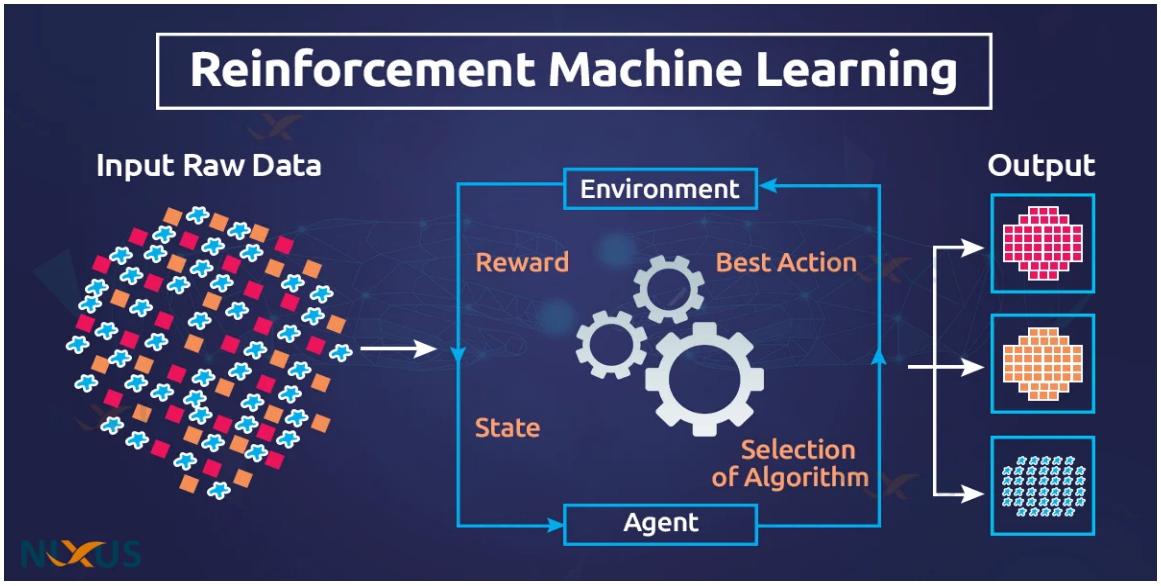
QUAID-I-AZAM UNIVERSITY
ISLAMABAD

Reinforcement Learning
Approaches for the Volcano Crossing Problem

Muhammad Bilal (04072113022)
Ali Ahmad Mumtaz (04072113023)



Environment Description

The reinforcement learning task is modeled using the `VolcanoCrossingEnv` environment. This environment represents a grid-world scenario where an agent navigates from a starting position to an end position while avoiding volcanoes. Key features of the environment include:

Grid Size

The grid is defined with a default size of 5×5, but it can be adjusted based on the problem instance.

Start and End Positions

The agent starts at a specified position (default: top-left corner) and aims to reach another position (default: bottom-right corner).

Volcanoes

Certain grid cells are designated as volcanoes, and the agent receives a substantial penalty if it steps on them.

Slip Probability

There is a configurable slip probability that introduces randomness into the agent's actions, simulating slippery steps.

State Space

The state space is represented by the agent's current position (x, y) on the grid.

Action Space

The agent has four possible actions:

1. Move Up
2. Move Down
3. Move Left
4. Move Right

Rewards

Default Reward

-1 is the default reward for each step.

Penalty for Volcanoes

Stepping on a volcano incurs a penalty of -100.

Reward for Reaching the End

The agent receives a reward of 100 for reaching the end position.

