**CS780: Deep Reinforcement Learning** **(Submission Deadline: 07/03/24, 09:59 AM)**

# Assignment #2

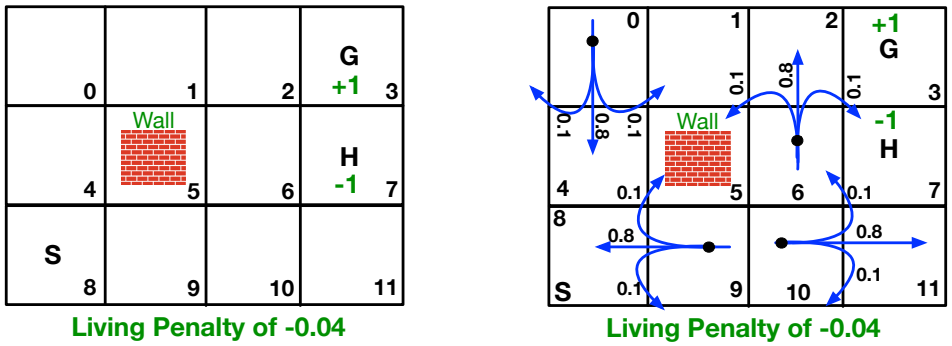
**Instructor**: Ashutosh Modi
**Submission Link**: https://forms.gle/LiXSz9sf4oaCv14T9

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

- The purpose of this course is that you learn RL and the best way to do that is by implementation and experimentation.
- The assignment requires your to implement some algorithms and you are required report your findings after experimenting with those algorithms. The report should be submitted in pdf format.
- Implement the code in Google Colab Notebooks and include links to your notebook in your report. Provide a shorter tinyurl link instead of the entire link. The code should be very well documented. There are marks for that. Also, you need to download the notebook (zip it) and submit it.
- In case you use any maths in your explanations, render it using latex in the notebook.
- 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.
- If you use any reference or material (including code), please cite the source, else it will be considered plagiarism. But referring to other sources that directly solve the problems given in the assignment is not allowed. There is a limit to which you can refer to outside material.
- This is an individual assignment.
- 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 assignments 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.
- In case you are struggling with the assignment, seek help from TAs. Cheating is not an option! I respect honesty and would be lenient if you are not able to solve some questions due to difficulty in understanding. Remember we are there to help you out, seek help if something is difficult to understand.
- 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.
- You have to submit your assignment 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.
- Since the assignment 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!
- Your code should be very well documented, there are marks for that.
- In your plots, have a clear legend and clear lines, etc. Of course you would generating the plots in your code but you must also put these plots in your notebook. Generate high resolution pdf/svg version of the plots so that it doesn't pixilate on zooming.

- For all experiments, report about the seed used in the code documentation, write about the seed used.
- In your notebook write about all things that are not obvious from the code e.g., if you have made any assumptions, references/sources, running time, etc.
- **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!**
- **Use of LLMs based tools or AI-based code tools is strictly prohibited! Use of ChatGPT, VS Code, Gemini, CO-Pilot, etc. is not allowed. NOTE VS code is also not allowed. Even in Colab disable the AI assistant. If you use it, we will know it very easily. Use of any of the tools would be counted as cheating and would be given a ZERO, with no questions asked.**

**Random-Maze Environment**




(a) Maze Environment (b) Maze Environment Transitions

In this assignment 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.

In this assignment we will be looking at control algorithms we learnt in Lectures. For each of the plot, **create the legend on the left/right side so that it doesn't overlay on the plot.** For all the algorithms below, this time we will not be specifying the hyper-parameters, please play with the hyper-params to come up with the best values. This way you will learn to tune the model. As you are aware from your past experience, single run of the algorithm over the environment results in plots that 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 all the plots below, you this strategy.

**Problem 1: Monte Carlo Control**
(40+20+20+5+5+5+5=100 points)

Implement the Monte Carlo Control for the Random Maze Environment (RME) described above. In particular, you need to implement First Visit Monte Carlo Control (FMVCC) for finding the optimal policy for RME. Use the function definition (given below) as given in Lecture slides.

`MonteCarloControl(env,` $\gamma$`,` $\alpha_0$`,` $\epsilon_0$`, maxSteps, noEpisodes, firstVisit = True)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 2: SARSA (TD Control)**
(40+20+20+5+5+5+5=100 points)

Implement the SARSA algorithm for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`SARSA(env,` $\gamma$`,` $\alpha_0$`,` $\epsilon_0$`, noEpisodes)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 3: Q-Learning**
(40+20+20+5+5+5+5=100 points)

Implement the Q-Learning algorithm for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`Q-Learning(env,` $\gamma$`,` $\alpha_0$`,` $\epsilon_0$`, noEpisodes)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 4: Double Q-Learning**

(40+20+20+5+5+5+5=100 points)

Implement the Double Q-Learning algorithm for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`Double-Q-Learning(env,` $\gamma$`,` $\alpha_0$`,` $\epsilon_0$`, noEpisodes)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 5: Comparing Control Algorithms**

(20+5+5+5+5+5=40 points)

For FVMCC, SARSA, Q and Double-Q algorithms implemented above, do the following:

(a) For each of the algorithm, in a single plot, plot the evolution of Policy Success Rate (in %) vs Episodes. Policy Success Rate is defined as number of times the agent reaches the goal state out of the total number of the episodes run using a specific policy. Basically implement the following function that would return the policy success percentage. As you are training the agent, at each episode, you will have a version of the policy, use that policy along with the function below to get the policy success rate.

