1/14/2024

REINFORCEMENT LEARNING ANALYSIS REPORT

**SUBMITTED TO:**

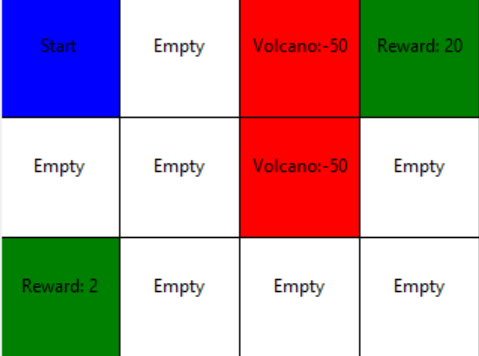
**Dr. AYYAZ HUSSAIN**

**SUBMITTED BY:**

**AMNA MUZAFFAR**

**PROBLEM DEFINITION:**

In volcano crossing problem we have a grid with different states and different states are assigned different positions like volcanos , reward , start position and we have to reach the reward state while avoiding the volcanoes. It is a Markov decision process where agent will move across different states and receive reward or penalties based on which state it has moved to. Parameters used are number of episodes, slip probability, and epsilon.

The Volcano Crossing environment is a 3x4 grid world where an agent starts at position (0, 0). The goal is to navigate to rewarding states (2, 0) and (0, 3) while avoiding volcanic states (0, 2) and (1, 2). Movement actions include "up," "down," "left," and "right," each leading to a new state. Successful moves yield a reward of 1, reaching volcanic states incurs a penalty of -50, and reaching rewarding states results in rewards of 2 and 20, respectively. The environment operates under a discounted factor (gamma) of 0.9 and a learning rate (alpha) of 0.1.

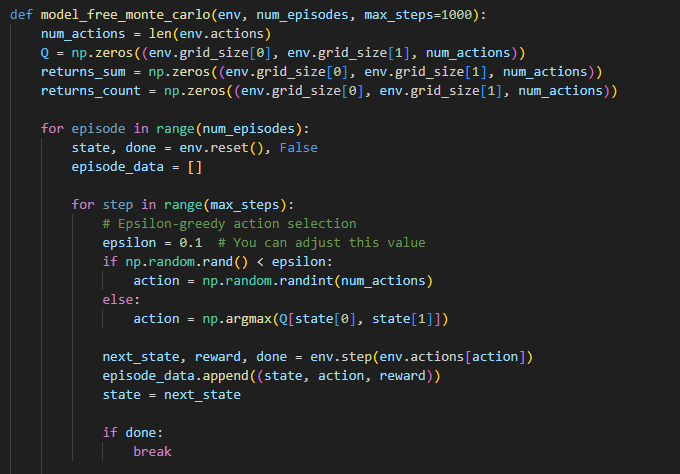
**USER INTERFACE:**

There is a GUI from where user can set the values of parameters like number of episodes, slip probability, and epsilon for each algorithm. user can run each algorithm separately or all algorithm at once from interface.

**ALGORITHM IMPLEMENTATION:**

1. **MODEL FREE MONTE CARLO:**

The model free Monte Carlo is implemented to learn the optimal policy by sampling episodes. it estimate the state action values through random transition of sates in grid environment using any random policy and perform that action and get next state and reward and update q values accordingly. With increase in number of episodes the average utility improved gradually



A computer code on a black background

Description automatically generated

Model-Free Monte Carlo is a reinforcement learning algorithm that learns from sampled episodes without requiring a model of the environment. In the context of the Volcano Crossing Problem, it estimates the value of states by averaging the returns observed in multiple episodes.

**STEPS**

* **Initialization:**

Initialize the Q-table, which represents the estimated values of state-action pairs.

Initialize arrays to store the sum of returns and count of visits for each state-action pair.

* **Episodic Sampling:**

**For each episode:**

Start from the initial state.

Use an epsilon-greedy strategy to select actions, balancing exploration and exploitation.

Interact with the environment, collecting state, action, and reward information until the episode terminates.

* **Update Q-values:**

For each time step in the episode:

Calculate the return (cumulative reward from that time step onward).

