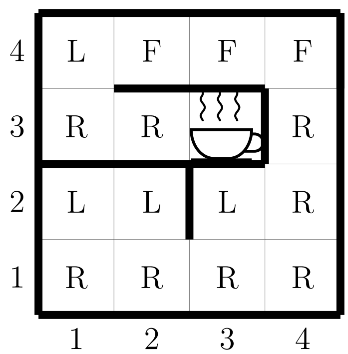
South:

4	F	R	R	R
3	L	L		R
2	F	F	L	L
1	L	L	L	L
	1	2	3	4

East:



West:



My policy is deterministic.

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

There is no advantage to having a stochastic policy for this MDP.
In general, when the environment is stochastic or the environment can't be fully observed, then stochastic policy would be better than deterministic policy.

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

In this section, consider the instance in Fig. 2. 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.

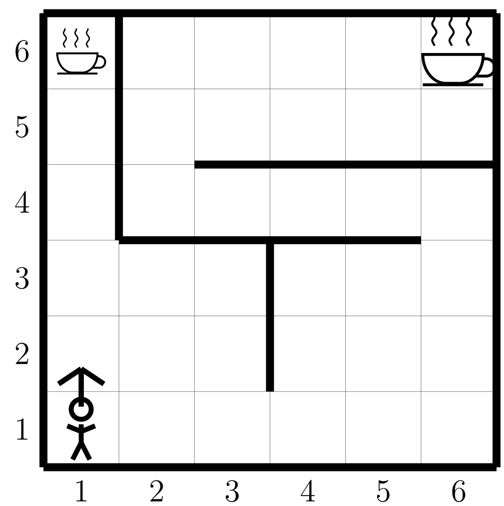


Figure 2: MDP for Problem 1, Part II. The agent starts in state $(1, 1, N)$. This time, the agent has a choice between a small coffee at $(1, 6)$ closer to the agent and a large coffee at $(6, 6)$ further away from the agent.

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 γ. (2 pts)

Solution

$\gamma^4 = \gamma^{22} \times 10$
$\gamma = 0.1^{-18} \approx 0.8799$
For $\gamma \in (0.1^{-18}, 1)$, the optimal policy behavior for the agent would always be heading to the large coffee.
For $\gamma = 0.1^{-18}$, the optimal policy behavior for the agent is heading to the large coffee or the small coffee.
For $\gamma \in (0, 0.1^{-18})$, the optimal policy behavior for the agent would always be heading to the small coffee.
For $\gamma = 0$, the optimal policy behavior for the agent is heading to the large coffee or the small coffee.

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

If $-0.5 < r < 0$, the optimal policy behavior for the agent would always be heading to the large coffee.
If $r < -0.5$, the optimal policy behavior for the agent would always be heading to

the small coffee.
If $r = -0.5$, the optimal policy behavior for the agent is heading to the small coffee or the large coffee.

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 agent will keep wandering in the map and try to avoid to fall into the coffee state.

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

Prefer small coffee:
Reward function gives +10 for the large coffee and +1 for the small coffee, -1 to every action the agent takes.

Prefer large coffee:
Reward function gives +10 for the large coffee and +1 for the small coffee, 0 to every action the agent takes.

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

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

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 4×4 map and a 8×8 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 4×4 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

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.

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()`,

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-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
       old-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-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Figure 3: 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 $V(\mathit{terminal}) = 0$

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

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)

Solution

Environment	Time (ms)	# Policy Improvement Steps	Total # Policy Evaluation Steps
Deterministic-4x4	3	9	61
Deterministic-8x8	27	17	173

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 5 for an example of the expected output. (3 pts)

```
LLLL
DDDD
UUUU
RRRR
```

Figure 5: 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

```
UURD
UUUD
ULUD
UULL
```

```
RRRRDDDD
RRRRDLLD
UUUUDUUD
DDDDDDUD
DLLLLLLL
DUUUUUUU
DUUUUUUU
RUUULULL
```

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 6 for an example. (3 pts)

