Reinforcement Learning Lab Report

Assignment 4

Gudepu Venkateswarlu (212011003)
Jayanth S (201081003)
Praveen Kumar N (201082001)
Rishabh Roy (201082002)

1 Algorithms analysed

- Q-learning
- SARSA Algorithm
- On-policy Monte-Carlo Control

2 Environments Considered

2.1 Classical Maze Problem

We considered the classical maze problem (Finite Markov Decision Process problem) and use value iteration and policy iteration to solve them. The maze environment consisted of *free cells* (including the start and terminal state) and *wall*. The agent/robot should find the shortest path from the start state to the terminal state such that it's total discounted reward is maximized. i.e.,

$$G = \sum_{t=0}^{\infty} \gamma^t R_t, \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor, R_t is the reward obtained at time instant t and G_T is the total discounted reward over the entire path from starting state to the terminal state.

The detailed problem setting is given as follows: In each state the agent can take one of the following actions :

1. Left ($<$)
2. Up ($\wedge$)
3. Right ($>$)
4. Down ($\vee$)

The reward structure is as follows:

- Movement to an adjacent state will cost = 0.1.
- Attempt to move towards a wall will cost = 0.75 and resulting state remains same as the starting state.

- Attempt to move towards boundary will cost = 0.8 and resulting state remains same as the starting state.
- Reaching the terminal state will result in a reward = 10.

2.2 Mountain Car

2.2.1 Environment Description

A car has to travel in a one-dimensional (horizontal) track with the road being similar to a valley between mountains on either side of it. The goal is to reach the top of right side mountain starting in the valley.

- *State space* : 2-D states representing (position, velocity)
- *Action space* :
  1. Push left (accelerate left)
  2. Push right (accelerate right)
  3. Do nothing
- *Reward* :
  - -1 for any action taken
  - 0 on reaching the goal

2.2.2 Additional Information of environment

- *Initial state* : $(x, 0)$ with x taking value between $[0, -0.4]$ uniformly.
- *Termination* :
  - On reaching the goal i.e. $x > 0.5$.
  - Else if length of episode is 200.

Also, $x \in [-1.2, 0.6]$, velocity $\in [-0.07, 0.07]$ and maximum speed is 0.07.

2.2.3 Discretization of the environment

Since mountain car is a continuous state space, we need to discretize the state-space to analyse the Monte-Carlo and Temporal Difference algorithms. Hence we used position bins of width 0.03 and velocity bins of width 0.005 resulting in $|\mathcal{S}| = 60 \times 28 = 1680$.

3 Observations

3.1 Classical Maze Problem

The Q-learning and SARSA algorithms converged at a faster rate to the optimal policy and close to optimal values for the given small maze problem. However we observed that monte-carlo every visit algorithm didn't converge to the optimal state values even after $3,00,000$ episodes however it was able to find a trajectory to reach destination from any point on the grid-world.

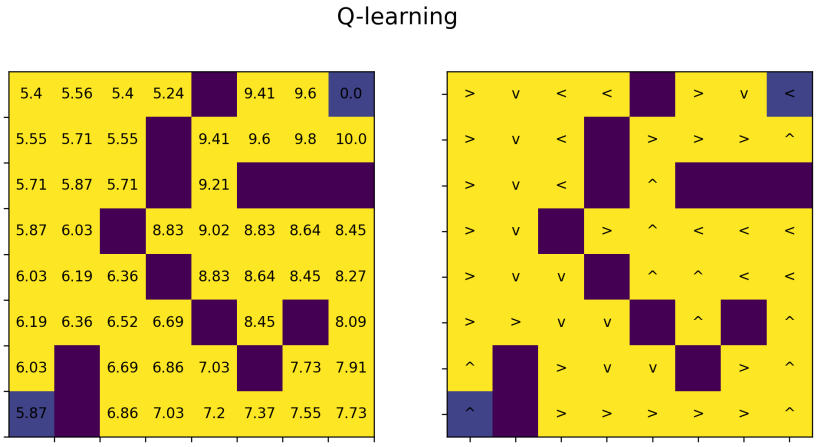

Figure 1: The value in each cell in left figure corresponds to the optimal state value for that cell. The right figure is the plot of optimal deterministic actions corresponding to each cell obtained using Q-learning algorithm. (Ignore the action of the terminal state)

