Reinforcement Learning (RBE595)

Programming Assignment 1

Implementation of Temporal Difference

Assignment By: Bhaavin Jogeshwar (WPI ID: 888602054)

Chinmayee Prabhakar (WPI ID: 101177306)

Swapneel Dhananjay Wagholikar (WPI ID: 257598983)

Introduction:

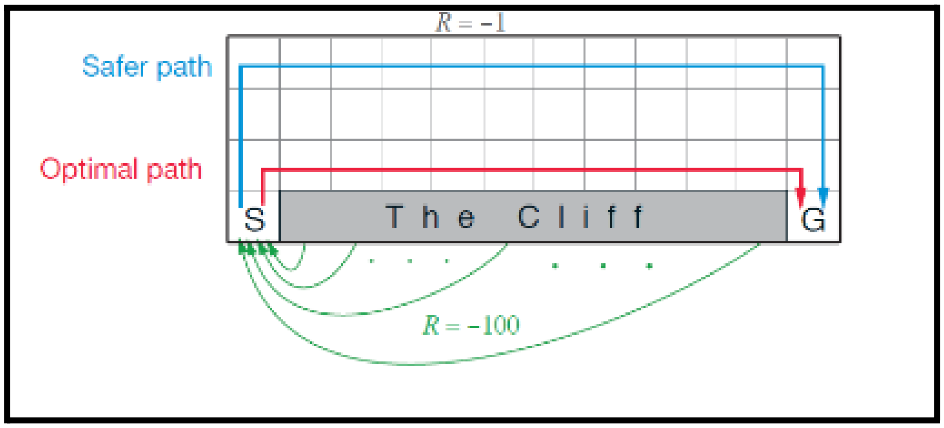


Fig 1: Cliff Walking Example

We are implementing two Temporal Difference (TD) policies, SARSA and Q Learning to solve the cliff walking example problem mentioned in the text book. Our algorithm has separate functions for the execution of the two policies. We have created a model of the map in a *.dat* file which is used for observing the 'States' that the robot traverses through. All possible actions that the robot can take are modelled as '*list_of_actions*' which allows the robot to move up, down, left and right under certain conditions(boundary states) at each state. Each episode execution starts from 'S', the start state and terminates when the robot reaches 'Goal' state. The reward in the case the robot enters the 'Cliff' state is '-100' and '-1' for every other transition. Over 500 episodes are run to observe the policy convergence for both SARSA and Q Learning. The cumulative rewards after each episode is plotted as a function of no. of episodes for both SARSA and Q Learning at different Epsilons. The best Action Policy at each state is symbolically plotted as well.

1. **SARSA (On-policy TD control):** SARSA (State-Action-Reward-State-Action) is a TD algorithm used to find an optimal policy for an agent to act in an unknown environment. In SARSA, the agent learns from its own experience by iteratively updating its Q-values based on the observed state, action, reward, and the next state and action pair. At each time step, the agent chooses an action using an epsilon-greedy policy based on its current Q-values, and then observes the resulting reward and next state. SARSA updates the Q-value for the current state-action pair based on the reward and the Q-value of the next state-action pair, weighted by a learning rate and a discount factor. SARSA due to the explorative nature of the epsilon-greedy policy tends

to form the less risk containing action policy, but under certain conditions such as at very low epsilon values, it can converge to an optimal policy.

Transition Function:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big].$$

Pseudo Code:

```
Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize $S$
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Loop for each step of episode:
        Take action $A$, observe $R$, $S'$
        Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until $S$ is terminal
```

2. **Q-Learning (Off-policy TD control):** Q-learning is a popular TD algorithm in reinforcement learning that is used to learn an optimal policy for an agent in an environment. Unlike SARSA, Q-learning does not continuously update its state-action value based on the immediate action it takes, but takes the best action that gives the maximum rewards for that state. This optimizes and reduces the faulty randomness that is introduced by the epsilon-greedy approach as seen in SARSA. The updates are weighted by a learning rate and a discount factor that balances the importance of current and future rewards. Q-learning always converges to an optimal policy as it only picks the state-policy that gives the maximum reward (value).