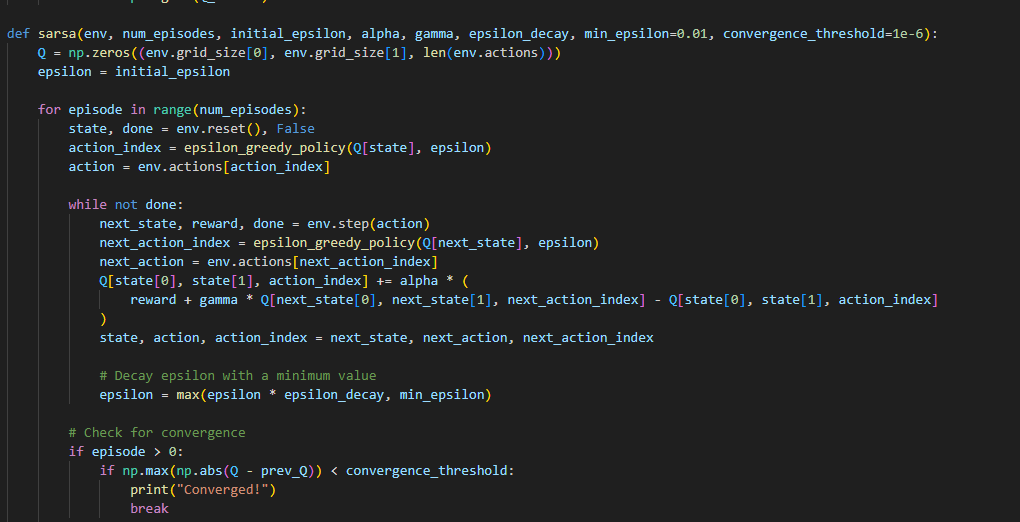
Update the Q-value for the state-action pair, averaging over all occurrences.

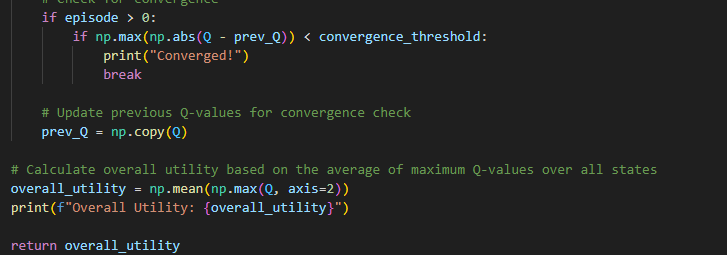
* **Overall Utility:**

The overall utility is calculated based on the maximum Q-value in the final state.

1. **SARSA (STATE-ACTION-REWARD-STATE-ACTION):**

SARSA uses on policy method, and it updates Q-values based on current policy based on agents sate transition. Actions are selected using epsilon greedy policy, and then execute action and observe next state and reward. Changing number of episodes shows convergence in Q-values.





SARSA is an on-policy temporal difference learning algorithm. It updates Q-values based on the current policy and uses an epsilon-greedy strategy for action selection.

**STEPS**

* **Initialization:**

Initialize the Q-table.

Set the exploration-exploitation parameters (epsilon, alpha, gamma).

* **Episodic Learning:**

For each episode:

Start from the initial state.

Choose an action using epsilon-greedy strategy.

Interact with the environment, observing the next state, reward, and next action.

Update the Q-value based on the observed information.

