Deep Reinforcement Learning (Spring 2023)
Assignment 1

March 8, 2023

In this homework, you will implement methods you learned from the first two weeks to solve the `FrozenLake` environment from Gymnasium (a maintained fork of OpenAI's Gym) under different conditions.

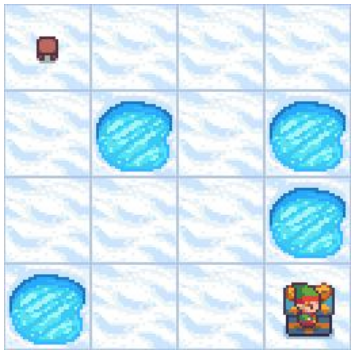
1 About the Environment

Winter is here. You and your friends were tossing around a frisbee at the park when you made a wild throw that left the frisbee out in the middle of the lake.

The lake is mostly frozen, but there are a few holes where the ice has melted. If you step into one of the holes, you'll fall into the freezing cold water. At this time, there's an international frisbee shortage, so it's absolutely imperative that you navigate across the lake and retrieve the disc. However, the ice is slippery, so you won't always move in the direction you intend.

TL;DR: the episode ends when you reach the goal or fall in a hole. You receive a reward of 1 if you reach the goal, and zero otherwise. In the slippery environment, you have a 1/3 probability of heading the corresponding direction of your action, and 1/3 of heading each side apart from the opposite direction.

Oooops!



Hooray!

For setup, please use `conda` or `venv` to create a new environment (we recommend using `Python 3.10` for the new env). Then run the command `pip install -r requirements.txt` to install the packages needed.

In this assignment, you should only write code under the "`YOUR IMPLEMENTATION HERE`" sections, except for Question (b) of Temporal Difference Learning where you may change the hyperparamaters. **Modifying other parts of the code may cause malfunction of our auto grader, so do it at your own risk.**

2 Dynamic Programming

In this problem, you will solve the `FrozenLake` environment through Dynamic Programming, that is, you are given the dynamics (transition probability) of the environment. [50pts]

(a) **(coding)** Read through `dp.py`. Implement `policy_evaluation`, `policy_improvement` and `policy_iteration`. Using the default hyparameters, return the optimal value function and the optimal policy. [20 pts]

(b) **(coding)** Implement `value_iteration` in `dp.py`. Return the optimal value function and the optimal policy. [20 pts]

(c) **(written)** Run both methods on the `FrozenLake-v1` and `SlipperyFrozenLake-v1` environments. In the second environment, the dynamics of the world is stochastic. How does stochasticity affect the resulting policy? Can you think of another `SlipperyFrozenLake` environment with the same state and action space and different transition function, such that the optimal policy is different? [10 pts]

3 Temporal Difference Learning

In this problem, you will solve the Frozen Lake environment through Temporal Difference learning. The dynamics of the world is unknown this time. [50 pts]

(a) **(coding)** Read through `td.py`. Implement `epsilon_greedy_policy` and `sample_action`. [20 pts]

(b) **(coding)** Implement `Q_learning_step` and `Sarsa_step`. Run both methods on the `FrozenLake-v1` environment with `is_slippery=False`. Return the optimal policy. You can adjust the hyparameters of α and γ, report your findings. [20 pts]

(c) **(written)** Can you think of one real world example of the MDP? Describe a MDP with discrete state and action space, explain the transition dynamics of it and how to turn it into a RL problem. [10 pts]