Programming Assignment 2

MM16B023[1]

[a]*Indian Institute of Technology Madras*

Keyword: QLearning, SARSA, Eligibility traces, Policy Gradient, OpenAI Gym

Abstract: This paper presents the solutions to the second programming assignment of Reinforcement Learning (CS6700) at *IIT Madras*. All the notations used are as according with the textbook Reinforcement Learning by S. Sutton and G. Barto

Problem 1

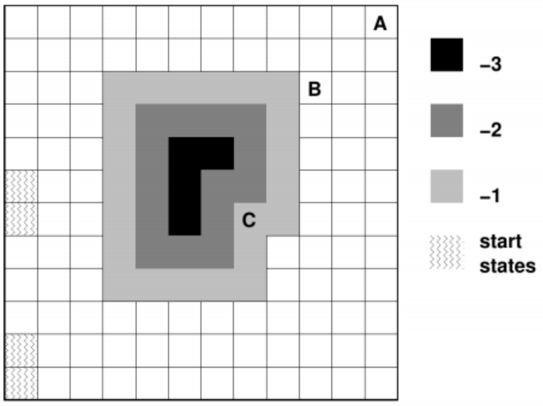
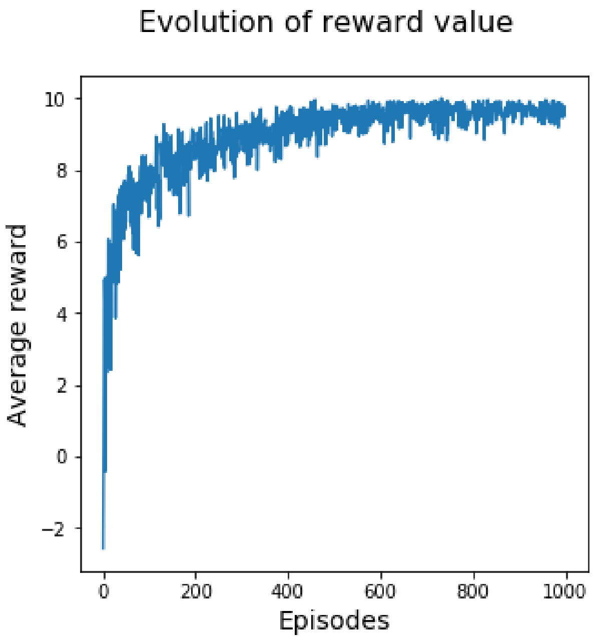


Figure 1: The puddle-world.