3.2 Mountain Car Problem

- The Q-learning and SARSA algorithms worked well for the standard mountain car environment. The least number of steps taken was around 86/87 by the temporal difference algorithms. After running the algorithm for $1,50,000$ episodes, the running average return (averaged over last 100 episodes) was around $140 - 145$.
- However, the Monte-Carlo algorithm did not work even after $5,00,000$ episodes. Increasing the episode length to 500/800 also didn't work. Hence we performed reward reshaping as follows:
  - If $x > -0.4$ give additional reward of $e^{3(x+0.4)}$
  - If the goal is reached, i.e., $x \geq 0.5$ give an additional reward of 100.

4 References

- Deep Reinforcement Learning for Maze Solving
- Open.ai gym environments

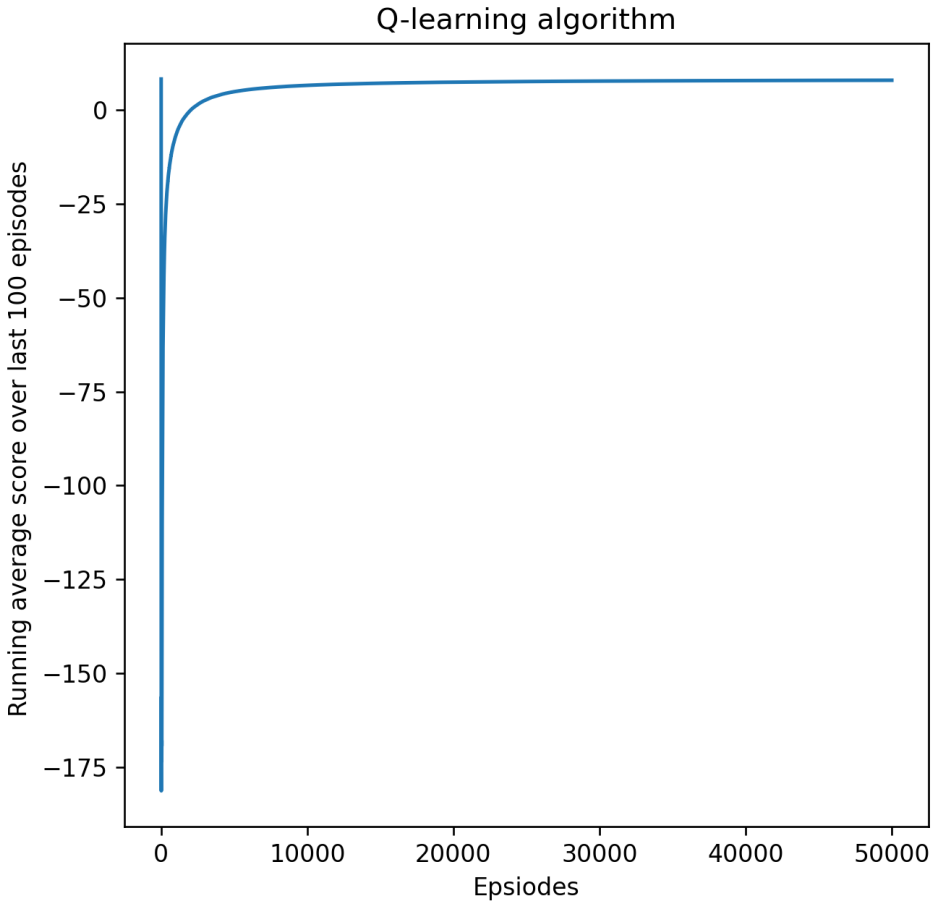

Figure 2: Running average of reward obtained for maze problem using Q-learning algorithm

