Sequential Decision-Making	**F2020**
Problem Set 2	
Handed Out: Oct 5^{th}*,* 2020	*Due: Oct* 23^{rd}*,* 2020

Except from Stopping by Woods on a Snowy Evening

The little horse must think it queer
To stop without a farmhouse near
Between the woods and frozen lake
The darkest evening of the year.

Robert Frost (1874–1963)

In this assignment you will implement various techniques for both model-based and model free reinforcement learning for the purposes solving the *frozen lake* environment provided via the Open AI Gym. As Gym provides pretty much all methods necessary for interfacing with the environment, no starter code has been provided (although we will go over some examples of how to interact with the environment in class). You will submit a writeup that summarizes your results and all code as a zip file. The writeup should be in AAAI format and, generally, be no longer than 3 to 4 pages. Your writeups can be longer, this is just meant to be a rough guideline. You can find the AAAI style files on Canvas. Submit the writeup (with attached source code) to the Canvas submission locker before 11:59pm on the due date.

The frozen lake environment is one that I would consider a "classic" environment. The environment, broadly described, is a grid world in which a tile can either be frozen (denoted by an F), the start state (denoted by an S, still frozen though), a hole (denoted by an H), or the goal (denoted by a G). There are 4 actions available to you: up, down left, and right. You get a reward of 1 for reaching the goal, and zero otherwise. This is a very sparse reward.

The big twist in this environment is that frozen spaces (and the start state) are considered *slippery*. This means that there is a good chance your actions will not do what you want them to do. In fact, there is only a 33% chance that you will do what you want. There is also a 33% that you will move 90 degrees clockwise or counterclockwise. Gym provides 2 versions of the frozen lake environment. For the first two parts of the assignment we will be working on the 4x4 version of the frozen lake environment. The final (potentially optional) part of the assignment will utilize the 8x8 version. To use these environments, you must first *import gym* and then initialize the environments by calling gym.make('FrozenLake-v0') for the 4x4 environment or gym.make('FrozenLake8x8-v0') for the 8x8 environment.

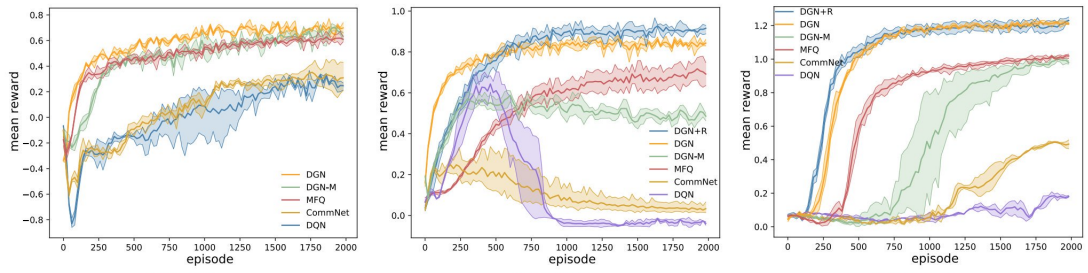
Tabular Model-Based RL (50 points)

So you know the environment, now we just have to solve it. Easy, right? Your first task is to implement our favorite model-based RL technique, Dyna-Q! In this part of the assignment you will use Dyna-Q to learn an optimal policy and then evaluate said policy in terms of its sample complexity. For this part of the assignment, use a table (or equivalent data structure) to keep track of your transition and reward models.

For this assignment, should create graphs that outline the performance of your methods. The general way that this is done for RL is to graph the average cumulative reward gained in an episode versus the amount of time that your agents have been learning (see Figure 1). Normally to do this, you would run your learning algorithm for an episode and then test the agent for some number of episodes (10 or so) and record the average reward obtained over those test episodes. Remember that when testing your agent you need to turn off any exploration policy that you may be using. The test agent should always greedily follow its policy. You can use whatever means you would like to create these graphs.

Details:

- I am sure there is a library somewhere that has this approach already implemented. Do not use them. Pretty much anything else should be fine. If you are not sure if a certain library is allowable, just ask and I will let you know.
- As mentioned at the start of class, all code should be written in Python 3.
- You are free to assign any parameters associated with Dyna-Q as you see fit.
- Explain any implementation choices you had to make in the final report. This includes the motivation behind using certain data structures or using certain libraries.
- Include a report of the learning rate (sample complexity) for Dyna-Q.



(a) jungle (b) battle (c) routing

Figure 4: Learning curves in terms of mean reward in jungle, battle, and routing.

Figure 1: Examples of reward graphs.

Tabular Model-Free RL (30 points)

So we've seen what a model-based approach can do, what about model-free approaches? In this part of the assignment you will solve the 4x4 frozen lake environment using both Q-Learning and SARSA. As before, you are free to set any parameters associated with these methods yoursel.f In fact, I encourage you to explore many different configurations to see the effect they have on learning. Also as in the previous section, please use a table or equivalent structure to keep track of state-action values. As always, please create graphs that show the learning rate of both Q-Learning and SARSA on this problem. In your writeup, please discuss any differences in learning rates you find between Q-Learning, SARSA, and Dyna-Q. Why might this be happening?

Details:

- As with Dyna-Q, please do not just find a library for Q-Learning or SARSA and use it for this project.
- As mentioned at the start of class, all code should be written in Python 3.
- Explain any implementation choices you had to make in the final report. This includes the motivation behind using certain data structures or using certain libraries.
- Report the learning rates for both SARSA and Q-Learning agents and a discussion of any differences in learning rates that you observe.

Presentation (20 points)

Your report must be complete and clear. A few key points to remember:

- AAAI Format: Your report must be written in AAAI format using either Word or LaTeX. I have included templates for each of these in the files for the course.

- Complete: the report does not need to be long, but should include everything that was requested.
- Clear: your grammar should be correct, your graphics should be clearly labeled and easy to read.
- Concise: I sometimes print out reports to ease grading, don't make figures larger than they need to be. Graphics and text should be large enough to get the point across, but not much larger.
- Credit (partial): if you are not able to get something working, or unable to generate a particular figure, explain why in your report. If you don't explain, I can't give partial credit.

Bonus (Required for CS 660): Model-Free RL using neural networks (10 points)

One of the problems with tabular reinforcement learning is that those methods can have trouble scaling to larger environments. This is where function approximation methods shine! In this part of the assignment, you will implement Q-Learning using a neural network instead of a table to represent your Q function. To make things a little more worthy of a neural network, you will be using this agent to solve the 8x8 version of the frozen lake problem. One thing to note up front, the 8x8 problem doesn't represent a drastic increase in state space. In fact, it is pretty easy to solve this with a tabular Q-learning agent; however, this is mainly about seeing how your approach to the problem changes when using a neural network versus tabular methods. Also to note, this is not a class on designing neural networks, therefore you are free to use whatever libraries you see fit to implement your neural net. Tensorflow? Sure. Pytorch? Awesome. Scikit-learn? Amazing. You still need to set up the RL problem yourself, but the neural net can be implemented using your favorite ML library.

As before, please construct graphs that show the learning rate for your agent. If you are feeling especially daring, you can compare your tabular Q-learning agent against the neural Q-learning agent on this problem in terms of learning rates. It's not required, but could be fun (as most things in life are!).