ME5406 Deep Learning for Robotics

Project for Part I: The Froze Lake Problem and Variations

Project Description and Requirement

Dr. Peter C. Y. Chen
Associate Professor
Department of Mechanical Engineering
National University of Singapore
Email: mpechenp@nus.edu.sg

Project report and code due on **4 March 2024, 11:59 PM** Singapore time

I. Objective

This project is designed for the student to demonstrate, through independent learning, (1) competence in implementing a set of model-free reinforcement learning techniques in a small scale problem setting, and (2) understanding of the principles of, and implementation issues related to, this set of techniques.

II. Problem Statement

Consider a frozen lake with (four) holes covered by patches of very thin ice. Suppose that a robot is to glide on the frozen surface from one location (i.e., the top left corner) to another (bottom right corner) in order to pick up a frisbee there, as is illustrated in Fig. 1.

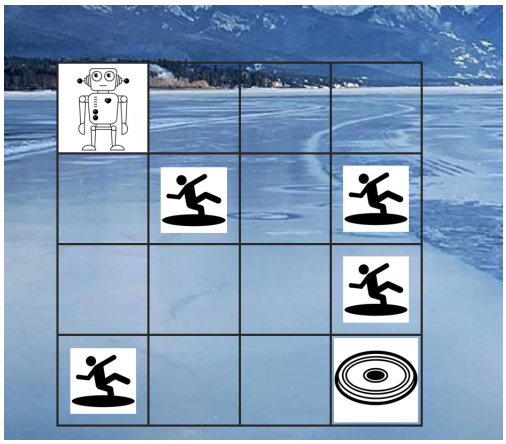

Fig. 1: A robot moving on a frozen lake. The 'falling-man' icon indicates a hole in the frozen lake.

The operation of the robot has the following characteristics:

1. At a state, the robot can move in one of four directions, left, right, up, and down.
2. The robot is confined within the grid.
3. The robot receives a reward of +1 if it reaches the frisbee, −1 if it falls into a hole, and 0 for all other cases.
4. An episode ends when the robot reaches the frisbee or falls into a hole.

III. Requirement

A. What to be done

Three tasks as described below are to be completed for this project. The percentage associated with each task indicates the mark weightage.

Task 1: Basic implementation (25%)

Write a Python program to compute an optimal policy for the Frozen Lake problem as described in Section II, using the following three tabular (i.e., not involving any use of a neural network) reinforcement learning techniques:

1. First-visit Monte Carlo control without exploring starts.
2. SARSA with an ϵ-greedy behavior policy.
3. Q-learning with an ϵ-greedy behavior policy.

You can choose the values for all the necessary parameters, such as discount rate, learning rate, etc.

Task 2: Extended implementation (25%)

Increase the grid size to at least 10×10 while maintaining the same proportion between the number of holes and the number of states (i.e., $4/16 = 25\%$). Dis-

tribute the holes randomly without completely blocking access to the frisbee. Repeat Task 1.

Task 3: Report (50%)

Write an individual report that describes the implementation and discusses the results. This report should be no more than 10 pages (excluding the cover page). The report should

1. compare and contrast the performance[1] of the three reinforcement learning techniques and the results that they have generated in your implementations,
2. provide explanations for any similarities and differences observed,
3. discuss any simulation outcomes that appear unexpected and provide plausible explanations,
4. highlight the difficulties encountered and describe how they were overcome during the project,
5. elaborate on your own initiatives (if any) to investigate and improve the efficiency of these techniques in solving the given problem, and
6. present any other issues that you think are significant (and explain why) concerning your implementation of the three reinforcement learning techniques.

B. Python *programming*

For setting up the "frozen lake environment", you can use publicly available toolkits (such as OpenAI gym) or write the code yourself. The advantage of the latter option is that you will learn how to implement the "low-level" features of a reinforcement learning problem.

Your Python code must be able to run on Windows or Linux under Python 3.6 — either as a Jupyter Notebook or in plain Python code, and use only standard and publicly available packages. **For programming, Jupyter is recommended**. Code consisting of multiple *.py* files must include a main *.py* file to execute everything. If you develop your code in MacOS, please test-run the code on a Windows/Linux machine before submitting it.

Coding convention is to be observed. In particular, clear and concise comments should be included in the source code to explain various calculation steps, e.g., how the number of first visits to a state-action pair is computed, and how an exploratory action in SARSA and Q-learning is selected, etc. The explanation should be detailed and specific; brief and general comments such as *"These lines compute the value for [something]"* are not adequate.

[1]You will need to define your performance evaluation scheme. You may want to (if needed) search for relevant information available on the internet as part of your independent-learning effort.

C. What to submit

1. An individual report in a PDF file. The name of the PDF file must be in the format:

   StudentNumber_ME5406_Project1.pdf

   The report must contain a cover page showing:
   a) Student name
   b) Student number
   c) Student email address
   d) Name of module
   e) Project title
2. The Python code in either plain text (**with the *.py* filename extension**) or as a Jupyter Notebook, together with (i) any auxiliary files needed in order to run your code (such as a background image for the environment, if any) and (ii) **step-by-step instruction** on how to run the code on a Windows or Linux machine.

D. How to submit

Only softcopy of the report (in PDF) and the Python code are to be submitted. Please put the report and the code file in a folder. Use your student number as the folder name. Generate a non-password-protected zipfile of this folder and upload this zipfile (again, with your student number as the filename of the zipfile) onto CANVAS in the folder *Project* 1 *report submission*, under the *Assignments* section of the course *ME5406 Deep Learning for Robotics*.

E. Due date

The project report and code are due on 4 March 2024, 11:59 PM, Singapore time.

IV. Assessment

The project will be assessed based on the contents and the presentation of the report, and the functionality and readability of the Python code.