10-703 Deep RL and Controls
Homework 1
Spring 2017

February 1, 2017

Due February 17, 2017

Instructions

You have 15 days from the release of the assignment until it is due. Refer to gradescope for the exact time due.

You may work with a partner on this assignment. Only one person should submit the writeup and code on gradescope. Make sure you mark your partner as a collaborator on gradescope and that both names are listed in the writeup.

Writeups should be submitted as PDF.

Problem 1

Consider an environment in which our agent requires caffeine to function[1]. 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. Once the agent reaches coffee, it will stick around and enjoy it.

In order to apply optimal control techniques such as value iteration and policy iteration, we first need to model this scenario as an MDP. Recall that an MDP is defined as tuple (S, A, P, R, γ), where:

S**:** The (finite) set of all possible states.

A**:** The (finite) set of all possible actions.

P**:** 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 $s \in S, a \in A$.

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

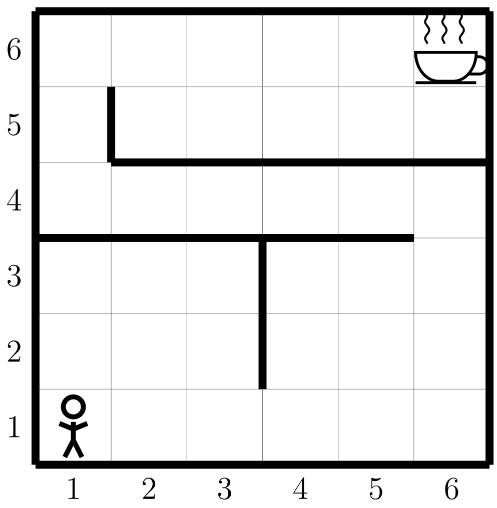

Figure 1: A particular instance of the shortest path problem. The goal is for the agent currently located in state $(1, 1)$ to have a policy that always leads it on the shortest path to the coffee in state $(6, 6)$.

R**:** The reward function $R : S \times A \times S \rightarrow \mathbb{R}$, 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**:** The discount factor, which controls how important are rewards in the future. We have $\gamma \in [0, 1)$, where smaller values mean more discounting for future rewards.

In order to encode this problem as an MDP, we need to define each of the components of the tuple for our particular problem. Note that there may be many different possible encodings.

For the questions, in this section, consider the instance shown in Figure 1. In the figure, the agent is at $(1, 1)$, but it can start at any of the grid cells. The goal, displayed as a coffee cup, is located at $(6, 6)$. The agent is able to move one square up, down, left and right (deterministically). Walls are represented by thick black lines. The agent cannot move through walls. All actions are available in all states. If the agent attempts to move through a wall, it will remain in the same state.

When the agent reaches the coffee cup, 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.

Part a (10pt)

For this section, assume we are modeling the MDP as an infinite horizon MDP. The coffee cup is still an absorbing state.

Using the above problem description answer the following questions:

a) How many states are in this MDP? (i.e. what is $|S|$).

Solution:

Each state in this MDP can be defined as a cell. Counting the number of cell, we have $|S|= 6 \times 6 = 36$ states.

b) How many actions are in this MDP? (i.e. what is $|A|$).

Solution:

Given the problem statement, each cell (state) has 4 options for movement (up, down, left, right). There are walls, but instead of not allowing this action to happen, the MDP can be modeled considering that the agent can try to move against a wall, but the next state of that action will still be the current state.

Therefore, $|A|= 4$

c) What is the dimensionality of the transition function P?

Solution:

The transition function P maps (s', s, a) to $P(s'|s, a)$. Therefore, we have a dimensionality of $dim(P) = |S|\times|S|\times|A|= 36 \times 36 \times 4 = 5184$

d) Fill in the probabilities for the transition function P.

Solution:

Admitting we have a deterministic world:

		s'				
s	**a**	(1,2)	(1,1)	(1,4)	(1,3)	(5, 6)
(1,1)	up	1	0	0	0	0
(1,1)	down	0	1	0	0	0
(1,3)	up	0	0	0	1	0
(6,6)	left	0	0	0	0	0

e) Describe a reward function $R : S \times A \times S$ and a value of γ that will lead to an optimal policy giving the shortest path to the coffee cup from all states.

