**Assignment 1 Reinforcement Learning (20p)**

**Goal**: To familiarize yourself with which you have learned for reinforcement learning.

**Due Date**: **December 7, 2020 at 08:00am**. Note that it is a **DEADLINE**. You will grade as zero if you return your writeup and code after the deadline unless you have good reasons and let me now in advance.

**Instructions**

* You work on this assignment by yourself.
* Writeups should be submitted as PDF.

**Problem 1 (30pt)**

Consider an environment in which our agent requires Guangtouqiang to function. Because Guangtouqiang 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 puerh tea. Once the agent reaches tea, 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×S×A* → [0,1], which maps (*s′,s,a*) to *P*(*s′|s,a*), i.e., the probability of transitioning to state *s′* ∈ *S* when taking action *a* ∈ *A* in state *s* ∈ *S*. Note that for all *s* ∈ *S*, *a* ∈ *A*.

*R*: The reward function *R : S × A × S → R*, which maps (*s,a,s′*) to *R*(*s,a,s′*), i.e., the reward obtained when taking action *a* ∈ *A* in state *s* ∈ *S* and arriving at state *s*′ ∈ *S*.

*γ*: The discount factor, which controls how important are rewards in the future. We have γ ∈ [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 tea 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 tea cup, the episode ends. Another way to think of this, is that every action in the teacup state, keeps the agent in the tea cup state.

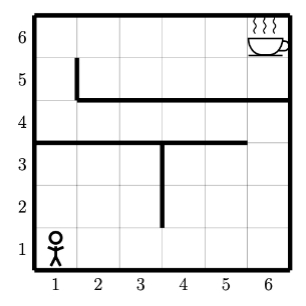


Figure 1: A particular instance of the shortest path problem. The goal is for the agent currently located in state (1, 1) to have a policy that always leads it on the shortest path to the tea in state (6, 6).

**Part a (20pt)**

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

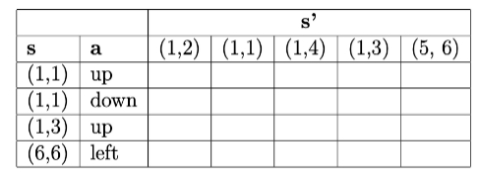
Using the above problem description answer the following questions:

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

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

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

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



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

f)  Does γ ∈ (0, 1) affect the optimal policy in this case? Explain why.

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

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.

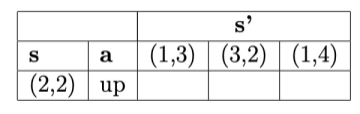
i)  Is your policy deterministic or stochastic?

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

**Part b (4pt)**

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 .



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

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

**Part c (6pt)**

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 tea in that time limit. We will model this as an episodic MDP.

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

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

b)  Refer to Figure 2. 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.

c)  Refer to Figure 2. Using your reward function, consider policy *πa* that in state (1,5) chooses action down, and a policy *πb* that in state (1,5) chooses action up. How does Vπa ((1, 5)) relate to Vπb ((1, 5))?

d)  Consider a policy *πa* and a policy *πb*, where *πa*(*sgreen*) = *πb*(*sgreen*) and *πa*(*sblue*)≠ *πb*(*sblue*). How do V*πa* relate to V*πb* ? Explain.

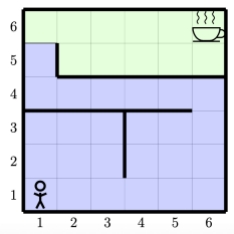


Figure 2: MDP for problem 1c.

**Problem 2 (50pt)**

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 (https://gym.openai.com/envs/FrozenLake-v0 ).

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 4*x*4 map is state 0, and the bottom right corner of the 4*x*4 map is state 15.

You will implement value iteration and policy iteration using the provided environments. You may use Python 3 or above. 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**

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

**Part a (30 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 policy evaluation steps. Use γ = 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 3 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 4 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 γ = 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 4 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 3 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.



Figure 3: Example policy for FrozenLake-v0

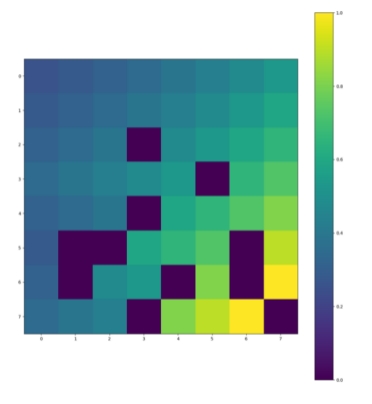


Figure 4: Example of value function color plot. Make sure you include the color bar or some kind of key.

**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 γ = 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?

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 (10 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 γ = 0.9. Plot the value function as a color map like in Part a.

b)  Is the value function different from the other deterministic 4*x*4 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 4*x*4 map? If so, pick a state where it differs and explain why the action is different.

**Problem 3 (15pt)**

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: *P*1, *P*2, and *P3* for queue 1, 2, and 3 respectively.

Implement this environment with the following sets of probabilities:

• *P*1=0.1, *P*2=0.9, *P*3=0.1  
• *P*1=0.1, *P*2=0.1, *P*3=0.1

**Feedback (5pt)**

* + - * Time you spend for this assignment, i.e., how many hours? (**1p**)
      * Comments for this course? (**2p**)
      * Comments for this assignment? (**1p**)
      * Suggestion for the following lectures? (**1p**)

**Writeups**

* Code for this assignment. The code should be based on the code template and using python language.
* The results and curves are suggested to include in this writeup.
* Feedback
* Your code should be bug free. You would get zero point if there are bugs and we can not run your code to get the results.

**Note:**

* If you have questions about the project, please contact with (Yalu Cheng ) as early as possible, two days or more before the deadline are preferred

**How to submit**

* Send an **email** (yalucheng@stu.pku.edu.cn), with the title AI\_homework\_1\_X\_Y,
  + where AI is the shortage for artificial intelligence;
  + X is your name in Chinese;
  + Y is your studentID;
  + We are sorry that we do not provide file server for you to upload your homework.
  + If you have any questions about this course or assignment, please use a separate email. Do not ask any questions in this assignment-return email.
* Pack your code AND writeup in one package, using the file name AI\_homework\_1\_X\_Y instead of AI or something like that.
  + Writeup is not suggested to put in the email.
  + You are suggested to pack writeup with your code in one package with the title: Writeup\_AI\_homework\_1\_X\_Y.
  + Writeup is suggested to be in PDF and in ENGLISH.
  + The default package might be in “.zip” format. If not, please show your format and how we can unpack it (i.e., software).
* Attach your package which includes the code and writeup you have created.
* TA email: (yalucheng@stu.pku.edu.cn)

**Honor Code**

* The honor code applies to all work turned in for this course.
* You must write and debug your own code.
* In particular, all code and documentation should be entirely your own work. You may consult with other students about high-level design strategies related to programming assignments, but you may not copy code or use the structure or organization of another student’s program.
* If you use any code or functions found from the internet, please tell us the reference link and how do you use it. ***Direct code copy from the internet would be considered violation of this policy***
* ***If we find there are two returned assignments same in large proportional code, both of the assignments would be considered violation of this policy***