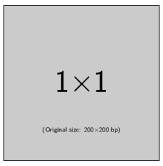


Carnegie University

Reinforcement Learning Project

# Frozen Lake

*Hao Xu*

Course-based project in
10703 Reinforcement LearningBrown

March 4, 2017

## Background

The FrozenLake environment is from the OpenAI Gym Environment API. For more information on the Gym and the API see: https://gym.openai.com/.

There are three different tile types: frozen, hole, and goal. When the agent lands on a frozen tile it receives 0 reward. When the agent lands on a hole tile it receives 0 reward and the episode ends. When the agent lands on the goal tile it receives +1 reward and the episode ends.

Map size: $4 \times 4$ and $8 \times 8$

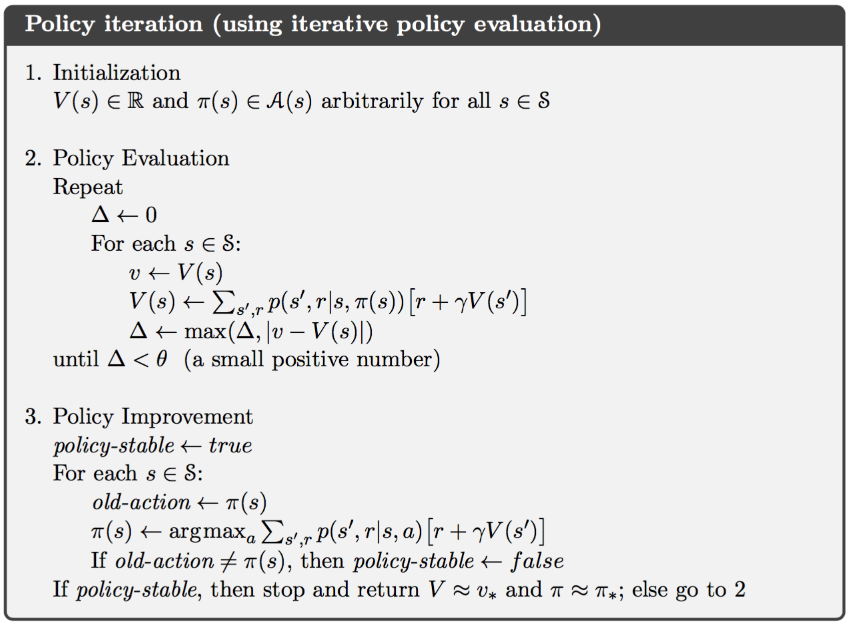
States are represented as integers numbered from left to right, top to bottom starting at zero. So the upper left corner of the 4x4 map is state 0, and the bottom right corner of the 4x4 map is state 15.

## SARSA method

I'm using Policy Iterative Method to practice the environment, for more information of the method, please review the book at page 87:

http://webdocs.cs.ualberta.ca/ sutton/book/bookdraft2016sep.pdf

Figure 1 shows the main algorithms used in the practice.



**Policy iteration (using iterative policy evaluation)**

```
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 $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s)) \big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$  (a small positive number)

3. Policy Improvement
   $\textit{policy-stable} \leftarrow \textit{true}$
   For each $s \in \mathcal{S}$:
       $\textit{old-action} \leftarrow \pi(s)$
       $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a) \big[r + \gamma V(s')\big]$
       If $\textit{old-action} \neq \pi(s)$, then $\textit{policy-stable} \leftarrow \textit{false}$
   If $\textit{policy-stable}$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```

**Value iteration**

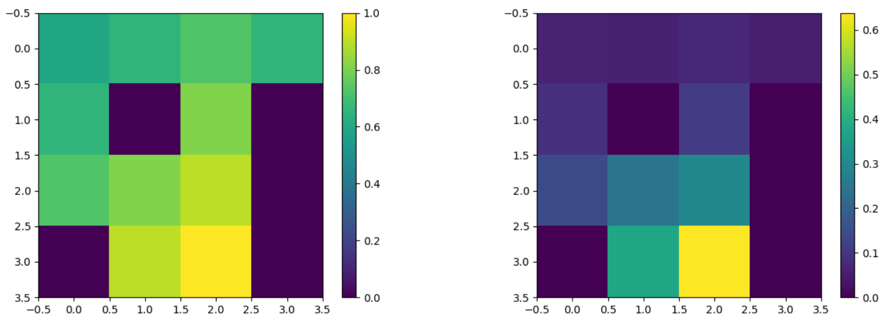
```
Initialize array $V$ arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

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

Output a deterministic policy, $\pi \approx \pi_*$, such that
    $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
```

In the deterministic versions, the up action will always move the agent up, the left will always move left, etc. In the stochastic versions, the up action will move up with probability 1/3, left with probability 1/3 and right with probability 1/3.

The left side of figure 2 shows the value function color plot for 4x4 determinstic, and 4x4 stochastic version environment. And The right side is their optimal policy respectively.



| | | | |
|---|---|---|---|
| D | R | D | L |
| D | L | D | L |
| R | D | D | L |
| L | R | R | L |

| | | | |
|---|---|---|---|
| L | U | L | U |
| L | L | L | L |
| U | D | L | L |
| L | R | D | L |

The policy iteration method is going to take 85 value steps and 6 improve steps, while the value iteration method is only taking 22 steps.