Transition Function:

$$\boxed{Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]}$$

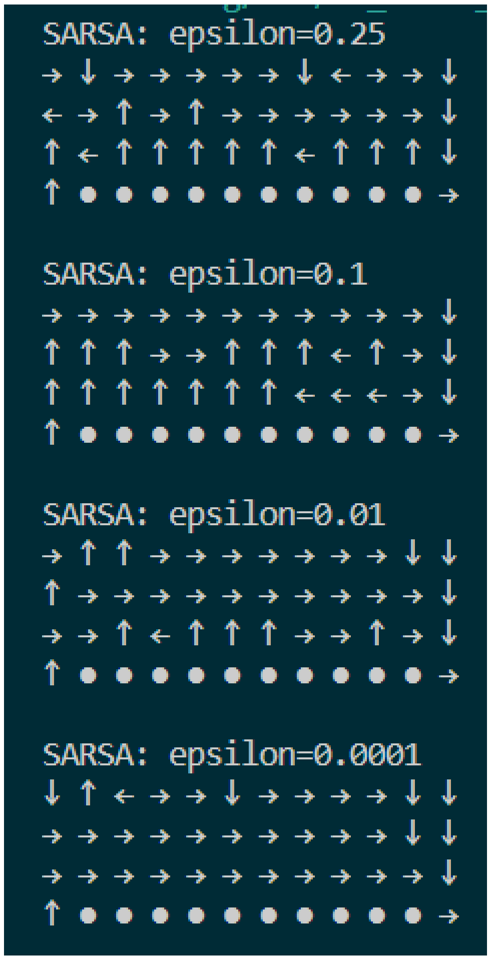
Pseudo Code:

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

```
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        Take action $A$, observe $R$, $S'$
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until $S$ is terminal
```

Results:

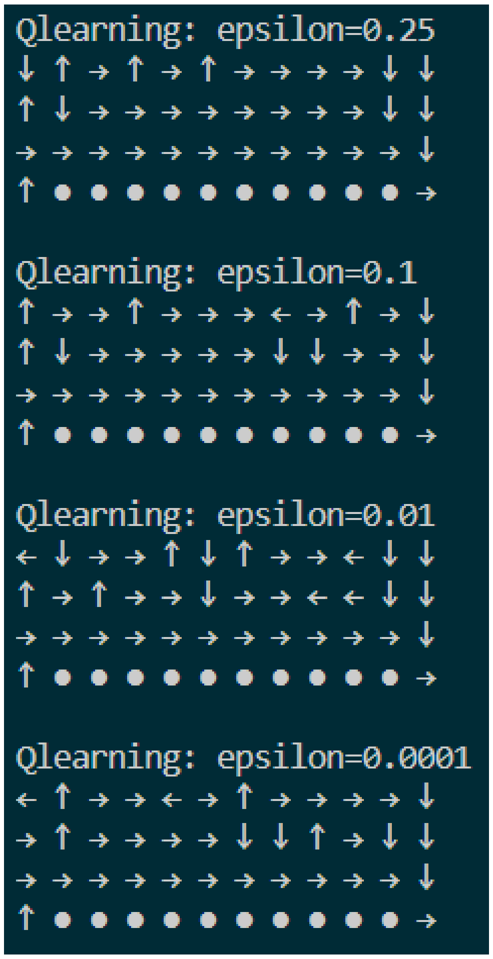


```
SARSA: epsilon=0.25
→ ↓ → → → → → ↓ ← → → ↓
← → ↑ → ↑ → → → → → → ↓
↑ ← ↑ ↑ ↑ ↑ ↑ ← ↑ ↑ ↑ ↓
↑ ● ● ● ● ● ● ● ● ● ● →

SARSA: epsilon=0.1
→ → → → → → → → → → → ↓
↑ ↑ ↑ → → ↑ ↑ ↑ ← ↑ → ↓
↑ ↑ ↑ ↑ ↑ ↑ ↑ ← ← ← → ↓
↑ ● ● ● ● ● ● ● ● ● ● →

SARSA: epsilon=0.01
→ ↑ ↑ → → → → → → → ↓ ↓
↑ → → → → → → → → → → ↓
→ → ↑ ← ↑ ↑ ↑ → → ↑ → ↓
↑ ● ● ● ● ● ● ● ● ● ● →

SARSA: epsilon=0.0001
↓ ↑ ← → → ↓ → → → → ↓ ↓
→ → → → → → → → → → ↓ ↓
→ → → → → → → → → → → ↓
↑ ● ● ● ● ● ● ● ● ● ● →
```



```
Qlearning: epsilon=0.25
↓ ↑ → ↑ → ↑ → → → → ↓ ↓
↑ ↓ → → → → → → → → ↓ ↓
→ → → → → → → → → → → ↓
↑ ● ● ● ● ● ● ● ● ● ● →

Qlearning: epsilon=0.1
↑ → → ↑ → → → ← → ↑ → ↓
↑ ↓ → → → → → ↓ ↓ → → ↓
→ → → → → → → → → → → ↓
↑ ● ● ● ● ● ● ● ● ● ● →

Qlearning: epsilon=0.01
← ↓ → → ↑ ↓ ↑ → → ← ↓ ↓
↑ → ↑ → → ↓ → → ← ← ↓ ↓
→ → → → → → → → → → → ↓
↑ ● ● ● ● ● ● ● ● ● ● →

Qlearning: epsilon=0.0001
← ↑ → → ← → ↑ → → → → ↓
→ ↑ → → → → ↓ ↓ ↑ → ↓ ↓
→ → → → → → → → → → → ↓
↑ ● ● ● ● ● ● ● ● ● ● →
```

