OOP Project

Group 18

Members

B123040037 陳聖繹

B123040040 余家睿

B123040061 陳偉財

PART 1

MOUNTAIN CAR

ENVIRONMENT SETUP

Step 1: Create & Activate Virtual Environment

python -m venv .venv

source .venv/bin/activate

Step 2: Install Gymnasium (Editable Mode)

cd group_project/Gymnasium

pip install -e .

Step 3: Install Additional Dependencies

pip install "gymnasium[classic_control]"

pip install matplotlib

Note (Windows Users)

.venv\Scripts\activate

RUNNING PART 1

Content

Train a reinforcement learning agent using Q-learning to solve the MountainCar-v0 environment.

Purpose

Verify that the Gymnasium environment and required dependencies are correctly installed and functioning.

Usage

python mountain_car.py [option]

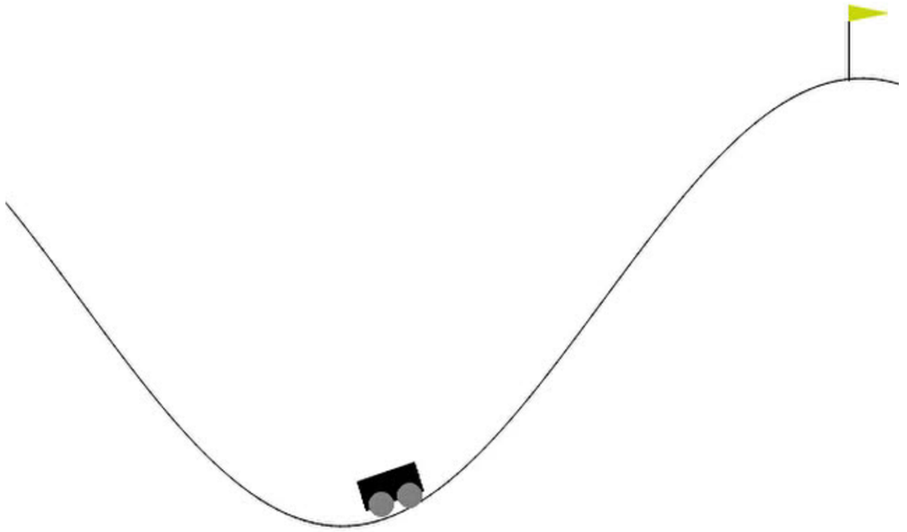
--train	Train the agent
--render	Render the environment
--episodes	Number of episodes to run

RUNNING PART 1

Example

python mountain_car.py --train --episodes 5000

python mountain_car.py --render --episodes 10



PART 2

FROZEN LAKE

FrozenLake-v1 (8x8)

Problem Definition

- Map: 8x8 Grid World
- Goal: Navigate from Start (S) to Goal (G) safely.
- Hazards: Avoid falling into Holes (H).

State & Action Space

- Discrete States: 0 to 63.
- Discrete Actions: 4 (Left, Down, Right, Up).

Key Challenge

- Slippery Surface: Stochastic transitions – The agent has a chance to move in a perpendicular direction, ignoring the chosen action.
- Sparse Reward: The agent receives a reward of +1 only at the goal and 0 everywhere else. This makes initial learning extremely difficult.

CORE ALGORITHM

Algorithm Choice

- Tabular Q-Learning

Key Hyperparameters

α (Learning Rate): How fast new information overrides old.

γ (Discount Factor): Importance of future rewards (Set to 0.999).

ϵ (epsilon) :Balance between Exploration (random) and Exploitation (best action).

Mechanism

- Maintains a Q-Table of size 64x4 to store the value of each action in each state.
- Agent learns by interacting with the environment (Trial-and-Error).

