CSC 498: Assignment 1

Matthew Zhang, Claas Voelcker, Liquan Wang, Prof. Animesh Garg

Released: Thurs 10/07/2021 – Due: Thurs 10/21/2021

Name, First Name: _____

Student number: _____

Total points: 60 + 20 bonus

To complete the exercise, you can use the tex template provided in the materials github. Insert your answers into the solution space below each question. In case you are unfamiliar with Latex, you may also submit handwritten solutions, but make sure they are clean and legible.

Submit the exercise before 23:59 pm on the due date on quercus. To submit, please bundle your completed exercise sheet, your jupyter notebook and any material for the bonus task into one zip file. Name the zip file `studentnumber_lastname_firstname.zip` and upload it on quercus.

Each student will have 3 grace days throughout the semester for late assignment submissions. Late submissions that exceed those grace days will lose 33% of their value for every late day beyond the allotted grace days. Late submissions that exceed three days of delay after the grace days have been used will unfortunately not be accepted. The official policy of the Registrar's Office at UTM regarding missed exams can be found here `https://www.utm.utoronto.ca/registrar/current-students/examinations`. If you have a compelling reason for missing the deadline, please contact the course staff as soon as possible to discuss hand in.

For assignment questions, please use Piazza and the office hours, but refrain from posting complete or partial solutions.

I Theoretical background

1. *Chain-walk MDP* (10)

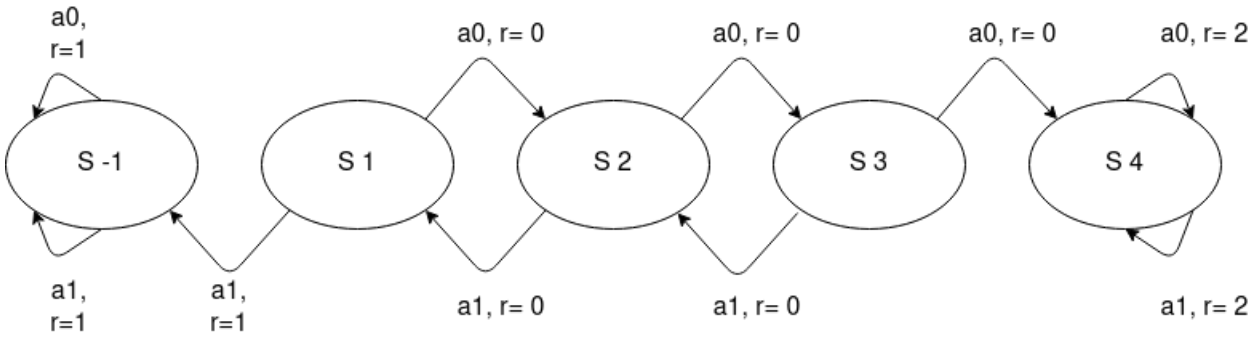

Figure 1: Simple Chain MDP

Consider a simple 5-state MDP shown in the Figure 1. The agent starts at state s_1, and has two actions available in each of the states s_i, with reward 0. Taking any action from state S_{-1} or S_4 results in a reward $r > 0$ and the agent stays in the state. The actions are deterministic and always succeed. Assume a discount factor $\gamma < 1$.

1. Compute the optimal values for each state with a discount of $\gamma = 0.9$. Show the equations and any simplifications so we can follow your reasoning. (3)
2. What are the optimal actions at states s_i given $\gamma = 0.9$? (3)
3. Does the optimal policy depend on the discount factor γ? What happens if it changes? Describe, in your own words, what the factor γ represents. (4)

2. *Runtime comparison* (10)

1. Compare the runtime of a single step of VI and PI. State your results in asymptotic notation, in terms of the size of the state space $\mathcal{S}$ and the size of the action space $\mathcal{A}$ (assume both are finite). (5)
2. Based on this, can you easily state whether PI or VI will be faster for any MDP? If yes, show the derivation, if no, argue why not. (5)

II Coding assignment

For the coding questions, provide your solutions by filling and uploading the jupyter notebook for the exercise. Make sure to only change code where we have marked the notebook with ??? and to only provide additional comments within the provided cells.

1. *Setup* (0)

To start, download all necessary code from github for assignment 1 from `https://github.com/pairlab/csc498-material`. Set up your Python environment and make sure you can run jupyter.

Run first section of the jupyter notebook assignment1.ipynb (This requires you to run all cells within Assignment 1, Task 1).

