CS698R: Deep Reinforcement Learning **(Submission Deadline: 19/09/21, 1900 Hrs)**

Mid-Semester Exam

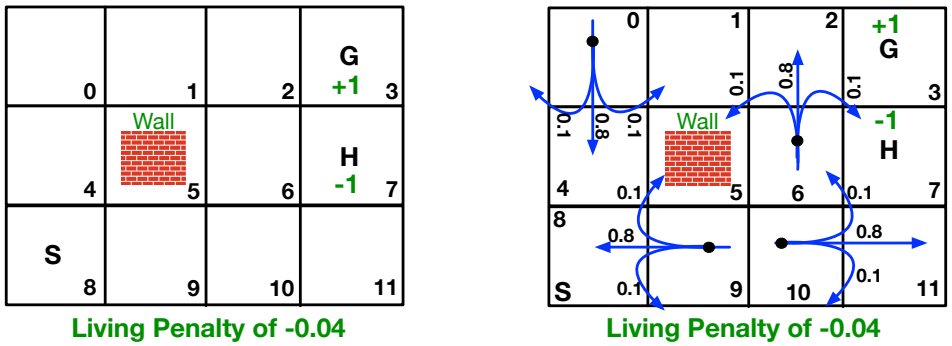
Instructor: Ashutosh Modi
Submission Link: https://forms.gle/Pv6WFo6ZVXvdReym6

Read all the instructions below carefully before you start working on the exam.

- This is a take home exam. You are given approx. 10 hours to solve the exam and submit your solutions.
- The exam requires your to implement some algorithms and you are required report your findings after experimenting with those algorithms.
- You are required to submit a report that must include the graphs/plots of the experiments you run and your findings.
- Solutions to the assignment need to be typeset in the LaTeX. Template for the report (Solutions-Midsem.tex file) is provided. DO NOT change the format of the template, use it as is provided to you without modifying it. In case you need any additional latex package, you can add it. You are required to compile the tex file and submit pdf version of the report. Do remember to fill your name and roll number in the solutions template.
- Implement the code in GitHub private repo and zip the repo and upload during submission. You need to follow the same directory structure as you did in the assignment else Gym environment would not work and we will not be able to reproduce your results. Include readme file with all details about how to run the code, etc. and have the code very well documented, there are marks for that.
- We would prefer if you write the code in Jupyter notebooks. The notebook should install the environment (via pip), using the following command in the first block:
  `!pip install -e relativePathTo/Environments`.
  Here `relativePathTo/Environments` is location of your environment relative to the current Jupyter notebook. Note due to some (known) problem with gym setup, aftrer you have run the above pip command in Jupyter, you need to restart the kernel. The notebook you develop should be able to run standalone without any need for any external file.
- There should be code that prints all plots and solutions in the notebook as well. Each problem requires creation of a separate Jupyter notebook. Read carefully the specifics in the question itself.
- You are expected to implement algorithms on your own and not copy it from other sources/class mates. Of course, you can refer to lecture slides. But no other external source other than the lecture slides or RL book by Sutton is allowed.
- There will be a plagiarism check, if your report is found to be plagiarized then you directly get zero marks, no questions asked.
- This is an individual exam and not a group exam, so please do not discuss your solutions with others, it will affect your grade only.
- In case your solution is found to have an overlap with solution by someone else (including external sources), all the parties involved will get zero in this and all future exams plus further more penalties in the overall grade. We will check not just for lexical but also semantic overlap. Same applies for the code as well. **Even an iota of cheating would NOT be tolerated.** If you cheat one line or cheat one page the penalty would be same.
- Be a smart agent, think long term, if you cheat we will discover it somehow, the price you would be paying is not worth it. Moreover, by letting others look at your solution, you will degrade your grade only and if you think in long term, it will affect your career only. So be a far-sighted agent!
- In addition to checking your code and report, very likely, we will be conducting one-on-one viva for the evaluation. So please make sure that you do not cheat!

- **In case something is not clear or you feel some information is missing, please make appropriate assumptions and specify your assumptions clearly in your report. Solve the problem using these assumptions. Since, this is an exam do not contact TAs and instructor or anyone else about clarification, make the best possible assumptions.**
- The deadline for the submission is given above. Submit at least 30 minutes before the deadline, lot can happen at the last moment, your internet can fail, there can be a power failure, you can be abducted by aliens, etc. Not even one second delay will be allowed.
- You have to submit your solution via following Google Form (link above)
- The form would close after the deadline and we will not accept any solution. No reason what-so-ever would be accepted for not being able to submit before the deadline. **DO NOT email your solutions, these will only go into spam.** Also, you can submit your solution only once, no revisions are allowed.
- Since the exam involves experimentation, reporting your results and observations, there is a lot of scope for creativity and innovation and presenting new perspectives. Such efforts would be highly appreciated and accordingly well rewarded. Be an exploratory agent!
- In your plots, have a clear legend and clear lines, etc. **Make sure the legends are not overlapping or shadowing the plot, make the legend on the side or above the plot.** Of course you would be generating the plots in your code but you must also put these plots in your report. Generate high resolution pdf/svg version of the plots so that it doesn't pixilate on zooming.
- For implementing a new environment in OpenAI gym, you can refer to this tutorial: https://github.com/openai/gym/blob/master/docs/creating-environments.md. Also check out the `env` class: https://github.com/openai/gym/blob/master/gym/core.py. You are not required to render the environment on the screen/terminal. Do remember to set the seed, this will be useful for reproducing your experiments. And we will use the seed provided by you to verify your results. Also for each instance of the environment that you create use a different seed and save these seeds, these will be useful for reproducing the results. Do not set the seed to 42 :P
- For implementing the environment use the same directory structure as explained by Samik and also previously sent over email.
- For all experiments, report about the seed used in the code documentation and also in your report, write about the seed used.
- In your report write about all things that are not obvious from the code e.g., if you have made any assumptions, references/sources, running time, etc.

