CSCI B659: Reinforcement Learning
Assignment 2: Experimental Results

LJ Huang

Spring 2025

Contents

1 Task 1: SARSA

In this section, we present our experimental results for Task 1, where we applied the SARSA algorithm to three distinct environments: FrozenLake4, FrozenLake8, and CartPole. Our goal is to demonstrate how SARSA performs across different reinforcement learning problems – from discrete navigation tasks in FrozenLake to the more dynamic control challenge of balancing a pole in CartPole.

1.1 Experimental Plots for Task 1

1.1.1 SARSA FrozenLake4

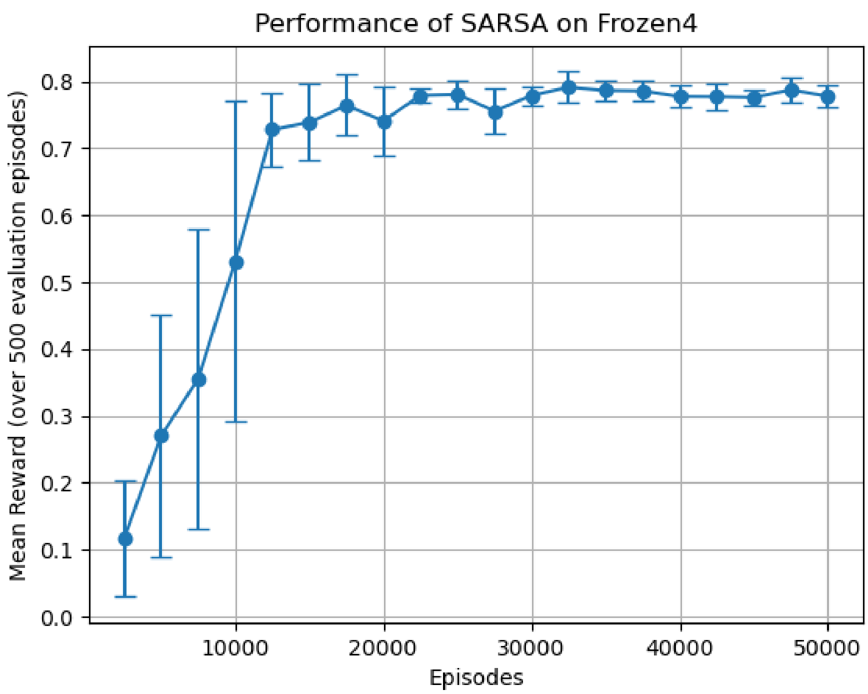


Figure 1: SARSA performance on FrozenLake4

1.1.2 SARSA FrozenLake8

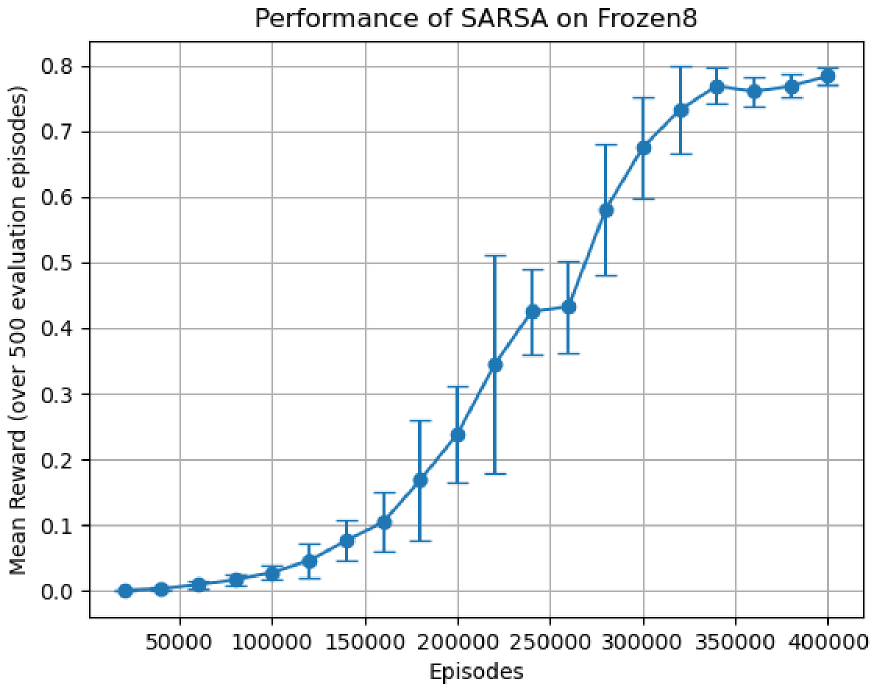


Figure 2: SARSA performance on FrozenLake8

1.1.3 SARSA CartPole

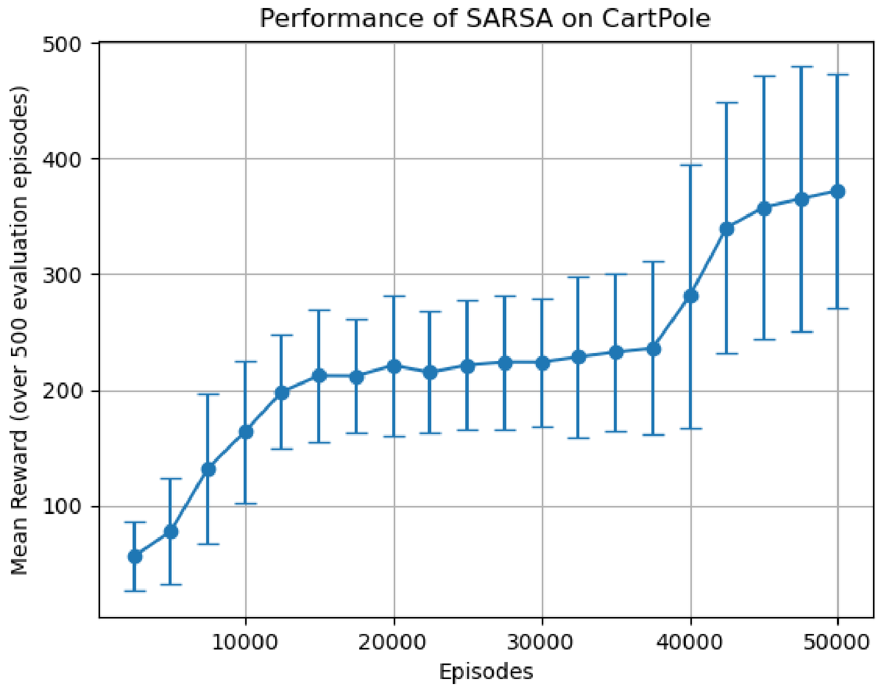


Figure 3: SARSA performance on CartPole

1.2 Observations on Task 1

FrozenLake4: SARSA converges within about 10,000 episodes to an average reward near 0.8, with error bars indicating stable performance after roughly 20,000–30,000 episodes.

