Reinforcement Learning Interactive Learning Platform

Project Report

Radwa Badran

December 29, 2025

Contents

1 Introduction

This report provides a detailed description of the web-based Reinforcement Learning (RL) interactive learning platform developed as part of the project. The platform is designed to help students and beginners intuitively understand key RL algorithms, environments, parameter adjustments, and visualization techniques through real-time interaction.

2 Algorithms Implemented

The platform implements both model-based and model-free reinforcement learning algorithms, including:

- **Policy Evaluation:** Computes the value function for a given policy.
- **Policy Iteration:** Alternates between policy evaluation and policy improvement until convergence.
- **Value Iteration:** Uses the Bellman optimality equation to iteratively compute the optimal value function.
- **Monte Carlo (MC):** Estimates value functions based on averaging returns from complete episodes.
- **Temporal Difference (TD) Learning:** Combines ideas from MC and dynamic programming to update value estimates incrementally.
- **n-step TD:** Extends TD learning by updating values based on multiple future steps.
- **SARSA:** On-policy TD control algorithm.
- **Q-Learning:** Off-policy TD control algorithm.

Files: `backend/app/algorithms/`

3 Environments Implemented

The platform provides a diverse set of environments for practicing RL, ranging from tabular MDPs to control and custom pedagogical environments:

- **GridWorld:** Classic discrete environment for policy/value learning.
- **FrozenLake:** Stochastic environment for navigation tasks.
- **CartPole:** Control task with continuous state and discrete action space.
- **MountainCar:** Continuous state environment for learning delayed rewards.
- **Breakout:** Custom environment for interactive gameplay-based learning.
- **Gym4ReaL:** Custom pedagogical environment designed for education.

Files: `backend/app/environments/`

4 Parameter Adjustment Capabilities

The platform allows users to dynamically adjust key RL parameters, providing hands-on experimentation:

- **Learning Rate (α):** Controls how quickly the agent updates its value function or Q-values.
- **Discount Factor (γ):** Determines the importance of future rewards.
- **Exploration Rate (ϵ):** Governs the trade-off between exploration and exploitation.

These parameters can be modified through the frontend interface using sliders and input fields, with changes immediately reflected in the agent's learning behavior.

Files: `backend/app/api/routes.py, backend/app/trainers/env_trainers.py, frontend/app`

5 Visualization Techniques

To facilitate intuitive understanding of RL, the platform provides real-time visualizations of both agent behavior and training progress:

- **Environment Rendering:** Agents and states are rendered on a canvas in real-time.
- **Agent Trajectory Animation:** Shows the path of the agent through the environment.
- **Value/Policy Updates:** Displays evolving value functions and policies.
- **Learning Curves:** Plots cumulative rewards and convergence over episodes.

Files: `frontend/app.js, backend/app/utils/visualizer.py, frontend/style.css`

6 System Architecture

The platform follows a clean, modular, and extensible architecture:

- **Backend:** Implements algorithms, environment logic, trainers, and REST API endpoints.
- **Frontend:** Handles user interactions, parameter adjustments, and real-time visualizations.
- **API Communication:** Uses REST endpoints to communicate parameter changes and training commands.

Files: `backend/app/main.py, backend/app/api/routes.py`