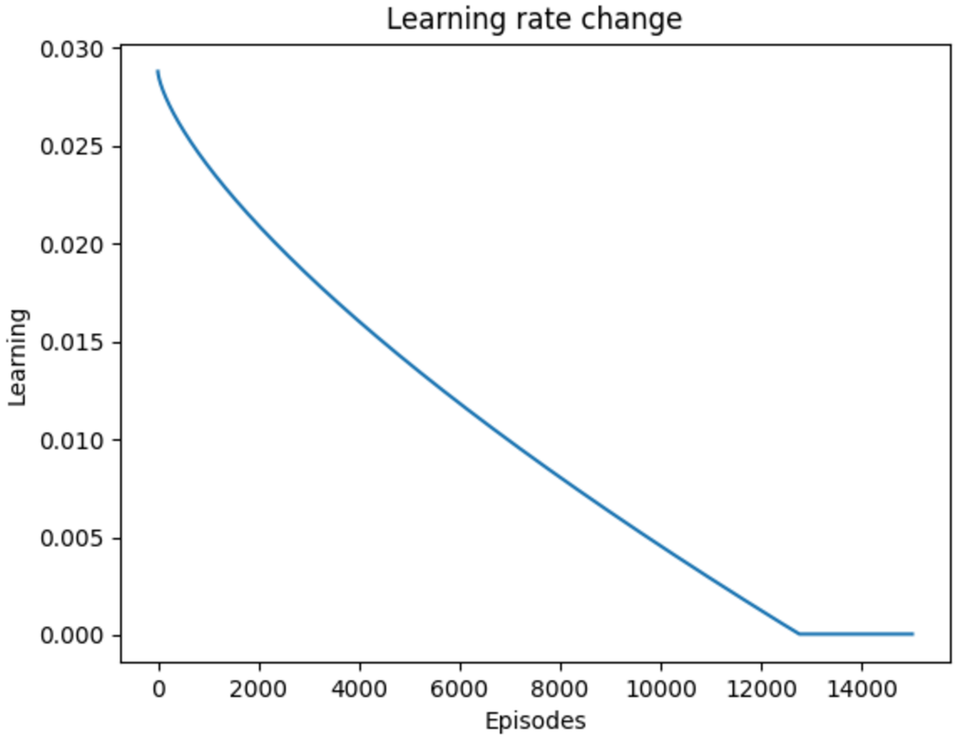
The Update Rule

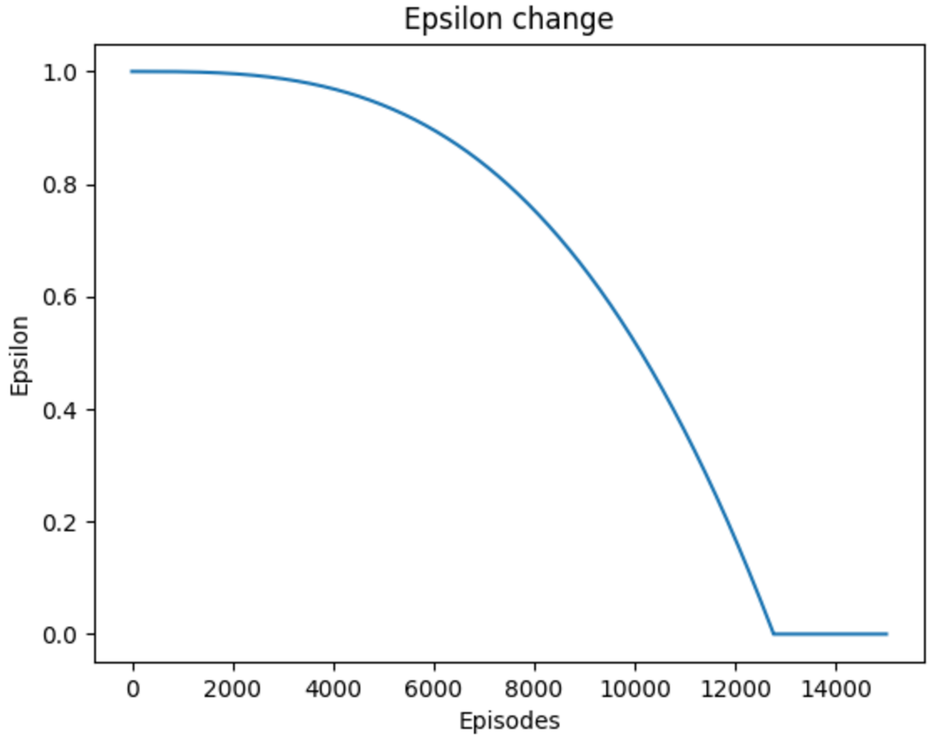
$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