a) QLearning

Case A



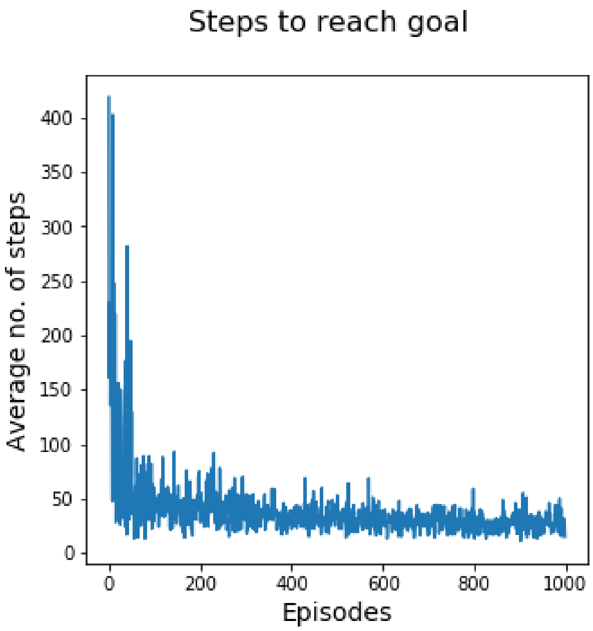


Fig 2: Performance of QLearning to achieve goal A

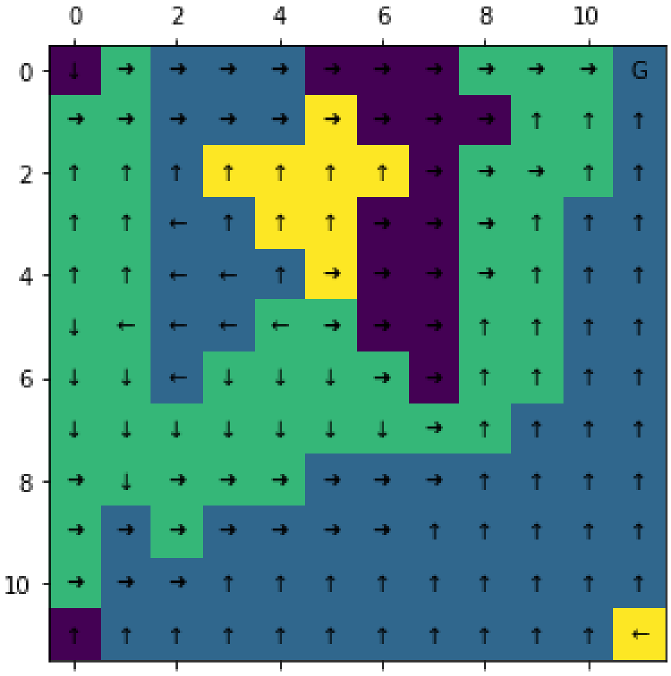


Fig 3: Optimal policy to arrive at A

Case B

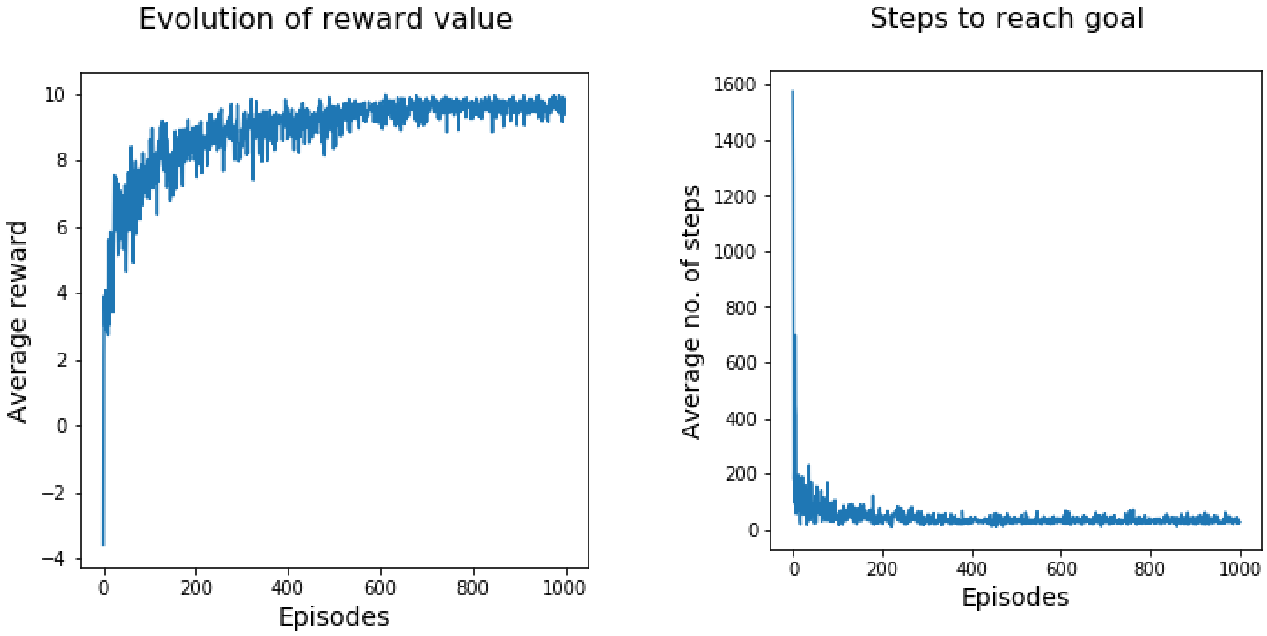


Fig 4: Performance of QLearning to achieve goal B

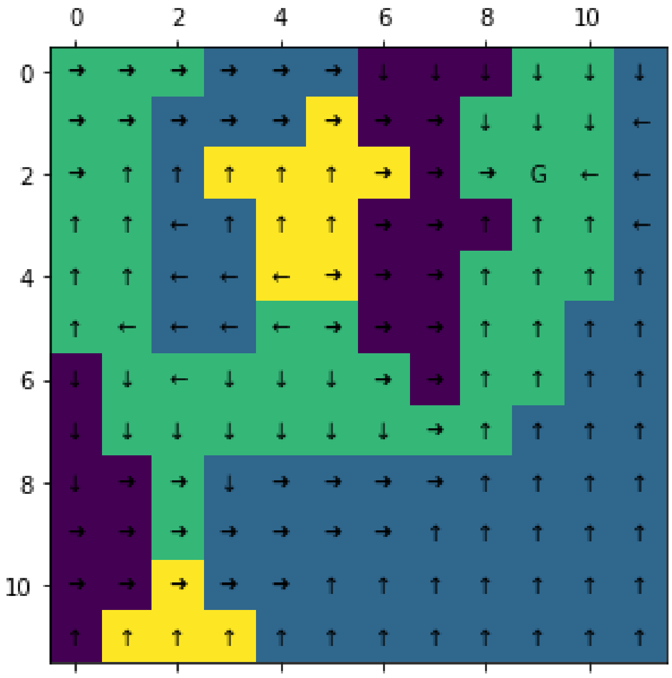


Fig 5: Optimal policy to arrive at B