FrozenLake8: Due to its larger state space, the agent requires hundreds of thousands of episodes to stabilize, eventually achieving mean rewards between 0.8 and 0.85 after extensive exploration.

CartPole: The mean reward steadily increases during the first 30,000 episodes and then accelerates, reaching values near 400–450 by 50,000 episodes, which reflects improved balance over time despite some variability.

Variant of SARSA Used: We employed an on-policy SARSA algorithm with an ϵ-greedy exploration strategy, where ϵ decays linearly from 0.5 to 0. Key hyperparameters are:

- Learning rate: $\alpha = 0.01$
- Discount factor: $\gamma = 0.999$

Sensitivity to Hyperparameters: SARSA is sensitive to these settings. A too-large α may cause unstable updates, while a too-small α can slow convergence. Similarly, if ϵ decays too quickly, the agent may exploit prematurely; if it remains high too long, convergence is delayed.

2 Task 2: Model-Based RL

2.1 Experimental Plots for Task 2

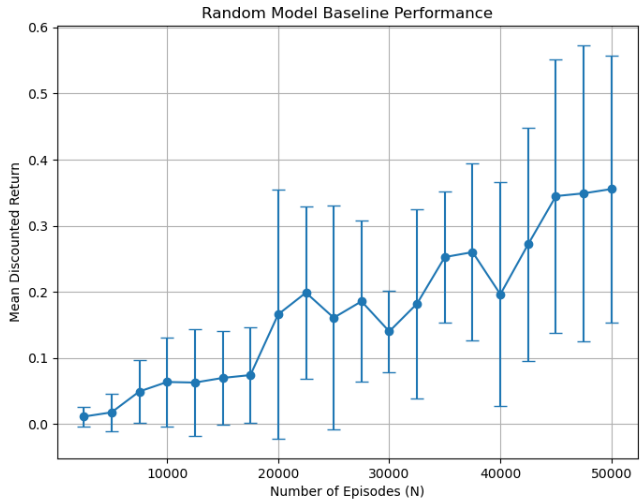
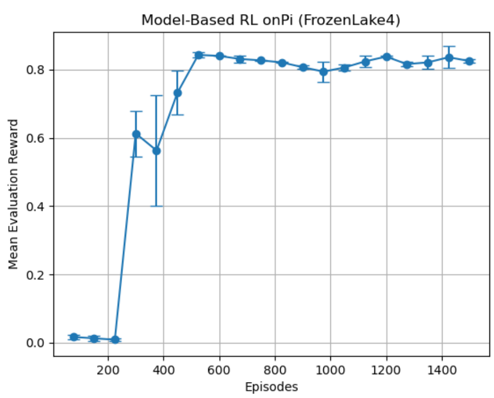
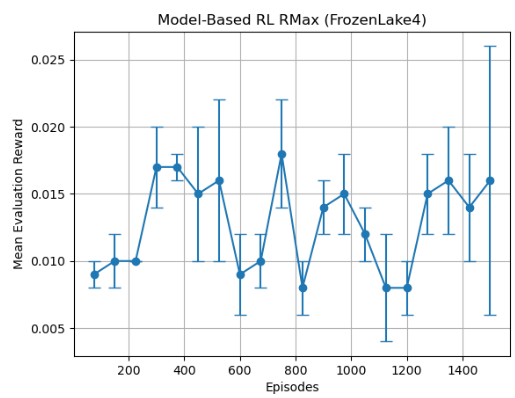