OUR OPTIMIZATION STRATEGY

- PBRS: Accelerate convergence
- Small random Q table initialization (0 ~ 0.001): Prevents rigid initial movements; ensures diversity.
- 85% exploration / 15% convergence:
  - phase 1 (0% ~ 85 %) :
    - epsilon is decayed slowly at the beginning to encourage sufficient exploration, and more aggressively in later stages to stabilize exploitation.
  - phase 2 (85% ~ 100%):
    - eps = 0, learning_rate = min_lr
    - Freezes the policy to fine-tune the best path found
- Optuna hyperparameter tuning
  - Search space:
    - learning rate: 0.02 ~ 0.04
    - min_lr: 0.00001 ~ 0.00005
    - min_eps: 0.001 ~ 0.003

OUR OPTIMIZATION STRATEGY





RUNNING PRAT 2

Purpose

Achieve consistent success rate > 0.70

Example

python frozen_lake.py --train

python frozen_lake.py --eval

Usage

python frozen_lake.py [--train|--tune|--eval] [--render]

--train	Train the agent
--tune	Tune hyperparameters
--eval	Evaluate the trained agent
--render	Render the environment

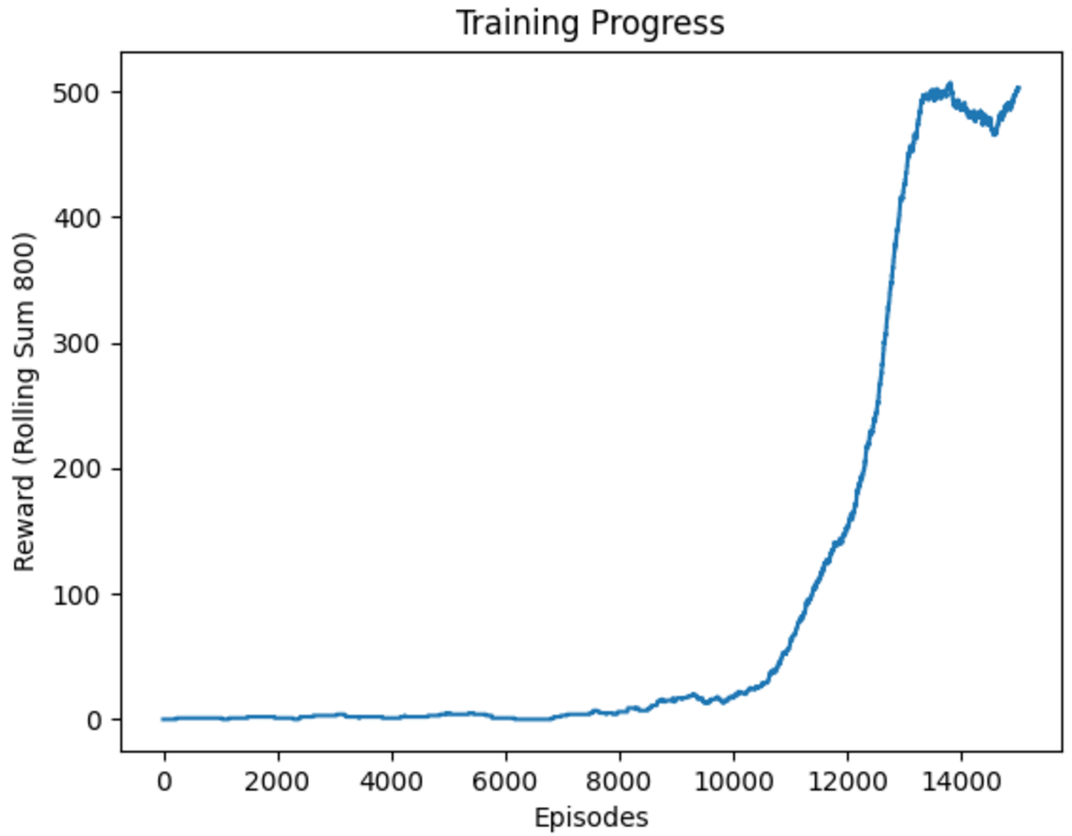
RESULT

Training phase

- Episodes: 15000
- learning_rate_a: 2.8789e-02
- min_lr: 4.9905e-05
- min_eps: 2.0718e-03

Evaluating phase

- Episodes: 800
- Final success rate: 60% ~ 65%



PART 3

PENDULUM