Case C

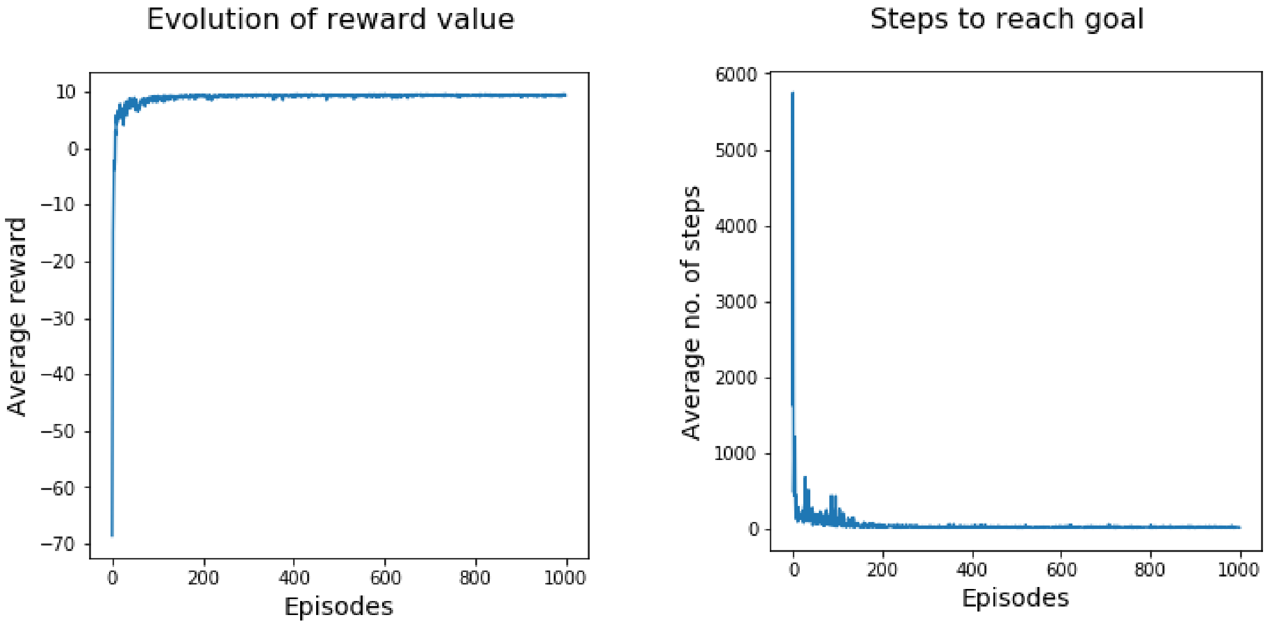


Fig 6: Performance of QLearning to achieve goal C

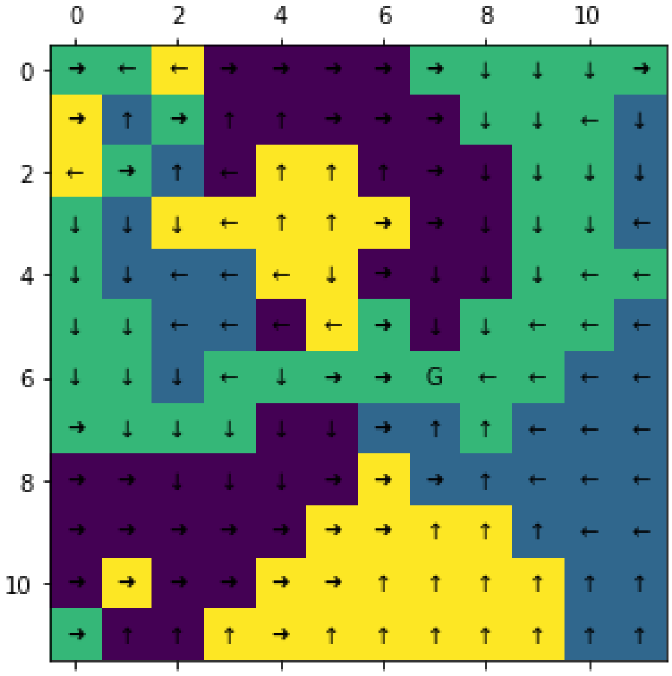


Fig 7: Optimal policy to arrive at C

Discussion

- Learning happens the most quickly in case of C being the target. This is expected, as the agent would want to quickly maximize the reward since it initially receives more negative rewards from the puddle as compared to the other cases.

b) SARSA

The algorithm ends up in a sub optimal policy which makes the agent move to the bottom right corner irrespective of the target. So, every time the agent is swaying away from the goal or the puddle, a penalty of -0.2 is given. This ensures that the agent doesn't get stuck in a local optima.

