ECE 1508: Reinforcement Learning

Chapter 1: Introduction

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Fall 2025

Frozen Lake Game

The guy needs to get to the treasure

- *It could walk over frozen cells*
- *If it runs to a shallow cell it fells into water*

If he decides for a direction, it could

- *move to direction with probability* $1 - p$
- *slip with probability* p *to another direction*

It wants to learn the best way, i.e.,

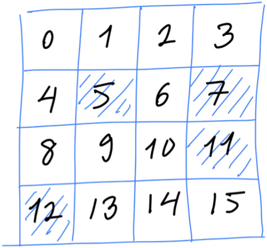
with minimal chance of failing

Playing in RL Framework

For this problem, we should first define *the main three components*

- *State*
- *Action*
- *Reward*

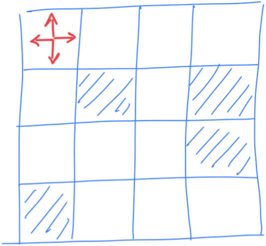
State $\equiv$ *Location of the guy on the game board*



$$\mathbb{S} = \{0, 1, \dots, 15\}$$

Playing in RL Framework

Action $\equiv$ *Possible moves on the board*

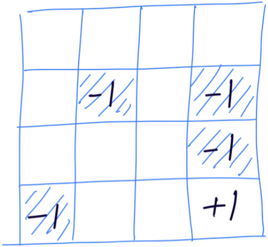


We can represent them with some integer

$$\mathbb{A} = \{0 \equiv \texttt{left}, 1 \equiv \texttt{down}, 2 \equiv \texttt{right}, 3 \equiv \texttt{up}\}$$

Playing in RL Framework

Reward $\equiv$ *Possible outcomes of the game*



We have three possible cases

$$R \in \{0 \equiv \text{ongoing}, 1 \equiv \text{winning}, -1 \equiv \text{losing}\}$$

Playing in RL Framework

+ *Can we describe the environment in this problem?*
– Sure!

Rewarding Function

Reward of each state-action pair is a random variable

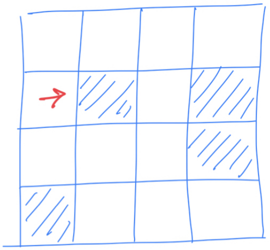
$$\mathcal{R}\left(s,a\right) = \begin{cases} \text{intended reward} & 1-p \\ \text{reward of slipping direction} & p \end{cases}$$

Transition Function

Transition of each state-action pair is also a random variable

$$\mathcal{P}\left(s,a\right) = \begin{cases} \text{intended state} & 1-p \\ \text{state of slipping direction} & p \end{cases}$$

Playing in RL Framework

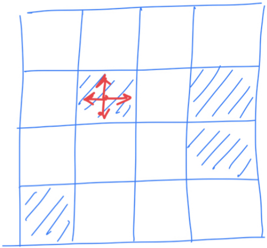


For example, in the above image we could say

$$\mathcal{P}\left(4, 2\right) = \begin{cases} 5 & 1-p \\ 0 \text{ or } 8 & p \end{cases} \qquad \textit{and} \qquad \mathcal{R}\left(4, 2\right) = \begin{cases} -1 & 1-p \\ 0 & p \end{cases}$$

Terminal State

+ *But, how can we specify the end of the game?*
– We need to define a terminal state



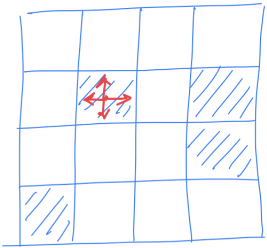
We can think of the end as a state which does not change anymore

$$\mathcal{P}(5, a) = 5$$

for all actions $a \in \mathbb{A}$

Terminal State

+ *What will be the reward then?*
– We keep getting zero reward



We can think of the end as a state which does not change anymore

$$\mathcal{R}(5, a) = 0$$

for all actions $a \in \mathbb{A}$

Terminal State

Terminal State

State s is called terminal if for any action $a \in \mathbb{A}$, we have

$$\mathcal{P}(s, a) = s \qquad \textit{and} \qquad \mathcal{R}(s, a) = 0$$

A basic property of terminal state is that *it has zero value*: *say s is terminal*

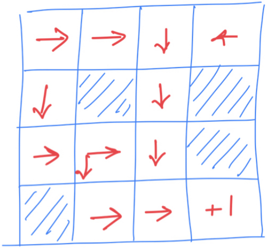
$$v_\pi(s) = \mathbb{E}_\pi\{G_t | S_t = s\} = \mathbb{E}_\pi\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s\right\}$$
$$= \mathbb{E}_\pi\left\{0 + \gamma 0 + \gamma^2 0 + \ldots | S_t = s\right\} = 0$$

Episode

The trajectory from starting state to a terminal state is often called an episode

Playing in RL Framework: *Policy*

Policy in this problem describes the *planned path towards the goal*

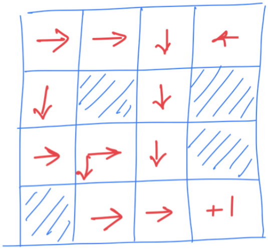


For instance in the above policy we have

$$\pi\left(a|1\right) = \begin{cases} 1 & a = 2 \\ 0 & a \neq 2 \end{cases} \qquad \text{and} \qquad \pi\left(a|9\right) = \begin{cases} 0.5 & a = 2 \\ 0.5 & a = 1 \\ 0 & a = 0, 3 \end{cases}$$

Playing in RL Framework: *Value*

Value describes the *chance of getting to the goal if we play the given policy*

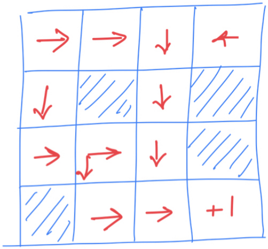


For instance, if slipping probability is $p = 0$ *the above policy leads to*

$$v_\pi(s) = \begin{cases} 1 & s \text{ is not terminal} \\ 0 & s \text{ is terminal} \end{cases}$$

Playing in RL Framework: *Optimal Policy*

Policy that maximizes the *chance of getting to the goal*



For instance, if slipping probability is $p = 0$ the above policy is optimal

Gymnasium

Gymnasium is an API with some pre-implemented environments

- *We can call environments and access the state and reward*
- *We can interact through actions*
- *This can is a powerful toolkit to test our RL algorithms*
  ↳ *At some point, we may prefer implementing our own environment*

Gymnasium: *Trying Frozen Lake Game*

We can call the frozen lake environment easily in Gymnasium

```python
import gymnasium as gym

def test():
    # Create FrozenLake instance
    env = gym.make('FrozenLake-v1', map_name="4x4", is_slippery=False, render_mode='human')

    env.reset()  # Initialize to state 0
    terminated = False      # True when agent falls in hole or reached goal
    truncated = False       # True when agent takes more than 200 actions

    while(not terminated and not truncated):

        action = env.action_space.sample()

        # Execute action
        state, reward, terminated, truncated, _ = env.step(action)
        print(reward)
        print(state)
    env.close()
```

We will play around with it in Assignment 1