2. *Implement VI* Task 2 contains scaffolding code for a VI agent. Replace all blocks marked with ??? with your own code. You need to code the value update and policy extraction steps. (10)

3. *Implement Counting Estimation VI* Task 3 contains scaffolding code for two parts: the first will obtain some number N of transitions of the form (S_i, A_i, S_i') from the environment. For this part, you are not given the transition matrix explicitly. Instead, write code that estimates the transition matrix using the following estimator: (10)

$$\mathcal{T}(S, A, S') = \frac{\#(S_i = S, A_i = A, S_i' = S')}{\#(S_i = S, A_i = A)} \tag{1}$$

where $\#(S_i = S, A_i, S_i' = S')$ represents the number of samples where the three equalities hold, and $\#(S_i = S, A_i = A)$ the number of samples where only the two equalities hold.

4. *Runtime tests with Exact and Counting Estimation VI* Run your VI agent again, but using the counting estimator instead for the transitions. Compare and plot the performance of the counting estimator compared to the exact VI, as you sample $N = 100, 500, 1000, 2000, 4000$ transitions. What behaviour do we expect as $N \to \infty$? (10)

III Policy Evaluation

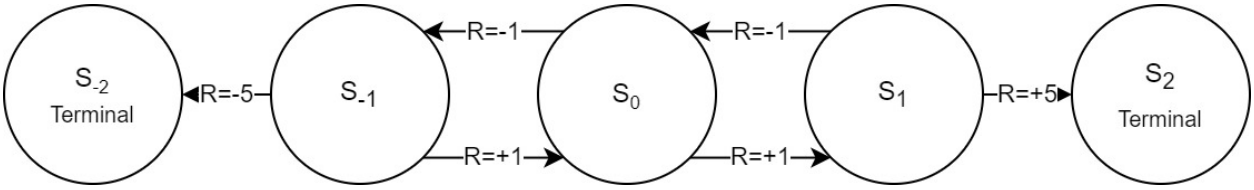

Figure 2: Chain-Walk MDP

Consider the chain-walk MDP, found in Figure 2.

Now suppose the agent currently estimates the value of each state to be 0. Then, the agent observes the following sequence of states and rewards:

1. *First-Visit Monte Carlo* (3)

Time	State	Action	Reward
1	S_0	A_1	-1
2	S_{-1}	A_1	$+1$
3	S_0	A_1	$+1$
4	S_1	A_1	$+5$
5	S_2	N/A	N/A

Table 1: Observed sequence of states, rewards and actions

What is the first visit Monte Carlo estimate of V at each state?

2. *Every-Visit Monte Carlo* (3)

   What is the every-visit Monte Carlo estimate of V at each state?

3. *Increment Monte Carlo* (4)

   What is the incremental Monte Carlo estimate of V at each state, if $\alpha = 1$?

IV Bonus challenge

The assignment will include a bonus question. These are meant as additional challenges for highly motivated students and require either prior knowledge or some independent learning. You will be able to get full points in all exercises without these questions, but we strongly encourage you to at least try to complete them. The bonus points will improve your final exercise score in the final grade calculations.

For each of the bonus questions, we will only provide minimal guidance and a high level task description. This means you are strongly encouraged to play around, think about different strategies and discuss your findings in your submission. Upload a description of your solution and relevant code alongside your submission.

1. *Model based pendulum swingup* (20)

   In the bonus task, you will tackle a more complex problem using value iteration and policy iteration. You will be using the OpenAI gym environment "Pendulum-v0".

   In the first step, you need to train a model of the environment using 50,000 samples. To obtain these, you should execute random actions in the environment and reset once the done signal is returned. You may use sklearn or torch for this, you do not need to implement your own ML model. The model should predict the next timestep and reward given the last observation.

   Next, you need to discretize the state and action space to use and policy iteration or value iteration approach. You are free to use any strategies here, there are no bounds

on your creativity (except your hardware limitations). We do suggest to start simple though.

Using your model and the discretization, produce a tabular MDP for the Pendulum swingup task and run your VI or PI agent. You may reuse code from the exercise for this step and it is a good idea to stick closely to the interfaces used in the previous task.

Finally, evaluate your agent using at least 16 independent runs of the original environment. Does the final reward align with the estimated value function of your agent? Are there failure cases and can you explain these? We expect the whole code to run in under 15 minutes.

To obtain full points, we expect clean code, a small written report containing a short discussion of your choice of ML model, your discretization scheme and a graph of reward over time steps showing the mean and standard deviation over all your runs. In addition, please add a small discussion of the final results. Please provide your code and all written parts together in form of a single jupyter notebook.