Case A

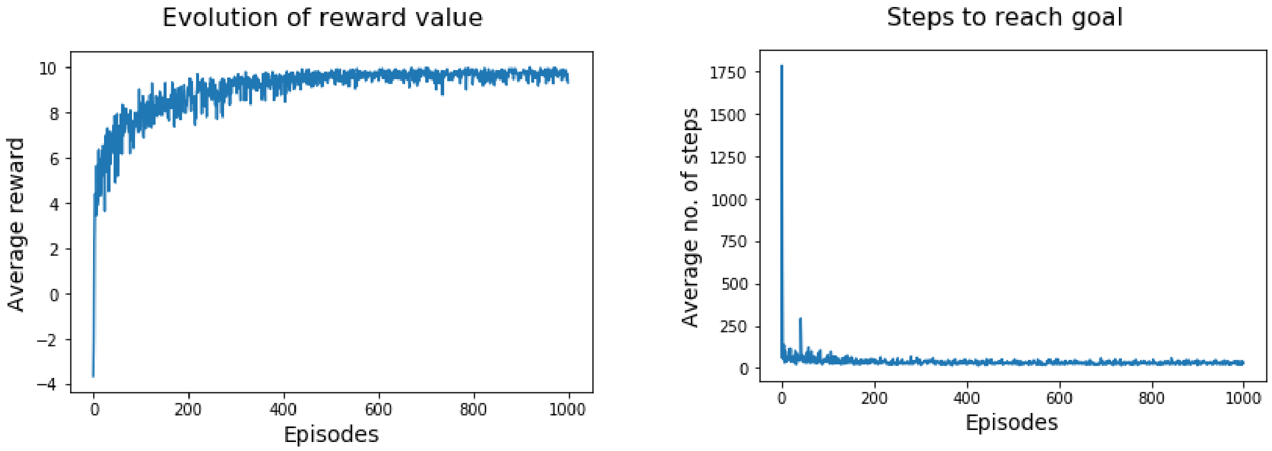


Fig 8: Performance of SARSA to achieve goal A

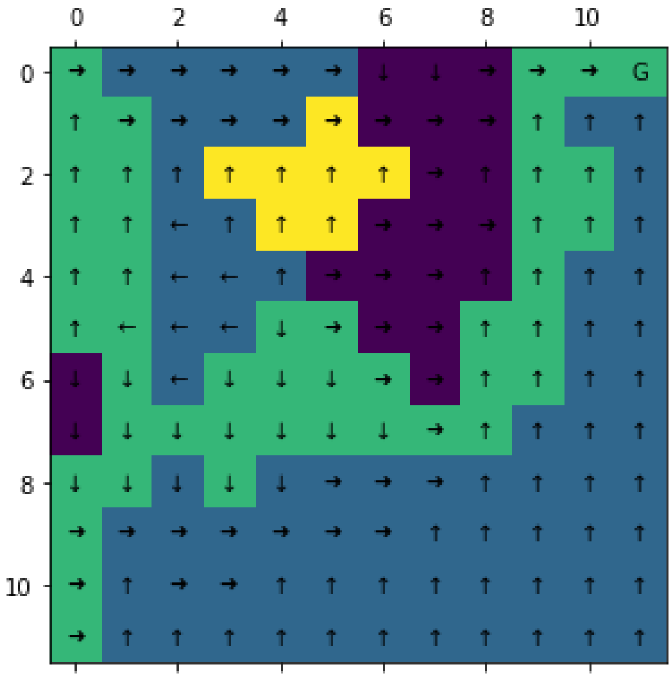


Fig 9: Optimal policy to arrive at A

Case B

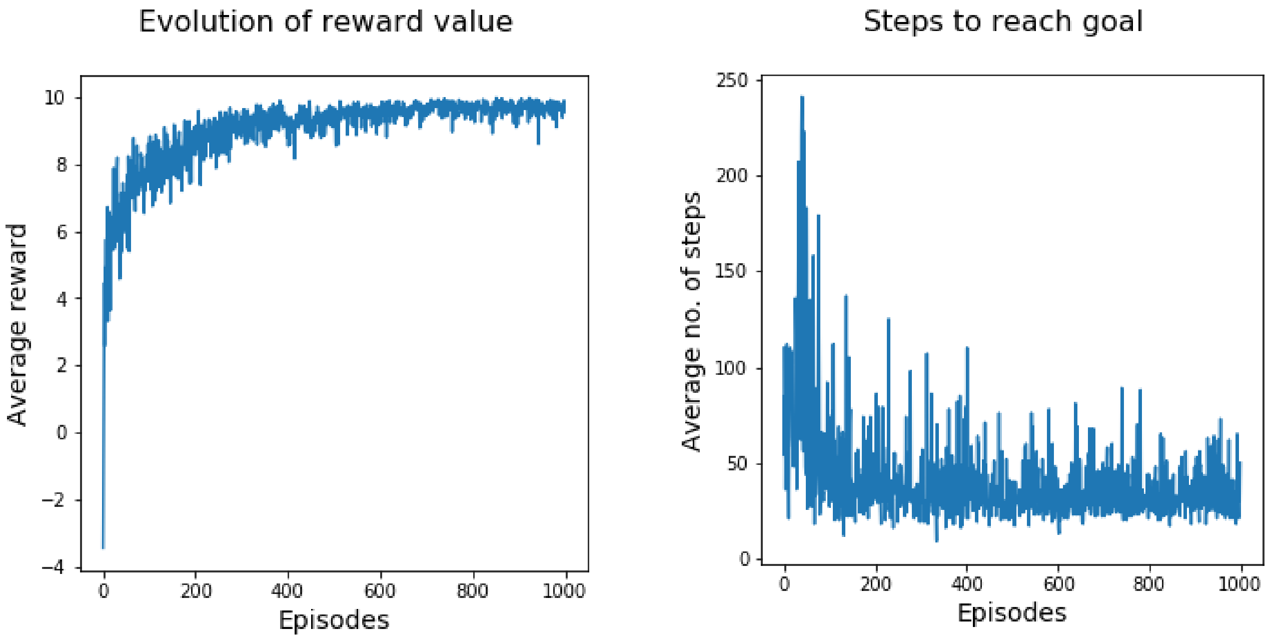


Fig 10: Performance of SARSA to achieve goal B