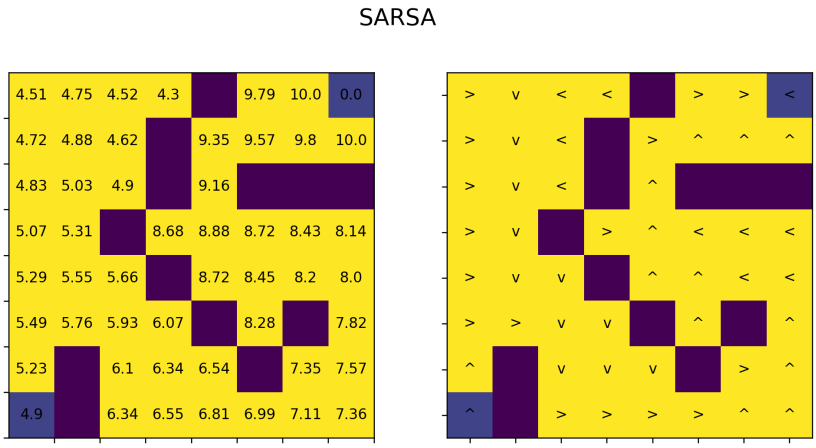

Figure 3: The value in each cell in left figure corresponds to the optimal state value for that cell. The right figure is the plot of optimal deterministic actions corresponding to each cell obtained using SARSA algorithm. (Ignore the action of the terminal state)

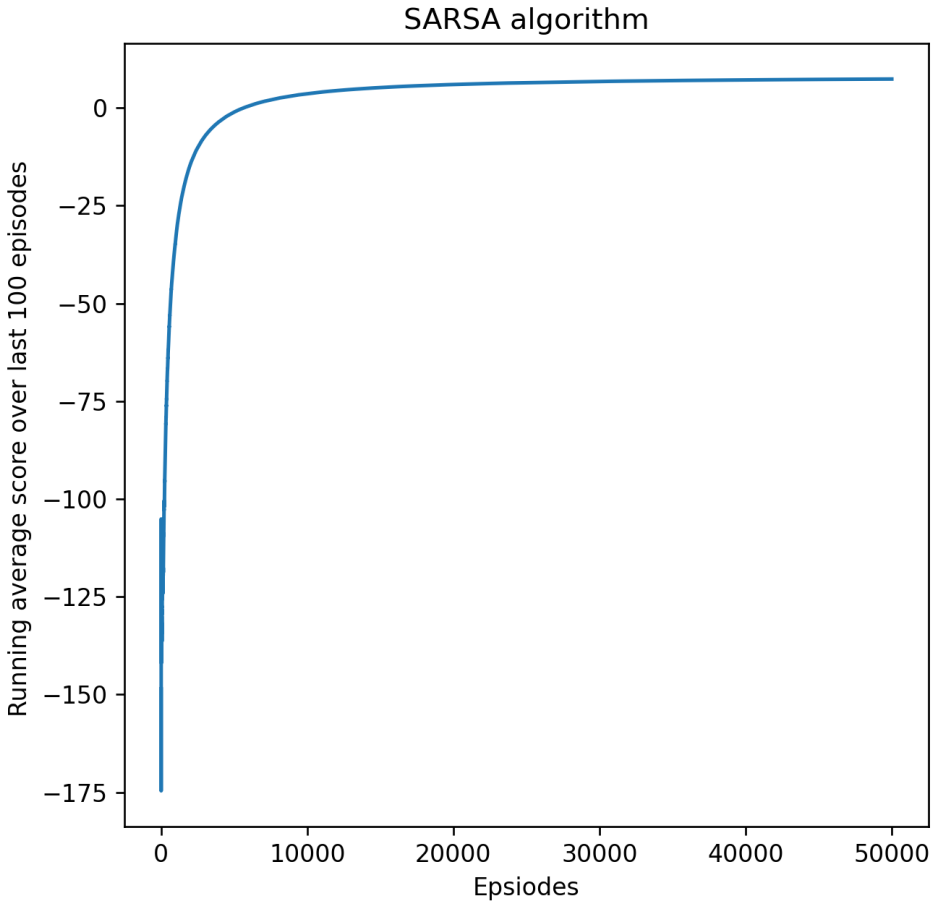

Figure 4: Running average of reward obtained for maze problem using SARSA algorithm