* **Epsilon-Greedy Policy:**

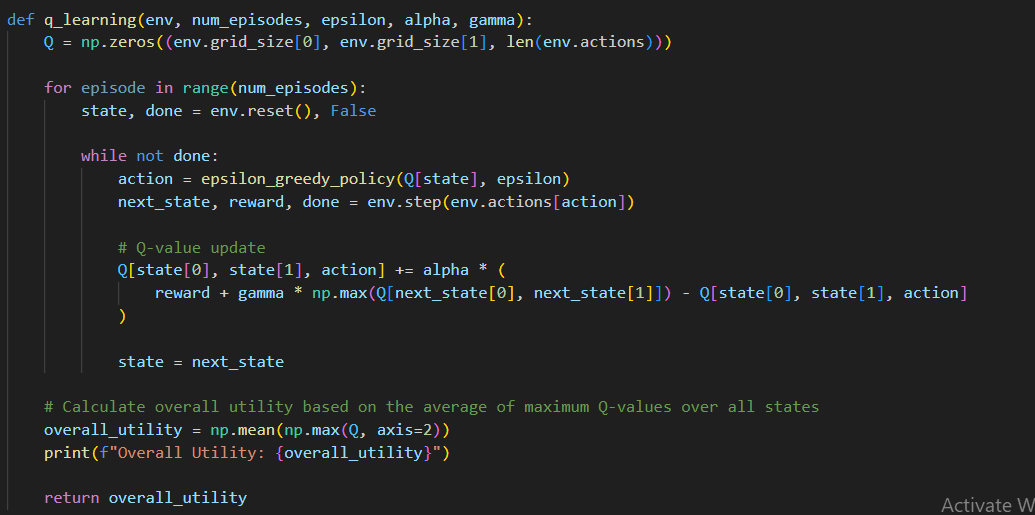
The epsilon-greedy policy is used to balance exploration (choosing a random action) and exploitation (choosing the best-known action).

* **Convergence Check:**

Optionally, a convergence check can be performed by monitoring changes in Q-values.

1. **Q-LEARNING:**

Q-Learning is off- policy and it updates values based on future reward and it is using epsilon greedy policy for exploration that will select the action to be taken like selecting initial state or initial action and then execute action and observe next state and reward.



Q-Learning is an off-policy temporal difference learning algorithm. It updates Q-values based on the optimal policy, regardless of the policy used for exploration.

**STEPS**

* **Initialization:**

Initialize the Q-table.

Set the exploration-exploitation parameters (epsilon, alpha, gamma).

* **Episodic Learning:**

For each episode:

Start from the initial state.

Choose an action using epsilon-greedy strategy.

Interact with the environment, observing the next state, reward, and taking the action that maximizes Q-values.

Update the Q-value based on the observed information.

* **Epsilon-Greedy Policy:**

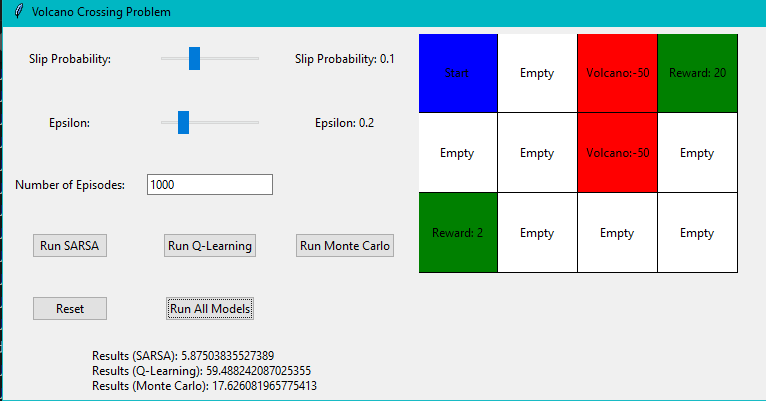
The epsilon-greedy policy is used for action selection during exploration.

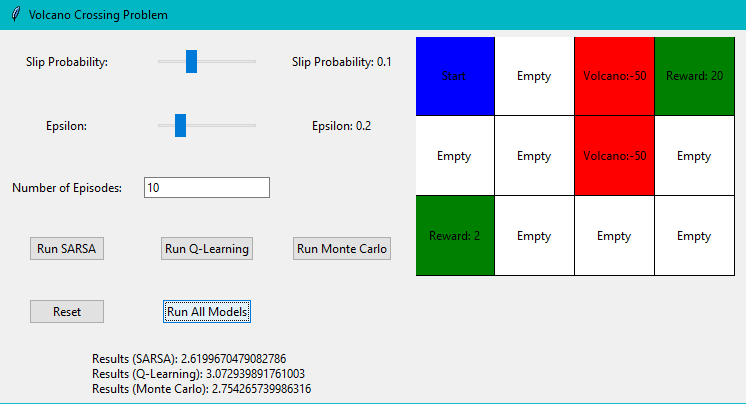
* **Overall Utility:**

The overall utility is calculated based on the average of maximum Q-values over all states.