Solution:

To obtain the optimal policy leading to the cup of coffee from all states, we can define the following:

$\gamma = 1$: no discounting of future rewards $R(6, 6) \times A = 1$, for any action the agent takes at $S = (6, 6)$ (also we know that the agent will not transition to any other state after arriving at $S = (6, 6)$). We set all other rewards to $R = -1$ (or any other non-zero negative value), because the user wants to get to the coffee as fast as possible.

f) Does $\gamma \in (0, 1)$ affect the optimal policy in this case? Explain why.

Solution:

In this particular example, the value of γ does not actually alter the optimal policy for getting to the cup of coffee as fast as possible. Even if future rewards are discounted, the optimal policy will still try to get to the end state as fast as possible, because it is the only state with a positive reward.

g) How many possible policies are there? (**All policies**, not just optimal policies.)

Solution:

There is an infinite number of policies to get to the cup of coffee. The agent could for example go back and forth between 2 states for any number of times before actually trying to go to the end state.

h) What is the optimal policy? Draw the grid and label each cell with an arrows in the direction of the optimal action. If multiple arrows, include the probability of each arrow. There may be multiple optimal policies, pick one and show it.

Solution:

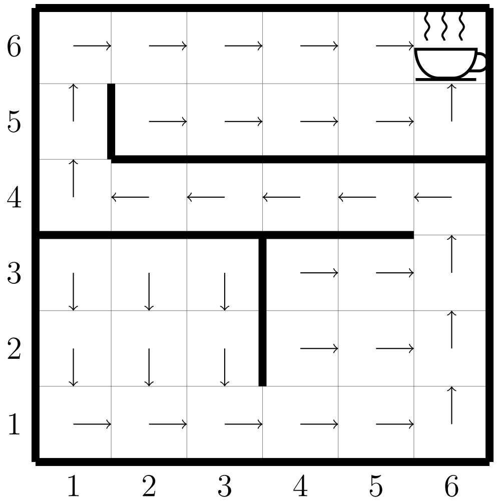

Figure 2: A particular instance of optimal policy to get to the cup of coffee as fast as possible.

i) Is your policy deterministic or stochastic?

Solution:

My policy is deterministic, because the problem statement asks us to assume a deterministic scenario. This means that the actions listed in the optimal policy will happen with probability 1.

j) Is there any advantage to having a stochastic policy? Explain.

Solution:

If we have a deterministic policy there is no room for exploration of the environment. A stochastic policy allows us to take actions out of policy π, therefore permitting us to learn possible improvements on the current policy. That is why many reinforcement learning algorithms implement an ϵ-greedy policy π.

Part b (2pt)

Now consider that our agent often goes the wrong direction because of how tired it is. Now each action has a 10% chance of going perpendicular to the left of the direction chosen and 10% chance of going perpendicular to the right of the direction chosen. Given this change answer the following questions:

a) Fill in the values for the transition function P.

Solution:

		s'		
s	**a**	(1,3)	(3,2)	(1,4)
(2,2)	up	0	0.1	0

b) Does the optimal policy change compared to Part a? Justify your answer.

Solution:

Yes, the optimal policy changes. In some states (such as $(2,3)$ and $(2,4)$) it was indifferent to go down or go left if we had a deterministic policy. However, in a stochastic policy, it is optimal to go to the right in these states, because this leaves the agent closer to the goal. Most of the states, however, remain unchanged.

c) Will the value of the optimal policy change? Explain.

Solution:

Yes. The value of the optimal policy in a stochastic world will change because the agent will on average take longer to achieve the coffee. Since we have a negative reward for each state it lands at before getting to the goal, the expected values of this new policy will be smaller.

Part c (4pt)

Now consider a deterministic MDP, like in Part a. But this time, the agent has a meeting with their adviser in 5 minutes. So they need a policy that optimizes getting the coffee in that time limit. We will model this as an episodic MDP.

Consider the case where each step takes 1 minute to execute.