Pendulum-v1 & Discretization Strategy

The Challenge: Continuous vs. Discrete

- Goal: Swing up the pendulum and keep it upright (counter-gravity).
- Problem: The environment is Continuous (Angle, Velocity, Torque), but Q-Learning requires Discrete states and actions.

Our Approach

- Transform Pendulum-v1 into a discretized, modular, and extensible RL system using Object-Oriented Programming (OOP).

pendulum_agent.py

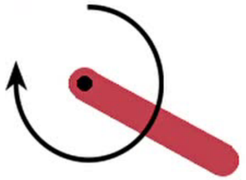
- This module wraps Gymnasium's “Pendulum-v1" environment and provides state/action space discretization, making the continuous control problem suitable for tabular Q-Learning algorithms.

pendulum_env.py

- Creates a Gymnasium-compatible environment wrapper that provides a standard Gym interface for the discretized Pendulum problem. This allows the environment to be used with any Gym-compatible RL framework.

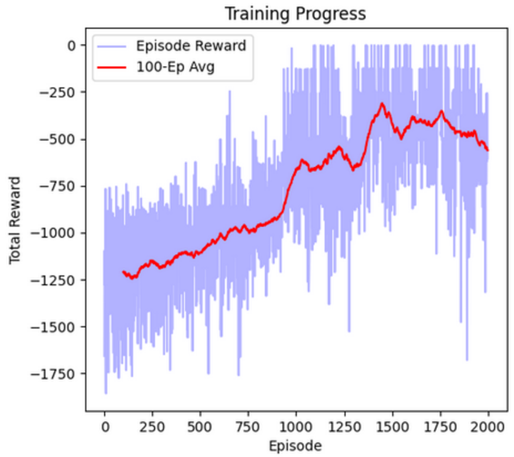
train_pendulum.py

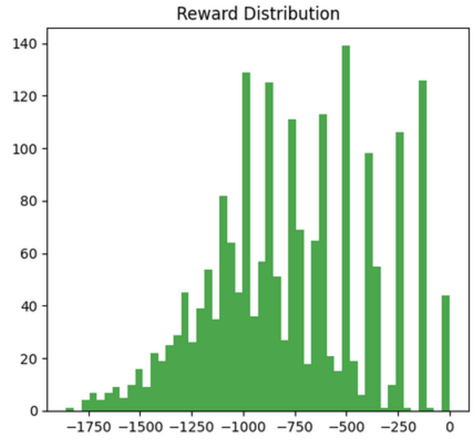
- Implements the complete Q-Learning training pipeline, including agent logic, exploration strategies, and experiment orchestration. This module demonstrates advanced OOP design patterns.



OOP Concepts

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction
- Composition





Thank You