Fig: Action policy shown under Q Learning and SARSA policies at different epsilons for each state.

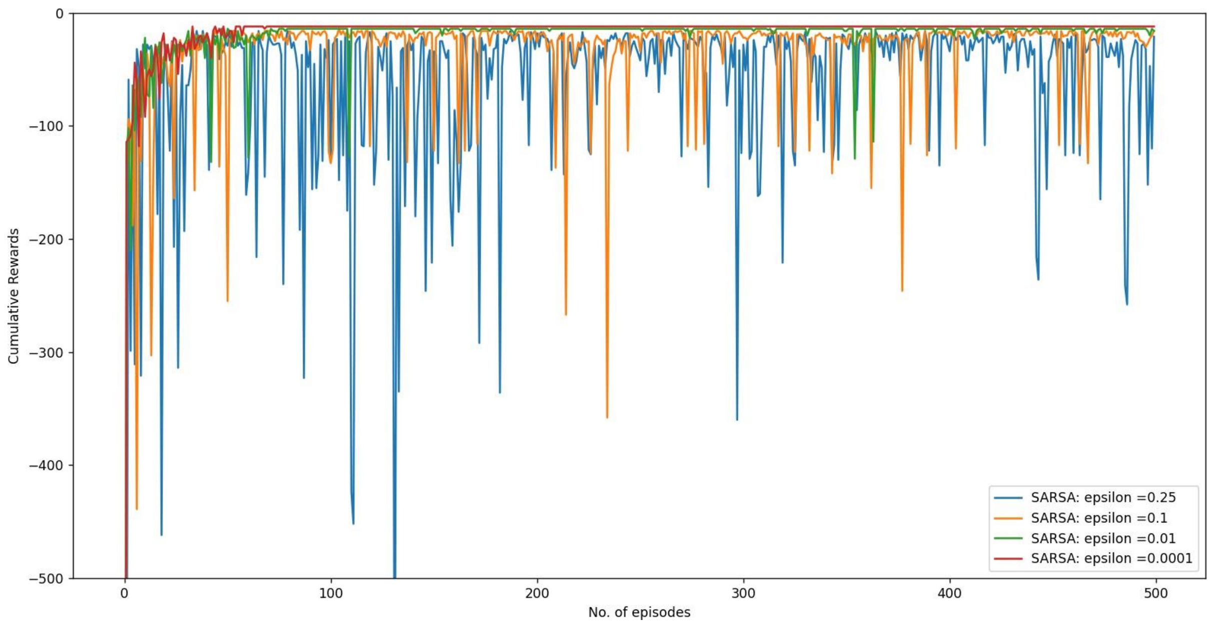


Fig: Cumulative rewards vs number of episodes for SARSA

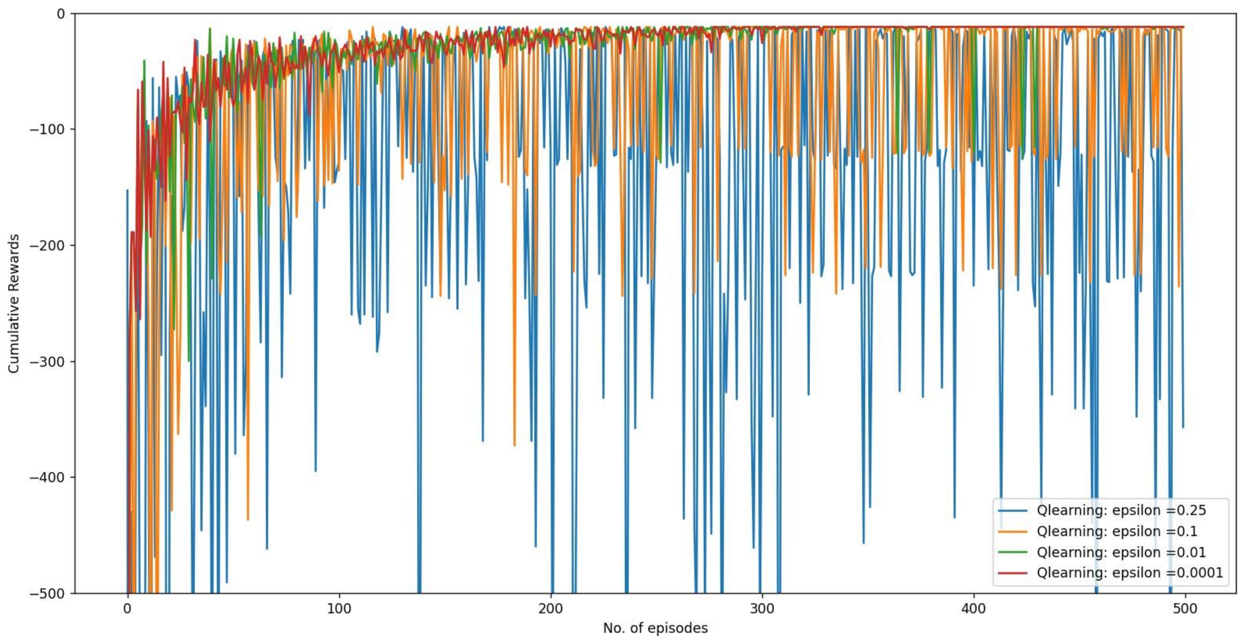


Fig: Cumulative rewards vs number of episodes for Qlearning

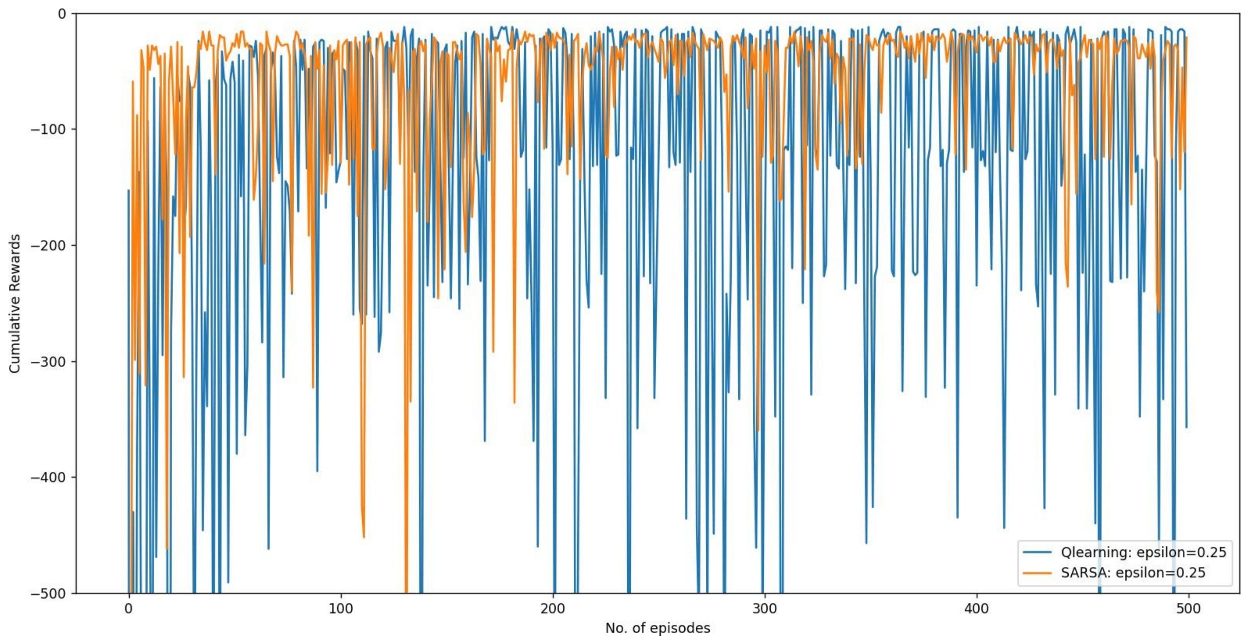


Fig: Graphically comparing the rewards of Qlearning and SARSA for a particular epsilon as a function of episodes.

Question 1: SARSA converges to the blue path and Q-learning converges to the red path (shortest path). Explain why is that?

The reason why Q-learning can converge to the optimal (shortest) path compared to SARSA is that Q-learning uses the maximum Q-value of the next state to update the Q-value of the current state-action pair, regardless of the action taken in the next state. This approach is known as the "greedy" approach, as