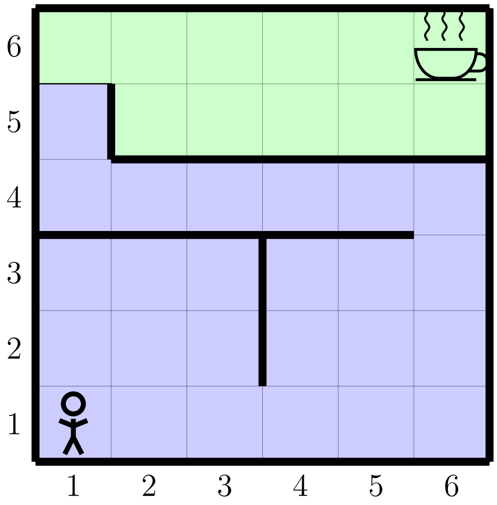

Figure 3: MDP for problem 1c.

a) Specify a reward function $R : S \times A \times S$ that will lead to the policy giving the shortest path to the coffee cup in states around the coffee cup. Try to keep the reward function as simple as possible.

Solution:

$R(6,6) \times A = 1$, for any action the agent takes at $S = (6,6)$ (also we know that the agent will not transition to any other state after arriving at $S = (6,6)$). We set all other rewards to $R = -1$ (or any other non-zero negative value), because the user wants to get to the coffee as fast as possible.

b) Refer to Figure 3. Using your reward function, Will the agent's policy in the green shaded region change compared to the MDP in Part a? Justify your answer.

Solution:

Policies in the green area will be equal to the ones in Part a

c) Refer to Figure 3. Using your reward function, consider policy π_a that in state $(1,5)$ chooses action down, and a policy π_b that in state $(1,5)$ chooses action up. How does $V_{\pi_a}((1,5))$ relate to $V_{\pi_b}((1,5))$?

Solution:

If we consider that our episodes contain only 5 steps, starting points in the blue shaded area will never be able to get to the coffee pot. Therefore, any actions taken inside the blue zone will lead to the same negative reward of value -5. If that is the case, then $V_{\pi a}((1,5)) = V_{\pi b}((1,5))$ because they will both not reach the coffee cup.

d) Consider a policy π_a and a policy π_b, where $\pi_a(s_{green}) = \pi_b(s_{green})$ and $\pi_a(s_{blue}) \neq \pi_b(s_{blue})$. How do V_{π_a} relate to V_{π_b}? Explain.

Solution:

As explained in the previous item, trajectories from initial states in the blue zone will never achieve the coffee pot. Therefore, their values will be -5 (because we have a reward of -1 for each time step that passes until the end of the 5 episodes). Also, for points inside the green zone both policies yield the same result. Knowing this, $V_{\pi a} = V_{\pi b}$, for both both in the green and blue zones.

Problem 2

In this problem you will program value iteration and policy iteration. You will use environments implementing the OpenAI Gym Environment API. For more information on the Gym and the API see: `https://gym.openai.com/`. We will be working with different versions of the FrozenLake environment [2].

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 versions, the up action will always move the agent up, the left will always move left, etc. In the stochastic versions, the up action will move up with probability 1/3, left with probability 1/3 and right with probability 1/3.

There are three different tile types: frozen, hole, and goal. When the agent lands on a frozen 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. We have provided you with two different maps.

States are represented as integers numbered from left to right, top to bottom starting at zero. So the upper left corner of the 4x4 map is state 0, and the bottom right corner of the 4x4 map is state 15.

You will implement value iteration and policy iteration using the provided environments. You may use either Python 2.7 or Python 3. Some function templates are provided for you to fill in. Specific coding instructions are provided in the source code files.

Note: Be careful 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.

Coding (30 pt)

Implement the functions in the code template. Then answer the questions below using your implementation.

Part a (20 pt)

Answer these questions for the maps `Deterministic-4x4-FrozenLake-v0` and `Deterministic-8x8-FrozenLake-v0`.

a) Using the environment, find the optimal policy using policy iteration. Record the time taken for execution, the number of policy improvement steps and the total number of

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

policy evaluation steps. Use $\gamma = 0.9$. Use a stopping tolerance of 10^{-3} for the policy evaluations.

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 4 for an example of the expected output.