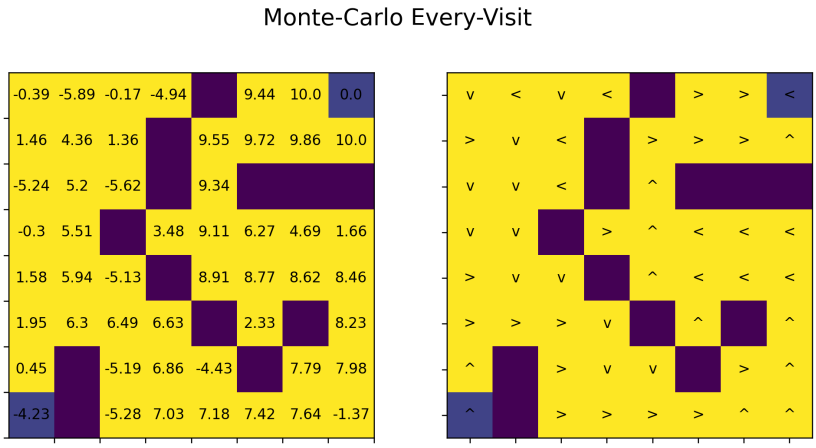

Figure 5: The value in each cell in left figure corresponds to the optimal state value for that cell. The right figure is the plot of optimal deterministic actions corresponding to each cell obtained using Every-Visit Monte-Carlo algorithm. (Ignore the action of the terminal state)

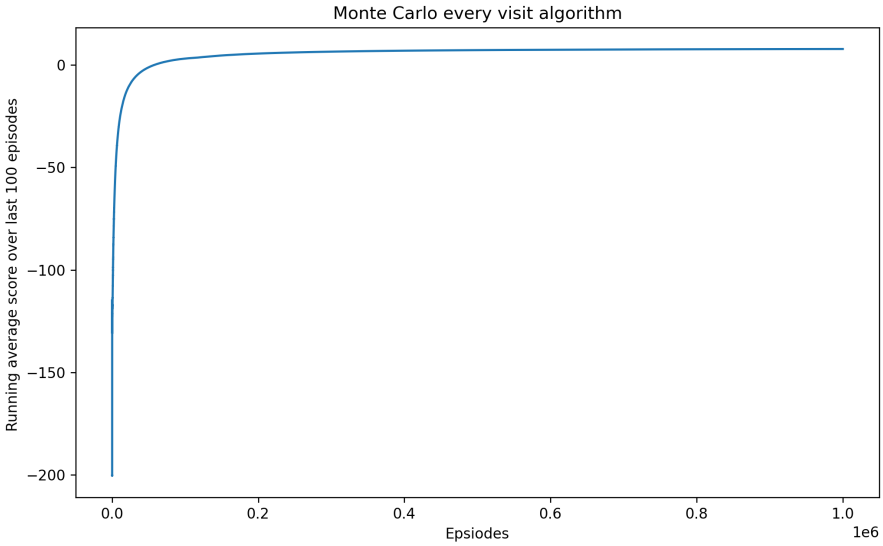

Figure 6: Running average of reward obtained for maze problem using Every-Visit Monte-Carlo algorithm