**IMPACT OF PARAMETERS VARIATION:**

**CHANGING NUMBER OF EPISODES:**

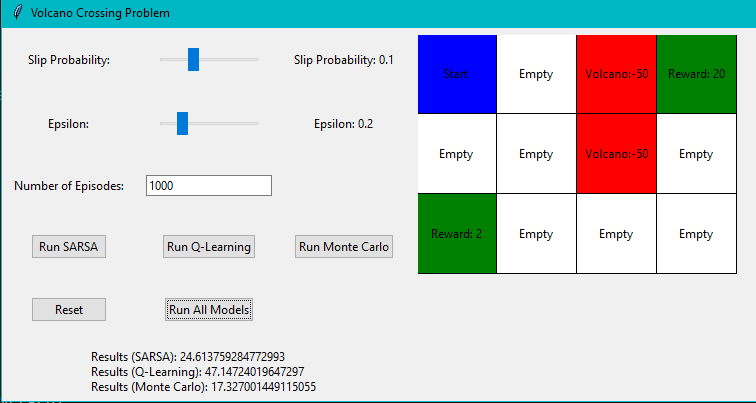


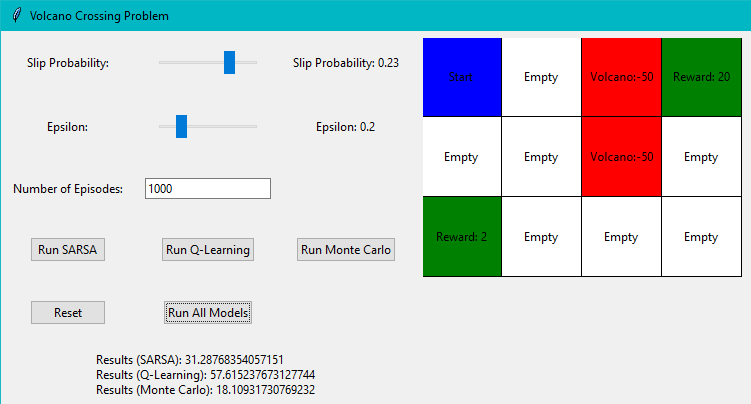


The number of episodes determine how many times agent interact with environment to learn and update policy. Increasing episodes allow agent to explore environment more which ultimately improves learned policy. As we can see in above figures increasing the value of episode increases the result value as well. So higher number of episodes leads to more accurate Q-values and estimates.

SARSA and Q-Learning are capable of updating their estimates after each action, allowing them to model the environment more effectively over time. Monte Carlo, on the other hand, relies on the completion of full episodes before updating Q-values. It estimates the expected return by sampling entire episodes. So, the increasing rate of monte Carlo result is slow than the SARSA and Q-Learning algorithm

