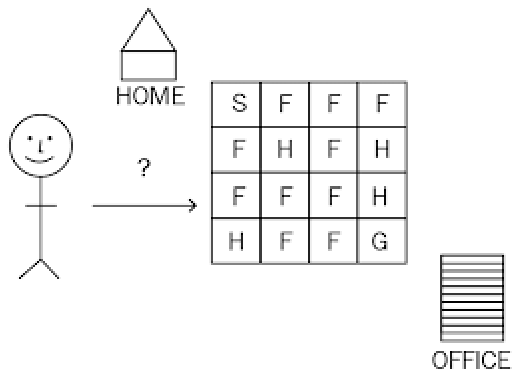
VirginiaTech Invent the Future — The Bradley Department Electrical & Computer Engineering

ECE 5984 – Deep Reinforcement Learning

Project Assignment #1: The Frozen Lake Problem
Dynamic Programming (I): Value Iteration

Problem: Imagine there is a frozen lake stretching from you home to your office; you have to walk on the frozen lake to reach your office. But oops! There are holes in the frozen lake so you have to be careful while walking on the frozen lake to avoid getting trapped in the holes:

Look at the proceeding diagram:



S is the starting position (home)
F is the frozen lake where you can walk
H are the holes, which you have to be so careful about
G is the goal (office)

Okay, now let us use an agent instead of you to find the correct way to reach the office. The agent's goal is to find the optimal path to go from S to G without getting trapped at H. We give +1 point as a reward to the agent if it correctly walks on the frozen lake and 0 point if it falls into a hole, so the agent can determine which is the right action. An agent will now try to find the optimal policy. Optimal policy implies taking the correct path, which maximizes the agent's reward.

In this assignment, you are asked to implement a *dynamic programming* (DP) algorithm to find the optimal policy. Note that you can use ***value iteration*** to solve the frozen lake problem in this assignment.

Value Iteration

A. Set up/import Python libraries for this project

```
import gym
import numpy as np

# make the frozen lake environment using OpenAI’s Gym
env = gym.make(‘Frozen Lake-v1’) # or the latest version

# explore the environment
print(env.observation_space.n)

print(env.action_space.n)
```

B. Implement the `value_iteration`() function outlined below:

```
def value_iteration(env, gamma = 1.0):
    """
    Inputs:
    - env: the frozen lake environment.
    - gamma: discount factor

    Returns:
    - value_table: state value function
    - Q_value: state-action value function (Q function)
.
    """
```

We can derive `optimal_value_function` using the `value_iteration():`

```
optimal_value_function = value_iteration(env=env, gamma=1.0)
```

C. Implement the `extract_policy()` function to extract the optimal policy from the `optimal_value_function, as` outlined below:

```
def extract_policy(value_table, gamma = 1.0):
    """
    Inputs:
    - value_table: state value function
    - gamma: discount factor

    Returns:
    - policy: the optimal policy
.
    """
```

We can extract the `optimal_policy` using the `extract_policy():`

```
optimal_policy = extract_policy(optimal_value_function,
                                gamma=1.0)
```

Thus, we can derive the optimal policy, which specifies what action to perform in each state, using ***value iteration*** to solve the frozen lake problem.

You need to prepare **a written report** (in the pdf format) including the following sections: (1) Approach(es), (2) Experimental Results, and (3) Discussion.

In addition, you need to attach your **implementation codes/notebooks** as separate files to the report.