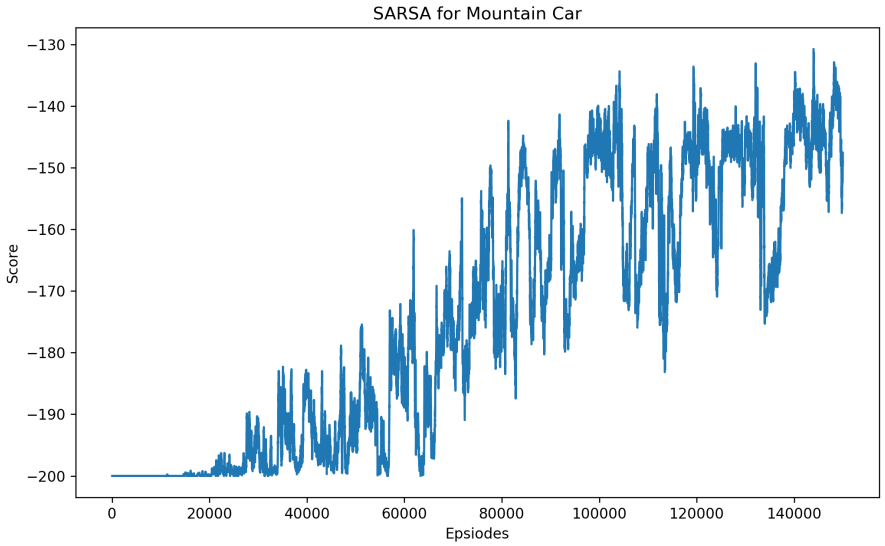

Figure 7: Running average of reward obtained for mountain car environment using SARSA algorithm

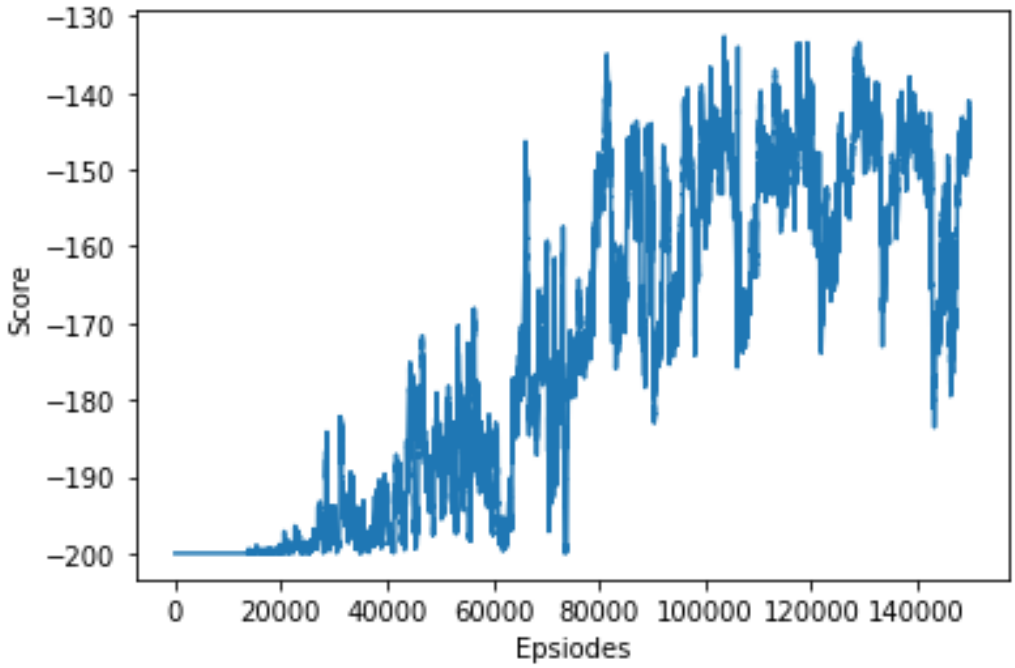

Figure 8: Running average of reward obtained for mountain car environment using Q-learning algorithm

