DDA 4230: Reinforcement learning **CUHKSZ**

Assignment 2

TA: Sheng Xu

Due Date: Oct. 31st, 11:59 pm
Total points available: 100 pts.

Note: *The skeleton codes used in this assignment are all provided, and you can find them in the zip file.*

Problem 1: Value and Policy Iteration Computation [30 pts.]

Consider a simple MDP with 3 states s_1, s_2, s_3 and 2 actions a_1, a_2. The transition probabilities and expected rewards are shown in Figure 1 (e.g., The transition probability of $P(s_1, a_1, s_2) = 0.6$, and the reward of $R(s_1, a_1) = 8.0$). Assume discount factor $\gamma = 1$.

```
s1: {
  a1: ({s1: 0.2, s2: 0.6, s3: 0.2}, 8.0),
  a2: ({s1: 0.1, s2: 0.2, s3: 0.7}, 10.0)
},
s2: {
  a1: ({s1: 0.3, s2: 0.3, s3: 0.4}, 1.0),
  a2: ({s1: 0.5, s2: 0.3, s3: 0.2}, -1.0)
},
s3: {
  a1: ({s3: 1.0}, 0.0),
  a2: ({s3: 1.0}, 0.0)
}
```

Figure 1: A simple MDP example for Problem 1.

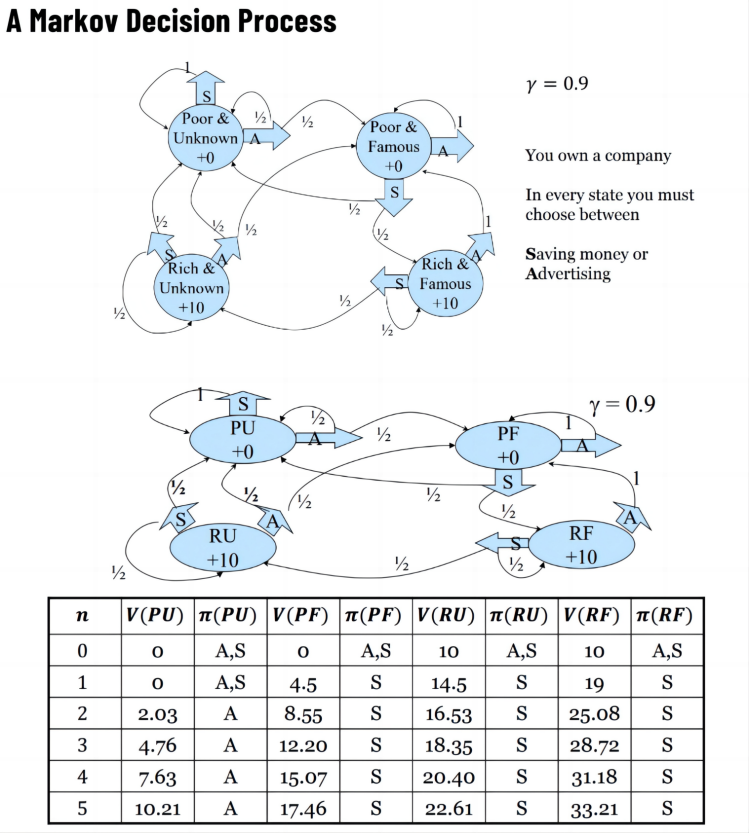
1. Initialize the value function for each state to be it's max (over actions) reward, i.e., we initialize the Value Function to be $V_0(s_1) = 10.0, V_0(s_2) = 1.0, V_0(s_3) = 0.0$. Then manually performs two iterations of value iterations. Show the final values for each state and the computation process. [15 points]

2. Let $\pi_k(s)$ denotes the extracted policy after k iterations of value iteration. Argue that $\pi_k(s) = \pi_2(s)$ for all states s. [15 points]

Problem 2: Value and Policy Iteration Implementation [40 pts.]

In this problem, you will program value iteration, policy iteration and modified policy iteration for Markov decision processes in Python. More specifically, fill in the functions in the skeleton code of the file *MDP.py*. The file *TestMDP.py* contains the simple MDP example as shown in Figure 2. You can verify that your code compiles properly with *TestMDP.py* by running "python *TestMDP.py*". Add print statements to this file to verify that the output of each function makes sense.

Note that you need to submit the following material:



n	$V(PU)$	$\pi(PU)$	$V(PF)$	$\pi(PF)$	$V(RU)$	$\pi(RU)$	$V(RF)$	$\pi(RF)$
0	0	A,S	0	A,S	10	A,S	10	A,S
1	0	A,S	4.5	S	14.5	S	19	S
2	2.03	A	8.55	S	16.53	S	25.08	S
3	4.76	A	12.20	S	18.35	S	28.72	S
4	7.63	A	15.07	S	20.40	S	31.18	S
5	10.21	A	17.46	S	22.61	S	33.21	S

Figure 2: A simple MDP example for Problem 2.

1. Your Python code, preferably one that is readable with some comments.
2. Test your code with the maze problem described in *TestMDPmaze.py*.
   (a) Report the policy, value function and number of iterations needed by value iteration when using a tolerance of 0.01 and starting from a value function set to 0 for all states. [10 point]
   (b) Report the policy, value function and number of iterations needed by policy iteration to find an optimal policy when starting from the policy that chooses action 0 in all states. [10 point]
   (c) Report the number of iterations needed by modified policy iteration to converge when varying the number of iterations in partial policy evaluation from 1 to 10. Use a tolerance of 0.01, start with the policy that chooses action 0 in all states and start with the value function that assigns 0 to all states. [10 point] Discuss the impact of the number of iterations in partial policy evaluation on the results and relate the results to value iteration and policy iteration. [10 point]

Problem 3: Q-learning [30 pts.]

In this problem, you will program the Q-learning algorithm in Python. More specifically, fill in the functions in the skeleton code of the file *RL.py*. This file requires the file *MDP.py* that you programmed for Problem 2 so make sure to include it in the same directory. The file *TestRL.py* contains a simple RL problem to test your functions (i.e. the output of each function will be printed to the screen). You can verify that your code compiles properly by running "python *TestRL.py*".

Note that you need to submit the following material:

1. Your Python code. Test your code with the maze problem described in *TestRLmaze.py*.
2. Graph 1 [10 point]: Produce a first graph where the x-axis indicates the episode (from 0 to 200) and the y-axis indicates the average (based on 100 trials) of the cumulative discounted rewards per episode (100 steps). The graph should contain 3 curves corresponding to the exploration probability epsilon=0.1, 0.3 and 0.5 (set temperature=0). The initial state is 0 and the initial Q-function is 0 for all state-action pairs.
3. Graph 2 [10 point]: Produce a second graph where the x-axis indicates the episode (from 0 to 200) and the y-axis indicates the average (based on 100 trials) of the cumulative discounted rewards per episode (100 steps). The graph should contain 3 curves corresponding to the Boltzmann exploration temperature=0, 10 and 20 (set epsilon=0). The initial state is 0 and the initial Q-function is 0 for all state-action pairs.
4. Discussion [10 point]: Explain the impact of the exploration probability epsilon and the Boltzmann temperature on the cumulative discounted rewards per episode earned during training as well as the resulting Q-values and policy.