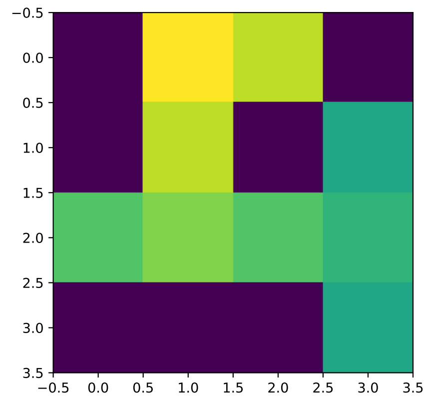
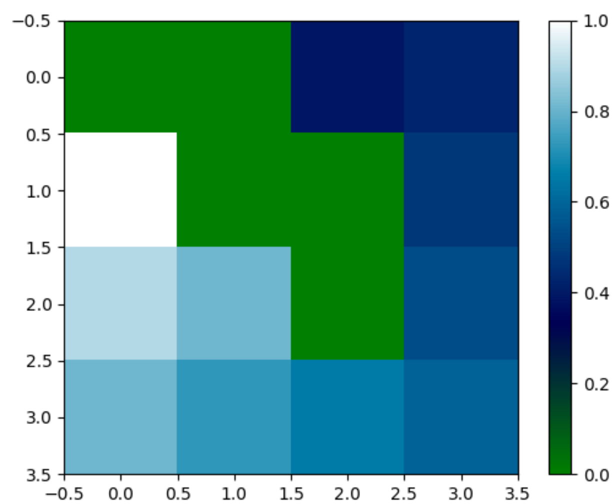
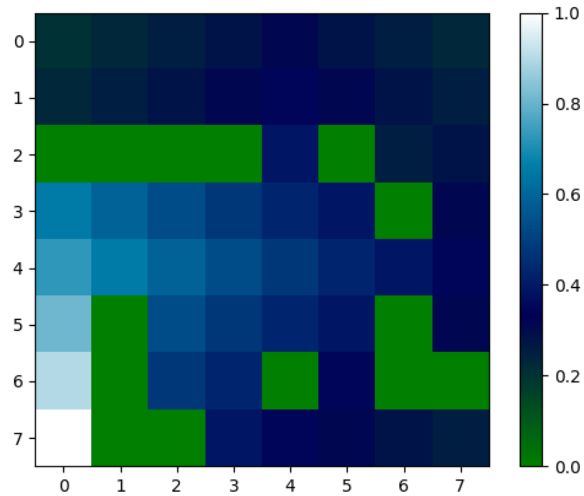


Figure 6: Example of value function color plot for a 4×4 map of the `FrozenLake-v0` environment. Make sure you include the color bar or some kind of key.

Solution





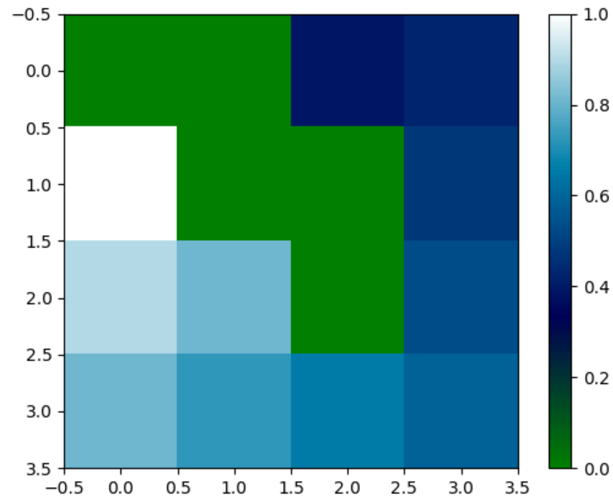
d) Find the optimal value function directly using **synchronous value iteration**. Specifically, you will implement `value_iteration_sync()` in `deeprl_hw1/rl.py`. Record the time taken for execution, and the number of iterations it took to converge. Use $\gamma = 0.9$ Use a stopping tolerance of 10^{-3}.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

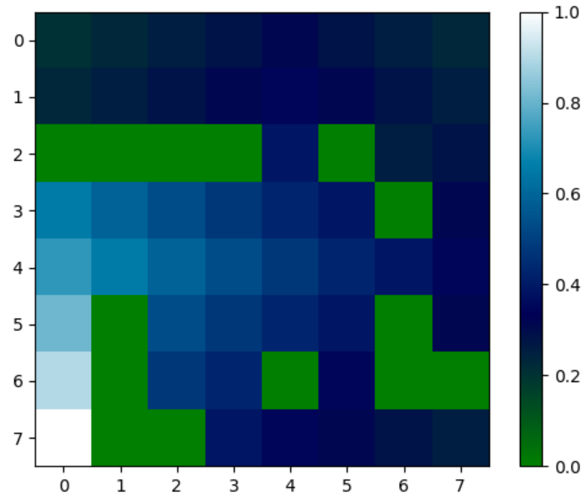
Solution

Environment	**Time (ms)**	**# Iterations**
Deterministic-4x4	2	11
Deterministic-8x8	10	17

e) Plot this value function as a color image, where each square shows its value as a color. See Figure 6 for an example. (3 pts)

Solution





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 is faster than policy iteration. Value iteration took fewer iterations. The value functions are basically the same.