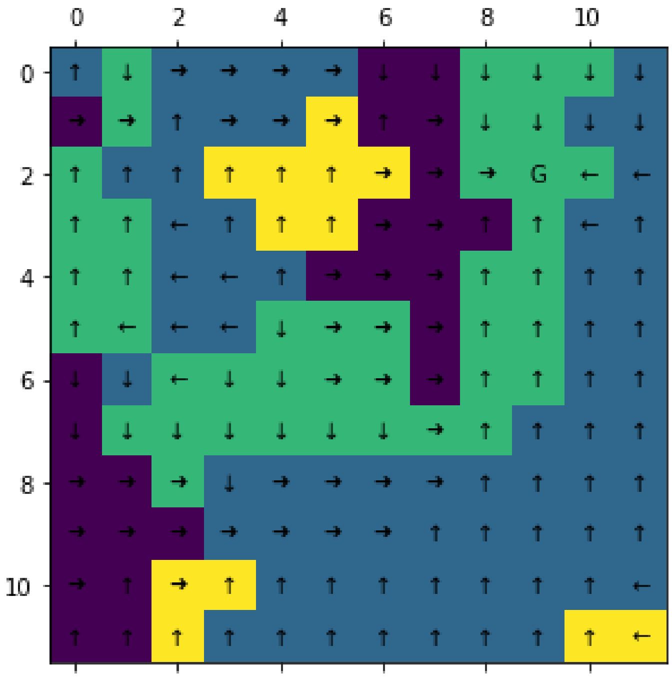


Fig 11: Optimal policy to arrive at B

Case C

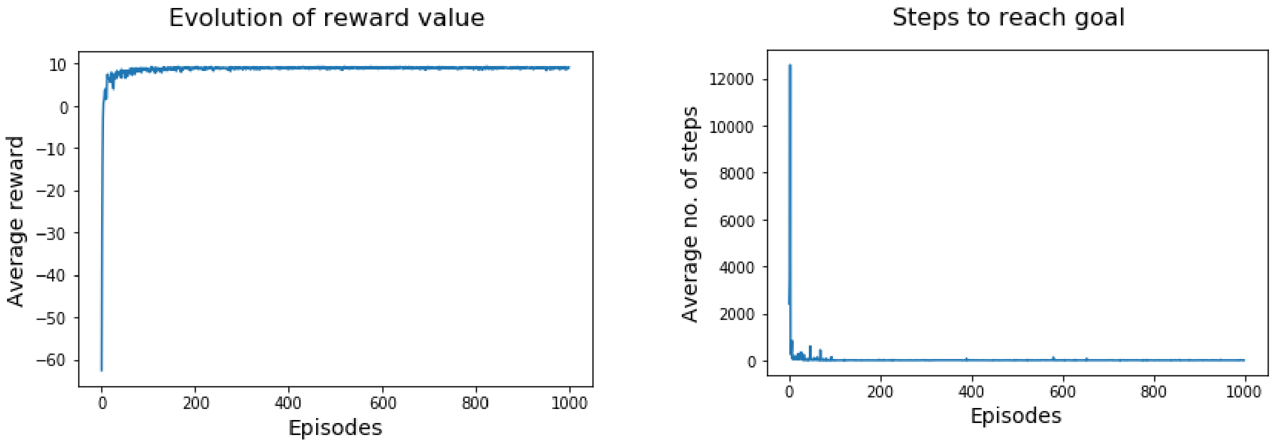


Fig 12: Performance of SARSA to achieve goal C

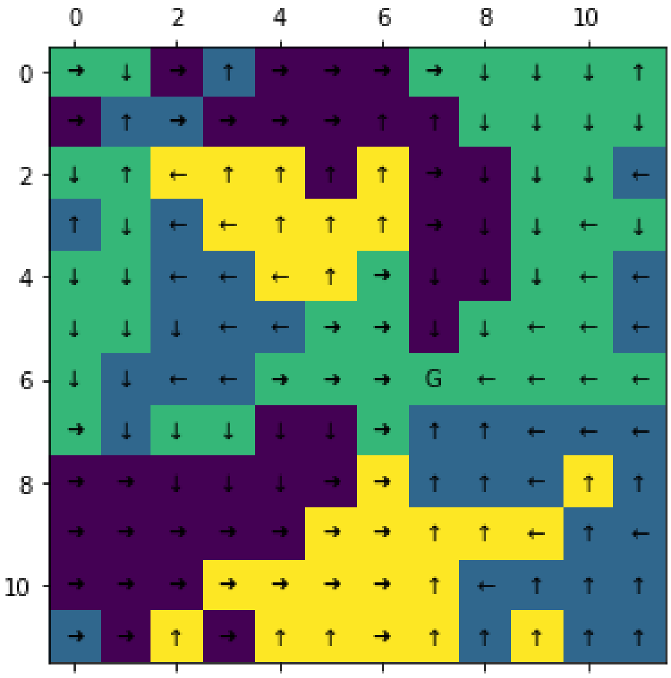


Fig 13: Optimal policy to arrive at C

Inferences