Random-Maze Environment



(a) Maze Environment (b) Maze Environment Transitions

In this exam we will be exploring a variant of the Random Maze Environment (RME) that we have been looking in the lectures. The environment is represented as a grid world in Figure 1a. Random maze environment is a highly stochastic environment with 11 states: two terminal states (a goal state (G) and a hole state (H)) and 9 non-terminal states and a wall in between the environment. The wall behaves similar to the wall on the periphery of the environment, basically if an agent bumps against the wall, it bounces back. The boundary of

the environment behaves similarly, if an agent hits the boundary it bounces back. The agent receives a reward of +1 when it lands in the goal state (3) and it receives a reward of -1 when it lands in the hole state (7). For rest of the transitions there is a reward of -0.04. Essentially the agent has the living cost of -0.04. The transitions are stochastic as shown in Figure 1b. In this environment, four actions are possible: left, top, right, and bottom. For every intended action, there is 80% chance of going in the intended direction and remaining 20% chances of going in either of the orthogonal directions. The 20% chance gets equally distributed between each of the orthogonal direction. The agent starts from state 8 (S). Assume $\gamma = 0.99$ for the problems below.

Problem 1: Random-Maze Environment Implementation
(30 points)

In OpenAI-Gym create an environment for Random Maze described above. Check if the environment works well by executing few test cases. Implement the test cases in the code and describe the test cases in your report as well.

Problem 2: RME Optimal Policy via Dynamic Programming
(40+40+10=50 points)

Assume we know the underlying MDP of the RME. We are interested in finding out the optimal policy for this environment. For each of the part below, keep your code modular, write a separate function for each component in the algorithm. There are marks for how well organized is your code and how well it is documented. Create a separate Jupyter notebook for this problem.

1. Start with any random policy of your choice and use Policy Iteration (PI) algorithm to find the optimal policy. Assume the threshold $\theta = 10^{-10}$. You are required to implement the code for finding the optimal policy via PI algorithm. Clearly describe the random policy initially chosen both in code and as well as in report. In the report, also draw the random policy chosen with the help arrows as done in the lectures. You can draw with hand on paper and insert that picture in the report. But please draw very neatly and clearly. The code should be very well documented. In the report, draw the final optimal policy that is obtained via PI. Write about how many iterations did you require to converge to optimal policy?

2. Start with any random policy of your choice and use Value Iteration (VI) algorithm to find the optimal policy. Assume the threshold $\theta = 10^{-10}$. You are required to implement the code for finding the optimal policy via VI algorithm. Clearly describe the random policy initially chosen both in code and as well as in report. In the report, also draw the random policy chosen with the help arrows as done in the lectures. You can draw with hand on paper and insert that picture in the report. But please draw very neatly and clearly. The code should be very well documented. In the report, draw the final optimal policy that is obtained via VI. Write about how many iterations did you require to converge to optimal policy?

3. Compare PI and VI, e.g., which shows faster convergence, if the optimal policy obtained via PI and VI is same. In general, will the optimal policy obtained via PI and VI will always be same? Describe your reason in the report, restrict your answer to maximum 10 lines.

Problem 3: RME Prediction with MDP Unknown
(5+5+5+5+40+40+10+5+5+5+10+10+10+5+5+5+10+10+10+5+5+5+5=220 points)

Now we assume that we do not know the underlying MDP and we are interested in solving the Prediction problem. For this part, pick up an optimal policy obtained in the previous problem. Create a separate Jupyter notebook for this problem.
As you might have noticed in Assignment 1, the MC and TD plots may have lot of variance and look very noisy. One way to overcome this is to create several different instances of the environment using different seeds and then average out the results across these and plot these. For the plots of MC-FVMC, MC-EVMC,TD, n-Step TD and TD(λ) mentioned below use this averaged out version. This will give you smoother plots. Record your seeds and report these along with the plots.

1. Implement a function that would simulate and generate a trajectory for RME for a given policy π and maximum number of steps. The function definition would be like this:

   `def generateTrajectory(env, π, maxSteps)`

   The function returns a list of experience tuples. Here, `maxSteps` parameter is used to terminate the episode if it exceeds maxSteps count. In such a case, the partial trajectory is discarded and and empty list is returned. Test the function using suitable test cases and make sure it is working.