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 5 for an example.

d) Find the optimal value function directly using value iteration. Record the time taken for execution, and the number of iterations required. Use $\gamma = 0.9$ Use a stopping tolerance of 10^{-3}.

e) Plot this value function as a color image, where each square shows its value as a color. See Figure 5 for an example.

f) Which algorithm was faster? Which took less iterations?

g) Are there any differences in the value function?

h) 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 4 for an example of the expected output.

i) 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.

```
LLLL
DDDD
UUUU
RRRR
```

Figure 4: Example policy for `FrozenLake-v0`

Part b (10 pt)

Answer the following questions for the maps `Stochastic-4x4-FrozenLake-v0` and `Stochastic-8x8-FrozenLake-v0`.

a) Using 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 $\gamma = 0.9$.

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

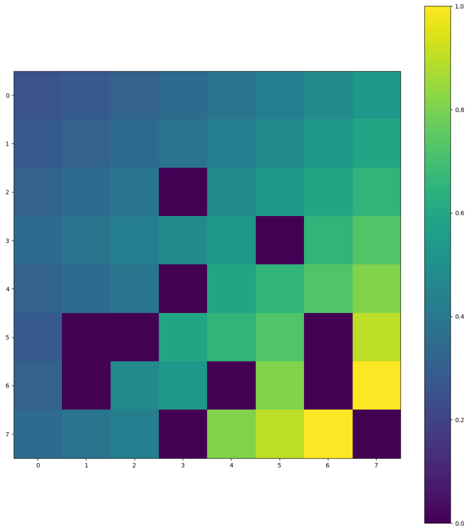

Figure 5: Example of value function color plot. Make sure you include the color bar or some kind of key.

c) Convert this value function to the optimal policy and include it in the writeup.

d) Does the 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.

e) 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.

Part c (4 pt)

We have provided one more version of the frozen lake environment. Now the agent receives a -1 reward for landing on a frozen tile, 0 reward for landing on a hole, and +1 for landing on the goal.

Answer these questions for map `Deterministic-4x4-neg-reward-FrozenLake-v0`.

a) Using value iteration, find the optimal value function. Use a stopping tolerance of 10^{-3}. Use $\gamma = 0.9$. Plot the value function as a color map like in Part a.

b) Is the value function different from the other deterministic 4x4 map?

c) Convert the value function to the optimal policy and include it in the writeup.

d) Is the policy different from the other deterministic 4x4 map? If so, pick a state where it differs and explain why the action is different.

Problem 3 (10pt)

In this problem you will practice implementing an environment. Given the following environment description implement an OpenAI Gym environment that matches the specifications.

You have a server which contains three queues. Each queue can contain up to five items. At every timestep the server is currently working on a specific queue. The server starts the environment on queue 1. The server has four actions: service an item from the current queue, switch to queue 1, switch to queue 2, switch to queue 3. Servicing an item from the queue when there is an item present gives a reward of +1. When the queue is empty no reward is given. Switching queues gives no rewards.

After each action each queue has a probability of receiving a new item: $P1$, $P2$, and $P3$ for queue 1, 2, and 3 respectively.

Implement this environment with the following sets of probabilities:

- $P1 = 0.1$, $P2 = 0.9$, $P3 = 0.1$
- $P1 = 0.1$, $P2 = 0.1$, $P3 = 0.1$

Problem 4 (10pt)

Consider an MDP $M = (S, A, P, R, \gamma)$, where the components of M are as described in Problem 1. In this problem, we will study some properties of value iteration. The Bellman optimality equation for the optimal value function $V^* : S \to \mathbb{R}$, which we also write $V^* \in \mathbb{R}^S$, is

$$V^*(s) = \max_{a \in A} \left(\sum_{s' \in S} P(s'|s,a) \left(R(s,a,s') + \gamma V^*(s') \right) \right).$$

Define the Bellman optimality operator $\mathcal{F}^* : \mathbb{R}^S \to \mathbb{R}^S$ as

$$\mathcal{F}^* V(s) = \max_{a \in A} \left(\sum_{s' \in S} P(s'|s,a) \left(R(s,a,s') + \gamma V(s') \right) \right),$$

where $\mathcal{F}^* V(s)$ is shorthand for $(\mathcal{F}^*(V))(s)$. Note that S is finite so value functions are essentially vectors in $\mathbb{R}^{|S|}$. The operator $\mathcal{F}^*$ maps vectors in $\mathbb{R}^{|S|}$ to vectors in $\mathbb{R}^{|S|}$.