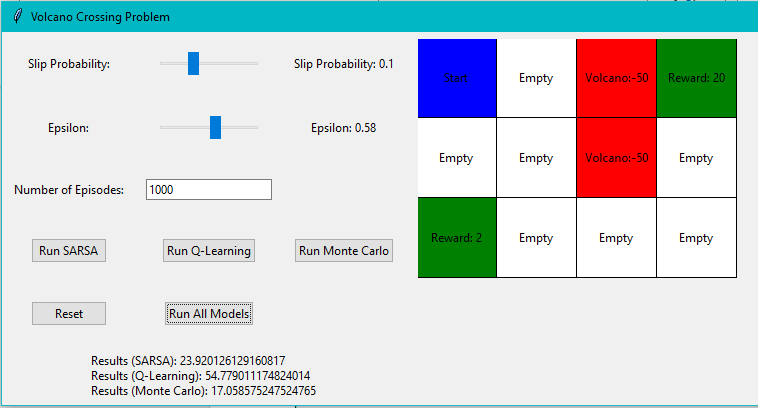
**CHANGING VALUE OF SLIP PROBABILITY:**

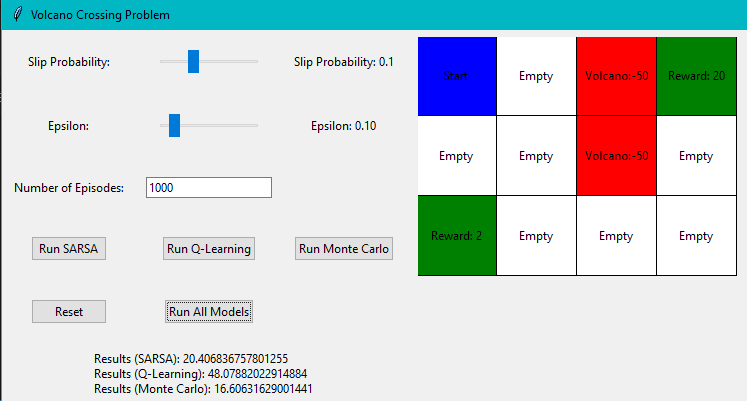




An increase in slip probability makes environment more challenging for agent as it has to explore more to understand the impact of certain action. Agent has to deal with more unpredictable variations in state transition as we can see in above figures, when we increase value of slip the result of all algorithm increases which means algorithm are generalizing policies very well and agent has become trained enough to make decisions when faced by unexpected slip events

**CHANGING VALUE OF EPSILON:**





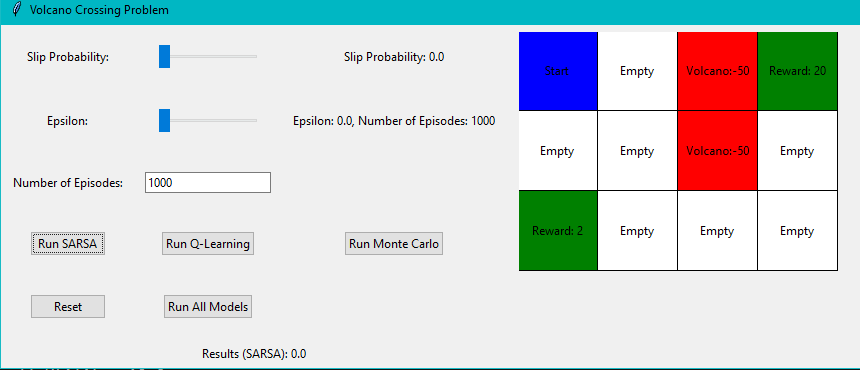
When value of epsilon increases, the probability of choosing a random action (exploration) also increases. This higher exploration rate encourages the agent to explore a broader range of actions and states, leading to a more comprehensive understanding of the environment.

When value of epsilon is increased, it means that the agent is more likely to choose exploration rather than exploitation. During exploration, the agent may encounter new state-action pairs, leading to updates in the Q-values for these pairs.

The Q-value updates are influenced by both immediate rewards and the estimated future rewards, allowing the agent to learn more about the environment.

**WHEN BOTH EPSILON AND SLIP ARE SET TO ZERO**

In SARSA, the epsilon parameter is associated with an epsilon-greedy policy used for action selection. When epsilon is set to zero, it means that the agent will always choose the action with the highest Q-value (pure exploitation) and will never explore random actions.



If both epsilon and slip are set to zero, the agent always exploits the current best action, and there is no randomness or slipping in the environment. In such a case, the agent effectively follows a deterministic and greedy strategy, always choosing the action with the highest Q-value.

**CONCLUSION**

In conclusion, the Volcano Crossing Problem and its solutions demonstrate the versatility and adaptability of reinforcement learning algorithms. The choice of algorithm parameters, especially slip probability and epsilon, significantly influences the learning process.

These algorithms iteratively update Q- values based on actions taken for moving to different positions. The parameters like number of episodes , slip probability, and epsilon highly impact the learning policy and the result it learned and ultimately impact the overall result

All three algorithms (Model-Free Monte Carlo, SARSA, and Q-Learning) were applied to the Volcano Crossing Problem. Model-Free Monte Carlo relies on Monte Carlo sampling and showed effectiveness in learning optimal policies. SARSA and Q-Learning, being temporal difference methods update Q-values and converge to optimal policies. The slip probability, epsilon (exploration parameter), and the number of episodes significantly impact algorithm performance. Higher epsilon values encouraged more exploration, potentially increasing Q-values. SARSA and Q-Learning algorithms included convergence checks to monitor changes in Q-values, ensuring stability. Model-Free Monte Carlo may not explicitly check for convergence due to its episodic nature but showed effectiveness in learning.