`def getPolicySuccessRate(env,` $\pi_{current}$`, goalState, maxEpisodes=100, maxSteps=200)`

(b) What are your observations from the Policy Success Rate (in %) plot.

(c) For each of the algorithm (in a single plot), plot the Estimated Expected Return (from the start state) vs Episodes.

(d) What are your observations for the Estimated Expected Return plot?

(e) For each of the algorithm (in a single plot), plot the State-value Function Estimation Error vs Episodes. State-value Function Estimation Error is defined as Mean Absolute Error across all V-function estimates (across all states) from the respective optimal value.

(f) What are your observations for the State-value Function Estimation Error plot?

**Problem 6: SARSA($\lambda$) Replacing**

(40+20+20+5+5+5+5=100 points)

Implement the SARSA($\lambda$) algorithm with Replacing Eligibility Traces for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`SARSA-Lambda(env, `$\gamma$`, `$\alpha_0$`, `$\epsilon_0$`, `$\lambda$`, noEpisodes, replaceTrace = True)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 7: SARSA($\lambda$) Accumulating**

(40+20+20+5+5+5+5=100 points)

Implement the SARSA($\lambda$) algorithm with Accumulating Eligibility Traces for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`SARSA-Lambda(env, `$\gamma$`, `$\alpha_0$`, `$\epsilon_0$`, `$\lambda$`, noEpisodes, replaceTrace = False)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 8: Q($\lambda$) Replacing**

(40+20+20+5+5+5+5=100 points)

Implement the Q($\lambda$) algorithm with Replacing Eligibility Traces for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`Q-Lambda(env, `$\gamma$`, `$\alpha_0$`, `$\epsilon_0$`, `$\lambda$`, noEpisodes, replaceTrace = True)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 9: Q($\lambda$) Accumulating**

(40+20+20+5+5+5+5=100 points)

Implement the Q($\lambda$) algorithm with Accumulating Eligibility Traces for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`Q-Lambda(env,` $\gamma$`,` $\alpha_0$`,` $\epsilon_0$`,` $\lambda$`, noEpisodes, replaceTrace = False)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 10: Dyna-Q**

(40+20+20+5+5+5+5=100 points)

Implement the Dyna-Q algorithm for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`Dyna-Q(env,` $\gamma$`,` $\alpha_0$`,` $\epsilon_0$`, noEpisodes, noPlanning)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 11: Trajectory Learning**
(40+20+20+5+5+5+5=100 points)

Implement the Trajectory Learning algorithm for the Random Maze Environment (RME) described above. Use the function definition as given in Lecture slides.

`TrajectorySampling(env,` $\gamma$`,` $\alpha_0$`,` $\epsilon_0$`, noEpisodes, maxTrajectory)`

(a) Plot evolution of State-value (V) function with time. Basically, plot V-function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true V-function for each of the state in the same plot.

(b) Plot evolution of action state value (Q) function with time. Plot Q function vs Episodes for states 0, 1, 2, 4, 6, 8, 9, 10, 11 in a single plot. Also plot the true Q-function for each of the state in the same plot.

(c) Describe over how many instances of the environments did you average the results? Write about the seeds used for each instance.

(d) Draw the environment diagram with optimal policy (shown using arrows) obtained using the algorithm.

(e) Write about the hyper-parameters you finally used for the algorithm and describe how did you arrive at these set of hyper-params.

(f) Write about your observations from the plots above.

**Problem 12: Comparing Control Algorithms**
(5+5+5+5+5+5=25 points)

For SARSA($\lambda$) Replacing, SARSA($\lambda$) Accumulating, Q($\lambda$) Replacing, Q($\lambda$) Accumulating, Dyna-Q, Trajectory Learning implemented above, do the following:

(a) For each of the algorithm, in a single plot, plot the evolution of Policy Success Rate (in %) vs Episodes.

(b) What are your observations from the Policy Success Rate (in %) plot.

(c) For each of the algorithm (in a single plot), plot the Estimated Expected Return (from the start state) vs Episodes.

(d) What are your observations for the Estimated Expected Return plot?

(e) For each of the algorithm (in a single plot), plot the State-value Function Estimation Error vs Episodes. State-value Function Estimation Error is defined as Mean Absolute Error across all V-function estimates (across all states) from the respective optimal value.

(f) What are your observations for the State-value Function Estimation Error plot?