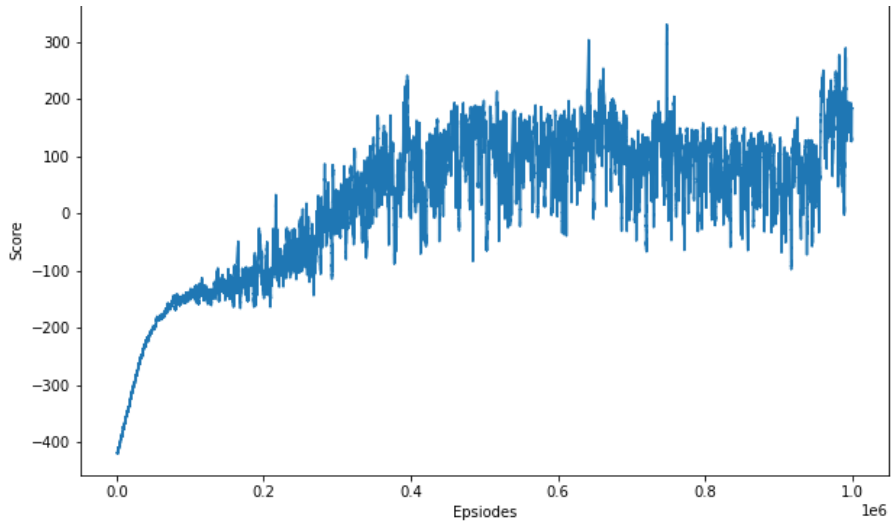

Figure 9: Running average (averaged over 1000 episodes) of reward obtained for mountain car environment using Monte-Carlo every visit algorithm

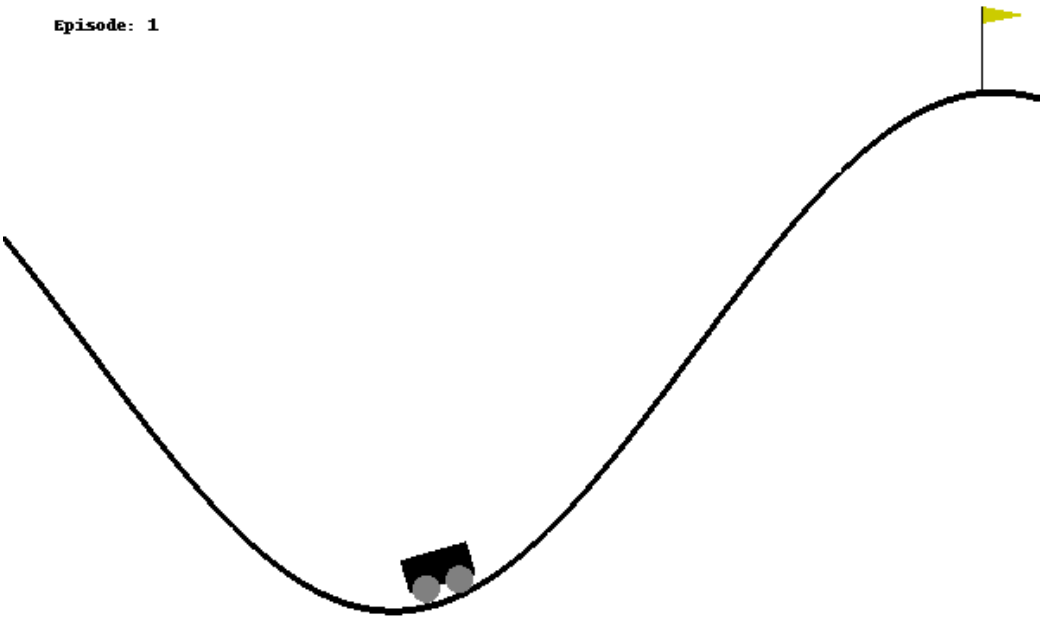

Figure 10: One episode in Mountain Car environment using the trained Monte Carlo every visit agent.