Value iteration amounts to repeated application of $\mathcal{F}^*$ to an arbitrary initial value function $V_0 \in \mathbb{R}^S$.

a) Prove that V^* is an unique fixed point of $\mathcal{F}^*$, i.e, $\mathcal{F}^* V^* = V^*$ and that if $\mathcal{F}^* V = V$ and $\mathcal{F}^* V' = V'$ for two value functions $V, V' \in \mathbb{R}^S$, then $V = V'$, i.e., $V(s) = V'(s)$ for all $s \in S$.

Solution:

Since we know that

$$V^*(s) = \max_{a \in A} \left(\sum_{s' \in S} P(s'|s,a) \left(R(s,a,s') + \gamma V^*(s') \right) \right) \tag{1}$$

And we also know that:

$$\mathcal{F}^*V(s) = \max_{a\in A}\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V(s')\right)\right) \tag{2}$$

Then, by applying the F^* operator to V^* we have:

$$\mathcal{F}^*V^*(s) = \max_{a\in A}\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V^*(s')\right)\right) \triangleq V^* \tag{3}$$

Therefore, we proved that $\mathcal{F}^*V^* = V^*$

Now, for the second proof, if we have:

$$\mathcal{F}^*V(s) = \max_{a\in A}\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V(s')\right)\right) \tag{4}$$

Then by the definition of the optimal policy, $V = V^*$. Analogously, $V' = V^*$. Therefore, $V = V'$ because the optimal policy is unique.

b) Prove that $(\mathcal{F}^*)^k V_0$ converges to V^* as $k \to \infty$ for any $V_0 \in \mathbb{R}^S$. Consider convergence in max-norm. The max-norm of a vector $u \in \mathbb{R}^d$ is defined as $||u||_\infty = \max_{i\in\{1,\ldots,d\}} |u_i|$.

Solution:

To prove the convergence of $(\mathcal{F}^*)^k V_0$ to V^* as $k \to \infty$, we consider the maximum norm of the difference of both terms:

$$\lim_{k\to\infty} ||V_{k+1} - V^*||_\infty = ||(F^*)V_k - V^*||_\infty \tag{5}$$

First, let's look at the value of:

$$\begin{aligned}|(F^*)V_k - V^*| = |\max_{a\in A}\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V_k(s')\right)\right) \\ -\max_{a\in A}\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V^*(s')\right)\right)| \leqslant \max_{a\in A}|\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V_k(s')\right)\right) \\ -\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V^*(s')\right)\right)| = \max_{a\in A}\gamma\sum_{s'\in S} P(s'|s,a)|V_k(s') - V^*(s')|\end{aligned} \tag{6}$$

Continuing:

$$\begin{aligned}\max_{a\in A}\gamma\sum_{s'\in S}P(s'|s,a)|V_k(s')-V^*(s')|&\leqslant \max_{a\in A}\gamma\sum_{s'\in S}P(s'|s,a)||V_k(s')-V^*(s')||_\infty\\&=\gamma||V_k(s')-V^*(s')||_\infty\max_{a\in A}\sum_{s'\in S}P(s'|s,a)\end{aligned}\tag{7}$$

$= \gamma||V_k(s') - V^*(s')||_\infty$

Therefore, we have the following inequality:

$$||(F^*)V_k - V^*||_\infty\leqslant \gamma||V_k - V^*||_\infty \tag{8}$$

Recursively, if we apply (F^*) k times we have:

$$||(F^*)^k V_0 - V^*||_\infty\leqslant \gamma^{k+1}||V_0 - V^*||_\infty \tag{9}$$

As $k \to \infty$, $\gamma^{k+1} \to 0$, and we prove that $\lim_{k\to\infty}||(F^*)^{k+1}V_k - V^*||_\infty= 0$

c) Given the optimal value function V^* write down the expression that recovers the optimal policy π^* as a function of V^* and the parameters of M. [3]

Solution:

$$\pi^* = argmax_{a\in A}\sum_{s'} P^a_{s,s'}(R^a_{s,s'} + \gamma V^*(s')) \tag{10}$$

[3] Actually, such a procedure works for any value function $V \in \mathbb{R}^S$. This procedure is called policy extraction.