CP8319: Reinforcement Learning (Winter 2022)
Assignment 2
Due: Monday March 14, 2022 (Midnight 11:59PM or 23:59)
Instructor: Nariman Farsad

Please Read Carefully:

- Submit your assignments on D2L before the deadline. The submission will be closed after the deadline.
- Show your work and write your solutions legibly. You can use applications such as Word or Latex to type up your solutions for the written portion of the assignment or use applications such as CamScanner to create a PDF file from your handwritten solutions. **For coding problems you need to submit your python source files or Jupyter notebook, and have the main plots, answers to questions, or general observations in the written portion.**
- Do the assignment on your your own, i.e., do not copy. If two assignments are found to be copies of each other, they BOTH receive 0.
- These questions require thought, but do not require long answers. Please be as concise as possible.
- If you have a question about this homework, we encourage you to post your question on our D2L discussion board.
- For instructions on setting up the python environment for the assignment read the "README.md" file.

1. (25 points) **Monte Carlo GLIE**

   In this question, you will implement first-visit Monte Carlo control for the Frozen Lake environment from OpenAI Gym (click on OpenAIGym for details). In the first assignment, we knew the dynamics of the system. In this second assignment, we learn the optimal policy while acting in the environment. Please note that we have provided a custom version of this environment in the starter code, which is a different setup compared to the first assignment. Make sure you read the comments provided with each function as there are helpful hints in there.

   (a) **(coding)** (5 pts) Read through `mc_td_and_qlearning.py`. Specifically, please take a look at the `sample_action`, `take_one_step` functions and try and understand what they do. Then use these function to implement the `generate_episode` and `generate_returns`. For this part you don't have to include anything in the written portion.

   (b) **(coding)** (5 pts) Implement `mc_policy_evaluation` in`mc_td_and_qlearning.py`, where given a policy, one random episode is generated according to the policy. Then Monte Carlo method is used for policy Q evaluation. For this part you don't have to include anything in the written portion.

   (c) **(coding/written)** (10 pts) Implement `epsilon_greedy_policy_improve` and `mc_glie`. Please note that for the ϵ-greedy policy, if 2 or more actions all have the same maximum Q-value, they should have the same probability of being executed. You can see stackoverflow on how to do this efficiently. Run the Monte Carlo GLIE algorithm on the Deterministic-4x4-FrozenLake-v0. Use 1000 iterations and $\gamma = 0.9$. Report the accuracy of the policy in reaching the goal.

   (d) **(coding/written)** (5 pts) Run the Monte Carlo GLIE algorithm on the Stochastic-4x4-FrozenLake-v0. Use 1000 iterations and $\gamma = 0.9$. You will observe that the you will not get good results. Try increasing the iterations to 10,000 and report the results. The policy will still fail most of the times. Can you explain why? Please do not spend time on trying to improve the results.

2. (35 points) **TD SARSA and Q-Learning**

   In this question, we continue working on the Frozen Lake environment and implement the temporal-difference SARSA and Q-Learning algorithms for learning the optimal policy.

   (a) **(coding/written)** (10 pts) Implement `td_sarsa`. Run the temporal-difference SARSA algorithm on the Deterministic-4x4-FrozenLake-v0. Use 1000 iterations, $\alpha = 0.1$, and $\gamma = 0.9$. Report the accuracy of the policy in reaching the goal over deterministic environment. Run the temporal-difference SARSA algorithm on the Stochastic-4x4-FrozenLake-v0. Use 1000 iterations, $\alpha = 0.1$, and $\gamma = 0.9$. You will observe that the you will not get good results. Try increasing the iterations to 10,000 and report the results. The policy will still fail most of the times. Can you explain why? Please do not spend time on trying to improve the results on the stochastic environment.

   (b) **(coding/written)** (10 pts) Implement `qlearning`. Run the Q-Learning algorithm on the Deterministic-4x4-FrozenLake-v0. Use 1000 iterations, $\alpha = 0.1$, and $\gamma = 0.9$. Report the accuracy of the policy in reaching the goal over the deterministic environment. Run the Q-Learning algorithm on the Stochastic-4x4-FrozenLake-v0. Use 1000 iterations, $\alpha = 0.1$, and $\gamma = 0.9$. You will observe that the you will not get good results. Try increasing the iterations to 10,000 and report the results. The policy will still fail most of the times. Can you explain why? Please do not spend time on trying to improve the results on the stochastic environment.

(c) **(coding/written)** (15 pts) Assume that you can manipulate the learning parameters for temporal-difference SARSA algorithm and the Q-Learning algorithm, as well as the rewards that the environment outputs. Note that if you choose to modify the rewards, you need to do that manually in the functions `td_sarsa` and `qlearning` (i.e., there is no need to change the OpenAI source code for that). Can you devise a technique for improving the performance on the Stochastic-4x4-FrozenLake-v0? **You are not allowed to use more than 100,000 iterations**. Please describe the methods you chose and the accuracy you are able to achieve **on average over 10 training sessions** on the Stochastic-4x4-FrozenLake-v0 environment. Also report the variance over the 10 training sessions. If you observe any improvements, describe why the method you chose improved the results.