Agent Class

__init__ Method

def __init__(self):

- Initializes the agent with default parameters such as grid size, slip probability, and epsilon.
- Sets up the environment (VolcanoCrossingEnv), action space, state space, Q-values, returns, visited states, policy, rewards, and memory.
- Initializes Q-values, returns, and visited counts for each state-action pair.
- Initializes a random policy for each state.

get_state Method

def get_state(self):

- Returns the current state of the environment.

max_action Method

def max_action(self, state):

- Takes a state as input and returns the action with the highest Q-value for that state.

update_Q Method

def update_Q(self, state, action, reward, next_state):

- Updates the Q-value of a state-action pair using the Q-learning update rule.

update_Q_sarsa Method

def update_Q_sarsa(self, state, action, reward, next_state, next_action):

- Updates the Q-value of a state-action pair using the SARSA (State-Action-Reward-State-Action) update rule.

update_Q_monte_carlo Method

def update_Q_monte_carlo(self, episode_data):

- Updates Q-values based on the Monte Carlo method. It iterates over the reversed episode and updates Q-values for each state-action pair.

choose_action Method

def choose_action(self, state, epsilon=0.1):

- Implements an epsilon-greedy strategy for action selection. With probability epsilon, it explores by choosing a random action; otherwise, it exploits by choosing the action with the maximum Q-value.

Reinforcement Learning Algorithms

In this project, three reinforcement learning algorithms—Model-Free Monte Carlo, SARSA, and Q-Learning—were applied to the Volcano Crossing problem. The experiments were conducted with varying numbers of episodes to observe the learning dynamics under a uniformly random policy. Additionally, the algorithms were tested with different slip probabilities ranging from 0.0 to 0.3 to assess their robustness in the face of environmental uncertainty.

SARSA Experiment Report

Overview

The SARSA algorithm was applied to the Volcano Crossing problem, evaluating its performance with a slip probability of 0.1, epsilon of 0.1, and 5000 episodes.

Key Results

Initial Q-Values (Episode 0)

State (0, 0), Action 0	State (1, 1), Action 2
-0.75	-0.7375

Final Q-Values (Episode 5000)

State (0, 0), Action 0	State (1, 1), Action 2
56.79	58.05

Average Reward Progression

Episode	Average Reward
0	-20.0
1000	91.80
2500	91.84
5000	92.48

Analysis

The SARSA algorithm demonstrated effective learning, with Q-values showing substantial improvement over episodes. The average reward increased from -20.0 to a stable 92.48, indicating successful adaptation to the Volcano Crossing environment. The algorithm showcased robustness in handling slip probability and balancing exploration-exploitation dynamics. Overall, SARSA proved efficient in learning optimal policies for complex reinforcement learning tasks

Q-Learning Algorithm Experiment Report

Overview

The Q-Learning algorithm was implemented on the Volcano Crossing problem using the specified parameters: slip probability of 0.1, epsilon of 0.1, and 5000 episodes.

Analysis

Q-Learning displayed consistent improvement in Q-values over episodes, converging to optimal values. The average reward steadily increased, reaching 92.24 by the end of 5000 episodes. The algorithm effectively balanced exploration-exploitation dynamics, showcasing its capability to learn and adapt to the Volcano Crossing environment. Overall, Q-Learning demonstrated robust performance and successful convergence to optimal policies.

Key Results

Initial Q-Values (Episode 0)

State (0, 0), Action 0	State (1, 1), Action 2
55.59	57.42

Final Q-Values (Episode 5000)

State (0, 0), Action 0	State (1, 1), Action 2
59.61	63.80

Average Reward Progression

Episode	Average Reward
0	92.0
1000	92.05
2500	92.15
5000	92.24

Model-Free Monte Carlo Experiment Report

Overview

The Model-Free Monte Carlo algorithm was applied to the Volcano Crossing problem using the specified parameters: slip probability of 0.3, epsilon of 0.1, and 1000 episodes.

Key Results

Average Reward Progression

Episode	Average Reward
0	-17,765,555.0
100	-273,736.56
200	-183,983.79
300	-302,809.97

Analysis

Model-Free Monte Carlo exhibited a unique learning trajectory. Despite a highly negative initial average reward, the algorithm displayed a notable improvement over episodes. The average reward fluctuated, suggesting the exploration of different strategies. It is essential to note that this algorithm may have high variance due to its reliance on episode rollouts.

While the initial negative rewards might indicate a challenging learning process, the subsequent improvement suggests adaptability and learning from experiences. Further analysis, especially in terms of convergence and stability, would be beneficial to assess the algorithm's overall performance.

