Homework: Frozen Lake

ELEN E6885: Introduction to Reinforcement Learning

1 Introduction

1.1 Algorithms

The pseudo code for Policy Iteration, Value Iteration, Q-learning and SARSA:

1. Initialization
   $v(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $temp \leftarrow v(s)$
   $v(s) \leftarrow \sum_{s'} p(s'|s, \pi(s))\Big[r(s, \pi(s), s') + \gamma v(s')\Big]$
   $\Delta \leftarrow \max(\Delta, |temp - v(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $temp \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s'} p(s'|s, a)\Big[r(s, a, s') + \gamma v(s')\Big]$
   If $temp \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return v and π; else go to 2

(a) Policy Iteration

Initialize array v arbitrarily (e.g., $v(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
$\Delta \leftarrow 0$
For each $s \in \mathcal{S}$:
$temp \leftarrow v(s)$
$v(s) \leftarrow \max_a \sum_{s'} p(s'|s, a)[r(s, a, s') + \gamma v(s')]$
$\Delta \leftarrow \max(\Delta, |temp - v(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
$\pi(s) = \arg\max_a \sum_{s'} p(s'|s, a)\Big[r(s, a, s') + \gamma v(s')\Big]$

(b) Value Iteration

Input: Arbitrary $Q(S, A)$ and $Q(\text{terminal state}, \cdot) = 0$
Output: (near-) optimal policy π^*
Repeat: for each episode
Initialize S
repeat
Choose action A from state S following the policy π derived by Q (e.g., ϵ-greedy policy)
Take action A, obtain reward R and the next state S'
Update Q values as

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$$

$S \leftarrow S'$
until S is the terminal state

(c) Q-Learning

Input: Arbitrary $Q(S, A)$ and $Q(\text{terminal state}, \cdot) = 0$
Output: (near-) optimal policy π^*
Repeat for each episode:
Initialize: S
Choose A from S following the policy π derived by Q (e.g. ϵ-greedy)
Repeat: For each step of the episode:

1. Take action A, obtain reward R and the next state S'
2. Choose A' for state S' following the policy derived by Q (e.g. ϵ-greedy)

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$$
$$S \leftarrow S', A \leftarrow A'$$

3. Until S is the terminal state

(d) SARSA

1.2 Implementation - OpenAI gym(FrozenLake-v1)

- Background
  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 water is mostly frozen, but there are a few holes where the ice has melted. If you step into one of those holes, you'll fall into the freezing 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. The episode ends when you reach the goal or fall in a hole [1].

- observation:
  The states form a 4 * 4 grid. There are 4 kinds of states. "S" is the safe starting point, "F" represents frozen surface, which is safe as well. "H" represents a hole. You "fall to your doom" if you enter "H" states. "G" is your goal where the frisbee is located.

  S F F F
  F H F H
  F F F H
  H F F G

- Actions:
  Type: Discrete(4)
  At each step, you can take 4 actions: "LEFT", "DOWN", "RIGHT", "UP" represented by indices 0 to 3 respectively. Your next state is then given by the environment.

- Reward:
  You receive a reward of 1 if you reach the goal, and 0 otherwise.

- Episode Termination:
  The episode ends when you reach the goal or fall in a hole.

2 Environment Setup

We are using Google Colab [2] to execute the code. Colaboratory, or "Colab" for short, is a product from Google Research, allowing anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

Steps to use the environment with start code:
1. Upload the start code onto your Google Drive Account and open the FrozenLake.ipynb by using colab. (Right click on the file ->open with ->Google Colaboratory)
2. Mount the Google Drive onto the Colab as the storage location
Following the instructions returned from the below cell. You will need to click a web link

and select the google account you want to mount (The one that you upload the start code), then copy the authorication code to the blank, press enter.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

3. Append the directory location where you upload the start code folder to the sys.path

```
import sys
sys.path.append('/content/gdrive/My Drive/<dir/to/start/code/folder>'))
```

4. Then follow the detailed instructions step by step in FrozenLake.ipynb.

3 Task

You need to implement the following algorithms in RLalgs folder: Epsilon-Greedy, Policy Iteration, Value Iteration, Q-Learning and SARSA. Follow the detailed instructions step by step in FrozenLake.ipynb which interactively show the result of each algorithms. We also have gym-tutorial.ipynb for you to get familiar with Gym package.

4 Submission Instructions

Please submit the following two files:
1. A pdf report (printed out from Jupiter notebook with all codes and outcomes).
2. A zip file including folder "RLalgs", and your Jupyter notebook source file (for us to check your results).

References

[1] http://gym.openai.com/envs/FrozenLake-v0/

[2] https://colab.research.google.com/notebooks/intro.ipynb#recent=true