2. Implement a function that would decay the step size parameter (α). The function definition would be like this:

   `def decayAlpha(initialValue, finalValue, maxSteps, decayType)`

   Here `decayType` can be linear or exponential. `maxSteps` is the maximum number of steps the step parameter should decay for. `initialValue` and `finalValue` are initial and final values of the step size parameter. The function should return a list of step size parameter values. Test the function by trying out different parameter settings. Plot value of α vs time step both for linear and exponential decays.

3. We would like to solve the prediction problem using `MonteCarloPrediction` algorithm. Implement the `MonteCarloPrediction` algorithm. Make use of the functions implemented in above two parts. Note `MonteCarloPrediction` should work for both FVMC and EVMC settings. Test the algorithm for RME using some pre-defined test cases and see the algorithm produces the desired results. Report your test cases and observations.

4. We would like to solve the prediction problem using `TemporalDifferencePrediction` algorithm. Implement the `TemporalDifferencePrediction` algorithm. Test the algorithm for RME using some pre-defined test cases and see the algorithm produces the desired results. Report your test cases and observations.

5. Now solve the prediction problem using `n-Step TD Learning` algorithm. Implement the `n-Step TD Learning` algorithm. Test the algorithm for RME using some pre-defined test cases and see the algorithm produces the desired results. Report your test cases and observations.

6. Finally, we will solve the prediction problem using `TD`(λ) algorithm (Backward-view). Implement the `TD`(λ) algorithm. Test the algorithm for RME using some pre-defined test cases and see the algorithm produces the desired results. Report your test cases and observations.

7. Calculate the true value of state for each state. Implement it in the code and show the full derivation in the report.

8. Plot the MC-FVMC estimate of each non-terminal state of RME as it progress through different episodes. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior.

9. Plot the MC-EVMC estimate of each non-terminal state of RME as it progress through different episodes. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior. How does EVMC fair against FVMC?

10. Plot the TD estimate of each non-terminal state of RME as it progress through different episodes. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior.

11. Plot the n-Step TD estimate of each non-terminal state of RME as it progress through different episodes. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant

α. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior.

12. Plot TD(λ) estimate of each non-terminal state of RME as it progress through different episodes. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior.

13. For episode number 100, in the TD(λ) algorithm, plot the progress of the eligibility trace vs time step for each non-terminal state. Plot eligibility trace for each non-terminal state in the same plot.

14. Plot the MC-FVMC estimate of each non-terminal state of RME as it progress through different episodes. But this time, the x-axis (episodes) should be log-scale. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. This plot will help to zoom in and observe the behavior of the estimates in the initial stages. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior.

15. Plot the MC-EVMC estimate of each non-terminal state of RME as it progress through different episodes. But this time, the x-axis (episodes) should be log-scale. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. This plot will help to zoom in and observe the behavior of the estimates in the initial stages. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior. How does EVMC fair against FVMC?

16. Plot the TD estimate of each non-terminal state of RME as it progress through different episodes. But this time, the x-axis (episodes) should be log-scale. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. This plot will help to zoom in and observe the behavior of the estimates in the initial stages. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior.

17. Plot the n-Step TD estimate of each non-terminal state of RME as it progress through different episodes. But this time, the x-axis (episodes) should be log-scale. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. This plot will help to zoom in and observe the behavior of the estimates in the initial stages. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior.

18. Plot the TD(λ) estimate of each non-terminal state of RME as it progress through different episodes. But this time, the x-axis (episodes) should be log-scale. In the same plot also plot the true estimate. Take maximum of 500 episodes. You can play with different settings of α, for example, the step size parameter (α) starts from 0.5 and decreases exponentially to 0.01 till 250 episodes and after that it is constant. Or else you can also try with small (< 1) value of constant α. This plot will help to zoom in and observe the behavior of the estimates in the initial stages. Analyze the plots for each state and report your observations, findings and possible reasons for the observed behavior.

19. Based on the plots, compare MC-FVMC, MC-EVMC,TD, n-Step TD and TD(λ) approaches and report your observations. Also, in a single plot, take any interesting (based on plots above) one non-terminal state, plot the progress of estimates of MC-FVMC, MC-EVMC,TD, n-Step TD and TD(λ) vs episodes to compare the algorithms. What do you observe, report your observations.

20. Plot the MC-FVMC Target value (G_t) for any one non-terminal state of RME as it progress through different episodes. Use the same setting as above. In the same plot also include the optimal value of the

state. The plot will be similar to discussed in slide 78 of lecture 8. What do you observe and what are the reasons for what you observe? Explain and Report.

21. Plot the MC-EVMC Target value (G_t) for any one non-terminal state of RME (use the same state as above) as it progress through different episodes. Use the same setting as above. In the same plot also include the optimal value of the state. What do you observe and what are the reasons for what you observe? Explain and Report.

22. Plot the TD Target value (G_t estimate) for any one non-terminal state of RME (use the same state as above) as it progress through different episodes. Use the same setting as above. In the same plot also include the optimal value of the state. The plot will be similar to discussed in slide 79 of lecture 8. What do you observe and what are the reasons for what you observe? Explain and Report.

23. Based on the plots, compare MC-FVMC, MC-EVMC and TD targets and report your observations.