In summary, Model-Free Monte Carlo demonstrated a dynamic learning process, with the average reward showing improvement over the course of 1000 episodes. Further investigation and fine-tuning may be required to optimize its performance for the Volcano Crossing environment.

Graph

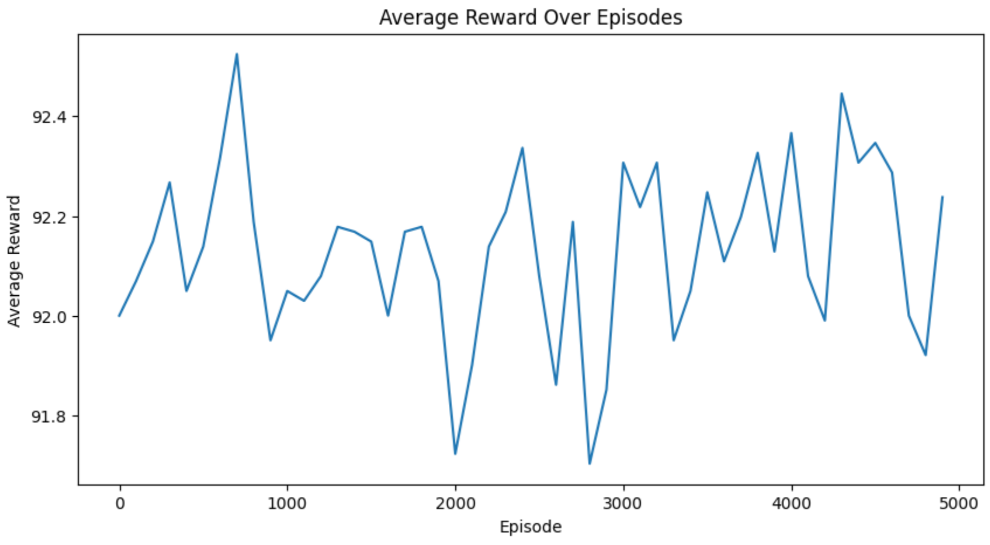


Figure 1: Q-Learning Graph

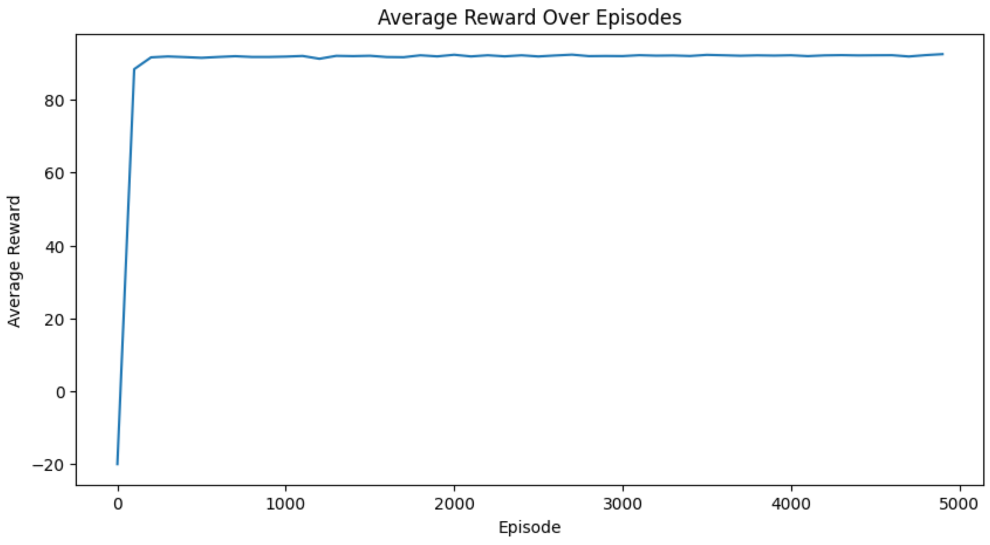


Figure 2: Sarsa Graph