the agent always selects the action with the highest Q-value. As a result, Q-learning tends to learn the optimal policy faster and more accurately, as it focuses on the best possible future reward. In contrast, SARSA is more conservative, as it updates the action-value function based on the current policy, which may not always select the optimal action.

Question 2: Explain why Q-leaning converges to lower average rewards even though it can find the optimal path?

Q-learning converges to lower average rewards because Q-learning can be prone to overestimating the value of certain state-action pairs, which can lead to suboptimal behavior and lower average rewards. This overestimation can occur because Q-learning updates the Q-values using the maximum Q-value of the next state, even if the selected action in the next state may not be the optimal action.

Furthermore, Q-learning can also be sensitive to the learning rate (i.e., step size) used in the update rule. A high learning rate can cause the algorithm to converge quickly, but it may also lead to instability and overestimation of certain state-action pairs.

Question 3: For both methods gradually reduce the ϵ (in epsilon-greedy) and show that both algorithms converge to optimal path and explain why.

Both Q-learning and SARSA can reach the optimal path when the value of epsilon is gradually decreased over time. A high value of epsilon allows the agent to explore the environment and find the rewards connected to various state-action pairings The algorithm effectively transforms into a purely greedy strategy, where the agent always chooses the action with the highest Q-value, as the value of epsilon gets closer to zero. This way SARSA essentially converges to the optimal policy similar to the Q-learning.