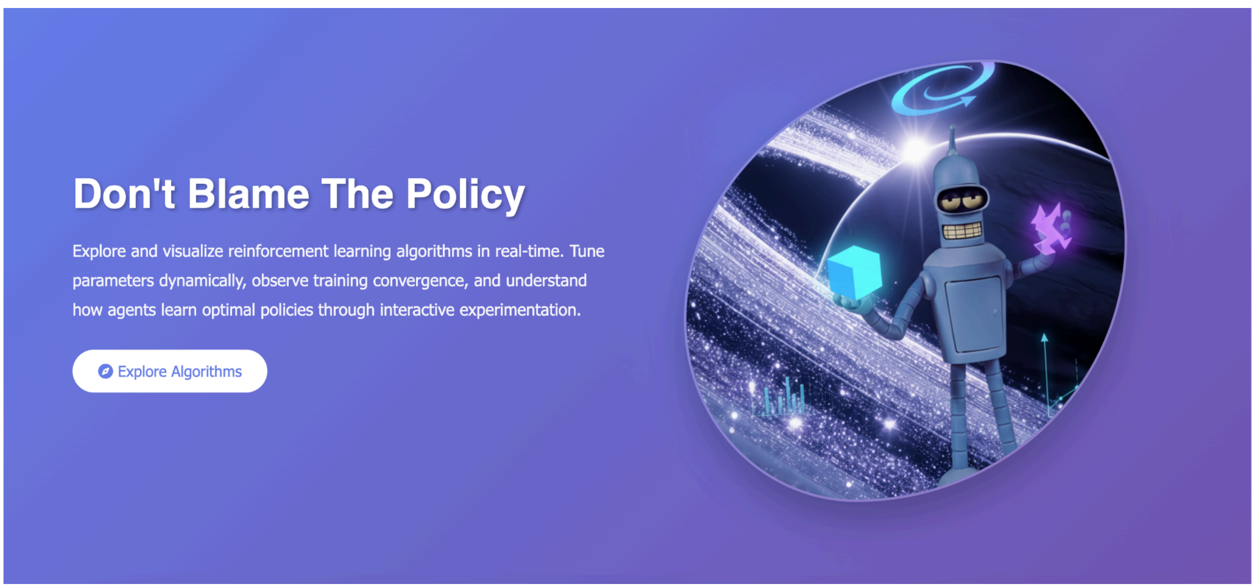

RL Web Tool Development Report

Project Overview

This project involved building an interactive web-based tool for learning reinforcement learning concepts. The tool provides hands-on experience with various RL algorithms across different environments, allowing users to visualize training processes, adjust parameters, and understand how different algorithms work in practice.

Implemented Environments

1. GridWorld

A classic grid navigation environment where an agent must move from start to goal while avoiding walls.

Key Features:
Configurable grid size (3x3 to 8x8)
Random wall generation with adjustable density
Visual representation of agent, goal, walls, and empty cells
Step-by-step agent movement visualization

Reward Structure:
Goal achievement: +10.0
Each step: -0.1
Wall collision: -1.0

2. FrozenLake

A slippery navigation challenge where an agent must cross a frozen lake to reach a goal while avoiding holes.

Key Features:

Configurable lake size (4x4 to 8x8)
Slippery surface option (33% chance of unintended movement)
Random hole generation with path preservation
Real-time agent animation

Reward Structure:
Goal achievement: +10.0
Falling in hole: -10.0
Each step: -0.1

3. CliffWalking

A risk-reward navigation environment where an agent must reach a goal while avoiding falling off cliffs.

Key Features:

Configurable dimensions (small: 4x12, medium: 5x15, large: 6x20)
Cliff region along the bottom edge
Visual cliff representation
Reset to start after cliff fall

Reward Structure:
Goal achievement: +10.0
Cliff fall: -100.0
Each step: -1.0

Implemented Algorithms

1. Value Iteration

A dynamic programming algorithm that computes optimal value functions through iterative Bellman updates.

Implementation Details:

Handles both deterministic and stochastic transitions
Proper stochastic transition modeling for FrozenLake slippery mode

2. Policy Iteration
An iterative DP algorithm alternating between policy evaluation and policy improvement.

Implementation Details:
Initial random policy generation

Policy evaluation with iterative value updates

Policy improvement through greedy action selection

3. Q-Learning
An off-policy temporal difference algorithm learning optimal action-value function.
Implementation Details:
Epsilon-greedy exploration strategy
Online Q-value updates using maximum next state value
Learning rate (alpha) controls update magnitude
Discount factor (gamma) for future reward importance

4. SARSA
An on-policy TD algorithm learning action-value function for current policy.
Implementation Details:
Uses actual next action in updates (on-policy)
Epsilon-greedy action selection
Continuous policy improvement
Suitable for stochastic environments

5. Monte Carlo Methods
Model-free learning from complete episodes without environment model.
Implementation Details:
First-visit MC for efficient learning
Episode-based value updates
Exploration control through epsilon parameter

Works well with both deterministic and stochastic environments

6. Temporal Difference Learning
General TD learning framework implemented as SARSA variant.

7. N-step TD
Multi-step TD learning balancing MC and TD approaches.

Parameter Adjustment Capabilities

1. Algorithm Parameters
For TD Algorithms (Q-Learning, SARSA, TD, N-step TD):
Learning Rate (α): Controls update magnitude (0.01 to 1.0)
Discount Factor (γ): Future reward importance (0.5 to 0.999)
Exploration Rate : Probability of random exploration (0.01 to 0.5)

Episodes: Training duration (100 to 5000)

2. Environment Parameters
GridWorld:

Grid Size: 3 to 8
Wall Density: 0 to 0.3
Random Walls: On/Off

FrozenLake:

Lake Size: 4 to 8

Slippery Surface: On/Off

Hole Probability: 0.1 to 0.3

CliffWalking:

Environment Size: Small/Medium/Large

Visualization Features
1. Real-time Training Visualization
Live chart showing reward/convergence over episodes/iterations
Color-coded progress indication

Algorithm-specific visualization (rewards for TD, delta for DP)

2. Environment Visualization
Grid-based visual representation
Color coding for different cell types
Agent animation during inference

3. Learned Policy Visualization
Action arrows (↑↓←→) showing optimal actions
Color gradient for value function heatmap

4. Value Function Visualization
Heatmap coloring based on state values
Numerical value display in each cell
Color legend indicating value range