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 5 for an example of the expected output. (3 pts)

Solution

```
UURD
UUUD
ULUD
UULL

RRRRDDDD
RRRRDLLD
UUUUDUUD
DDDDDDUD
DLLLLLLL
DUUUUUUU
DUUUUUUU
RUUULULL
```

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)

Solution

Heuristic	**Time (ms)**	**Policy Improvement Steps**	**Total Policy Evaluation Steps**
Ordered	17	17	115
Randperm	34	17	114

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)

Solution

Heuristic	**Time(ms)**	**# Iterations**
Ordered	8	13
Randperm	8	10

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)

Solution

Env	$V(s_0)$ **Computed**	$V(s_0)$ **Simulated**
4x4	0.43046721000000016	0.43046721
8x8	0.28242953648100017	0.282429536481

The value basically matches the value computed for the starting state.

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`.

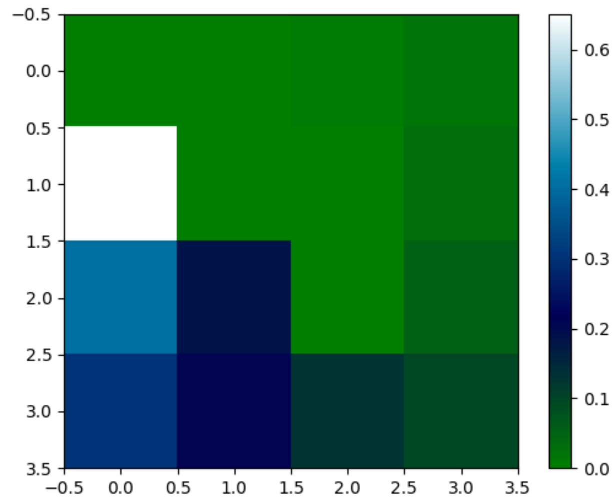
a) Using **synchronous value iteration**, find the optimal value function. Record the time taken for execution and the number of iterations required. Use a stopping tolerance of 10^{-3}. Use $\gamma = 0.9$. (2 pts)

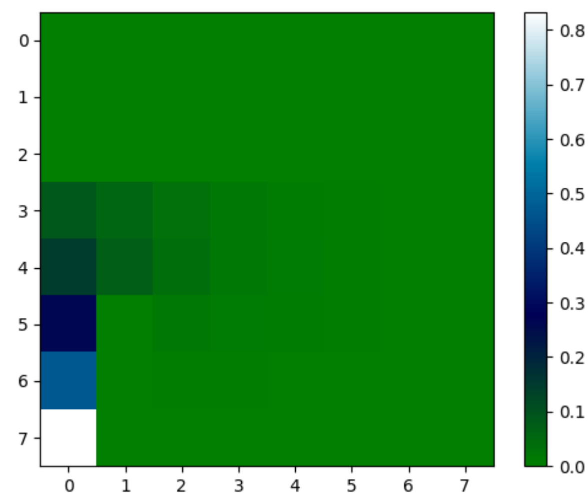
Solution

Environment	**Time (ms)**	**# Iterations**
Stochastic-4x4	4	29
Stochastic-8x8	25	26

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





The value function is different compared to the deterministic versions of the maps.

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

```
UUUR
LUUR
LDUR
LLDD
```

```
URRRRLLL
UUUURUUD
UUUURUUR
DDDDDDUR
LUULLLUU
LURUULUU
LUULURUU
DUURDLDD
```

The optimal policy differs from the optimal policy in the deterministic map. For example in 4*4 map, state 7, in deterministic map, the action is down, however in stochastic map, the action is right. Because in stochastic map, if the agent tries to get down, there is a possibility of going left, which is a hole and will end the episode.

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)

Solution

Env	$V(s_0)$ **Computed**	$V(s_0)$ **Simulated**
4x4	0.019719537665388504	0.0253528105157
8x8	0.00032865956988919576	0.00239260493057

The values are pretty close but not exactly same, because the envirment is stochastic, 100 times is not enough for the simulated value converging to the computed one.