Figure 4: Random Model Baseline (FrozenLake)

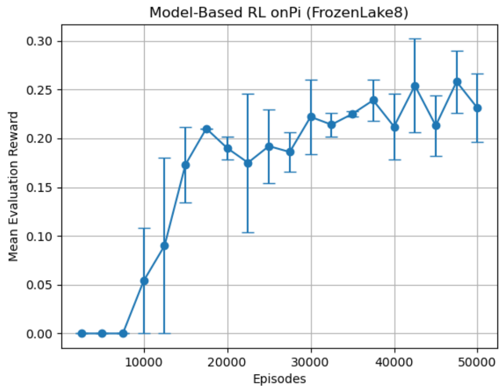


(a) onPi Variant on FrozenLake4

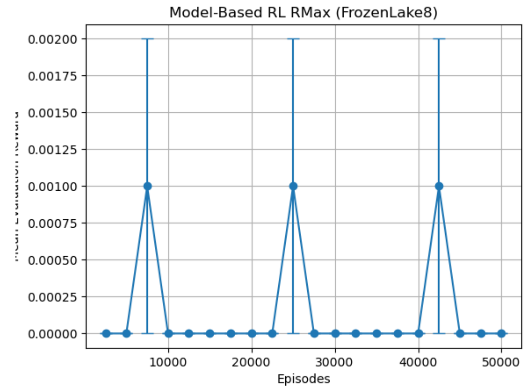


(b) RMax Variant on FrozenLake4

Figure 5: Results on FrozenLake4



(a) onPi Variant on FrozenLake8



(b) RMax Variant on FrozenLake8

Figure 6: Results on FrozenLake8

2.2 Observations on Task 2

Success of Each Algorithm:

- **Random:** Learns a workable policy eventually but is slow and requires large amounts of data.
- **onPi:** Achieves good performance on FrozenLake4 quickly and learns a reasonable policy on FrozenLake8, indicating faster convergence.
- **RMax:** RMax did not perform well in our experiments, possibly due to code implementation issues in `Model_Based_RL.py` or the need for additional parameter tuning.

Performance and Speed of Learning:

- **onPi** demonstrates the fastest and most stable learning, particularly in FrozenLake4.
- **Random** improves only with large data collection and is slower overall.
- **RMax** may have potential when correctly implemented and tuned but did not show a clear advantage in these experiments.

3 README File

```
# CSCI B659: Reinforcement Learning - Assignment 2
**Author:** LJ Huang
**Semester:** Spring 2025

## Overview
This repository contains the code and report for Assignment 2. The assignment covers two main tasks:
1. **Task 1:** SARSA implementation in FrozenLake4, FrozenLake8, and CartPole.
2. **Task 2:** Model-based RL in FrozenLake4 and FrozenLake8 environments.

## Directory Structure
- **VI_PI_MPI.py**
  Contains the implementation for Task 1.
- **Model_Based_RL.py**
  Contains the implementation for Task 2.
- **pp2starter.py**
  The provided startup file for setting up the FrozenLake environment (including rewards, number of states, and actions).
- **hw2_report.pdf**
  The report containing code printouts, experimental results, plots, and discussion of the findings.
- **README.md**
  This file.

## Dependencies
- Python 3.x
- Gymnasium (Install with ‘pip install gymnasium‘)
- NumPy (Install with ‘pip install numpy‘)
- Matplotlib (Install with ‘pip install matplotlib‘)

## Installation and Running
1. **Clone or Download the Repository:**
   Clone the repository or download the zip file and extract its contents.

2. **Run the Code:**
   \begin{verbatim}
   python SARSA.py        # Task 1
   python Model_Based_RL.py  # Task 2
   \end{verbatim}
```

Listing 1: README.file