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)

Solution

custom heuristic:

Env	**Time (ms)**	**# Individual State Updates**
Deterministic-4x4	1	66
Deterministic-8x8	3	300
Stochastic-4x4	2	220
Stochastic-8x8	9	850

Ordered heuristic:

Env	**Time (ms)**	**# Individual State Updates**
Deterministic-4x4	1	96
Deterministic-8x8	5	832
Stochastic-4x4	6	320
Stochastic-8x8	12	1344

Randperm heuristic:

Env	**Time (ms)**	**# Individual State Updates**
Deterministic-4x4	2	96
Deterministic-8x8	4	576
Stochastic-4x4	4	384
Stochastic-8x8	13	1408

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

Solution

The heuristic is to sort the state in the order of distance to goal, from nearest to farthest. For every iteration, sweep the states in this order. Also, exclude the terminal states, because the values of these states are always zero, so we don't need to sweep the entire state space.

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 π 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 π 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 π 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

Assume r_m is the maximum among all the reward in the MDP, which is a constant value.

$$\mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \mid s_t = s\right] <= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_m\right] = \mathbb{E}_\pi\left[\frac{r_m}{1-\gamma}\right] = \frac{r_m}{1-\gamma} < \infty$$

Thus $V^\pi(s)$ is well-defined.

(b) Show that for every state $s \in S$, there exists a deterministic policy π_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 π). (3 pts)

Solution

Because the action space is finite, so for every state $s \in S$, the policy space is finite. So there exists a policy where the value function is biggest. Thus for every state $s \in S$, $\exists$ a policy π_s s.t $\pi_s(s) = argmax_\pi V^\pi(s)$.

(c) Given state-optimal deterministic policies π_s as defined in part (b), write down a deterministic policy π^* 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 π). (3 pts)

Solution

$\pi^*(s) = \pi_s(s)$ for all states 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

Assume we have the same MDP as problem 1 part 1, the reward of the goal is 0. For all other states, the reward is 1. The policy is trying to not hit the goal state. So the episode will never terminate, and the value function of all other states is infinite.

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

Solution

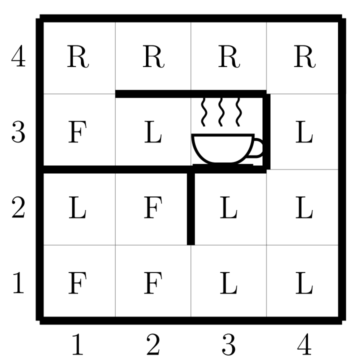
For two policies π_1 and π_2, and specific state s, the next states are s_1 and s_2 when excute $\pi_1(s)$ and $\pi_2(s)$, if the shortest path from s_1 to terminating state is shorter than the shortest path from s_2 to terminating state, theb π_1 is superior.

(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)

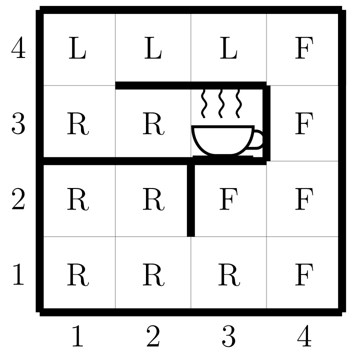
Solution

Assume we have the same MDP as problem 1 part 1, the reward of the goal state is 0. For all other states, the reward is 1. As shown below, the policy is trying to get far away from goal state. The policy is optimal because the policy will always increase the distance to the terminating state.

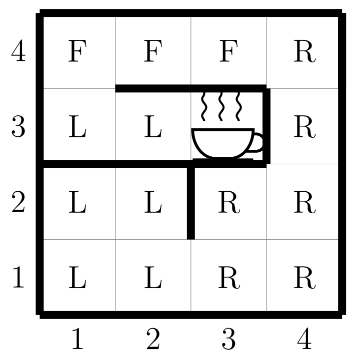
North:



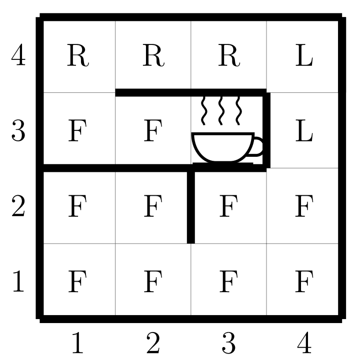
South:



East:



West:



(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

Assume we have the same MDP as problem 1 part 1, but without terminating state, for all state, the reward is 1. The policies are randomly moving around in the map. Then there is no clear criterion for deciding which is superior, because the cumulative reward for all policy is exactly same.

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

Solution

The set of MDPs which only terminating states can have the non-zero reward, all the other states have zero reward.