- The above policy figures indicate that, in this example, SARSA explores more than QLearning and the latter was observed to learn the optimal policy more quickly.

c) SARSA(λ)

Case A

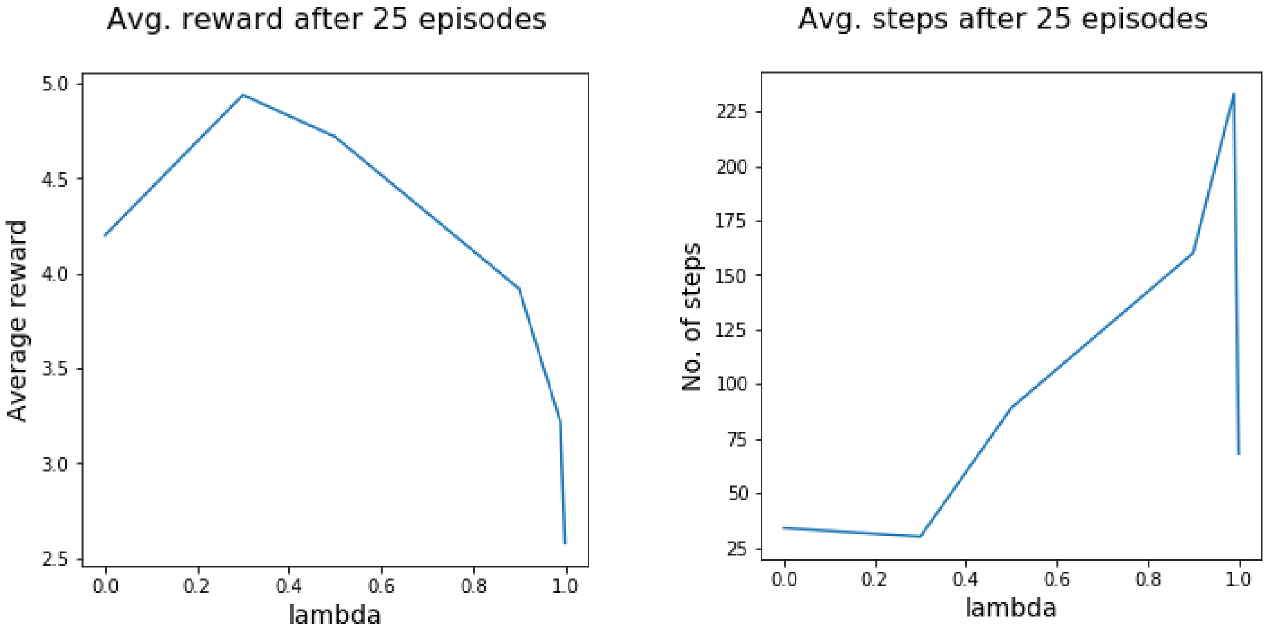


Fig 14: Performance of SARSA(λ) in arriving at A

Case B

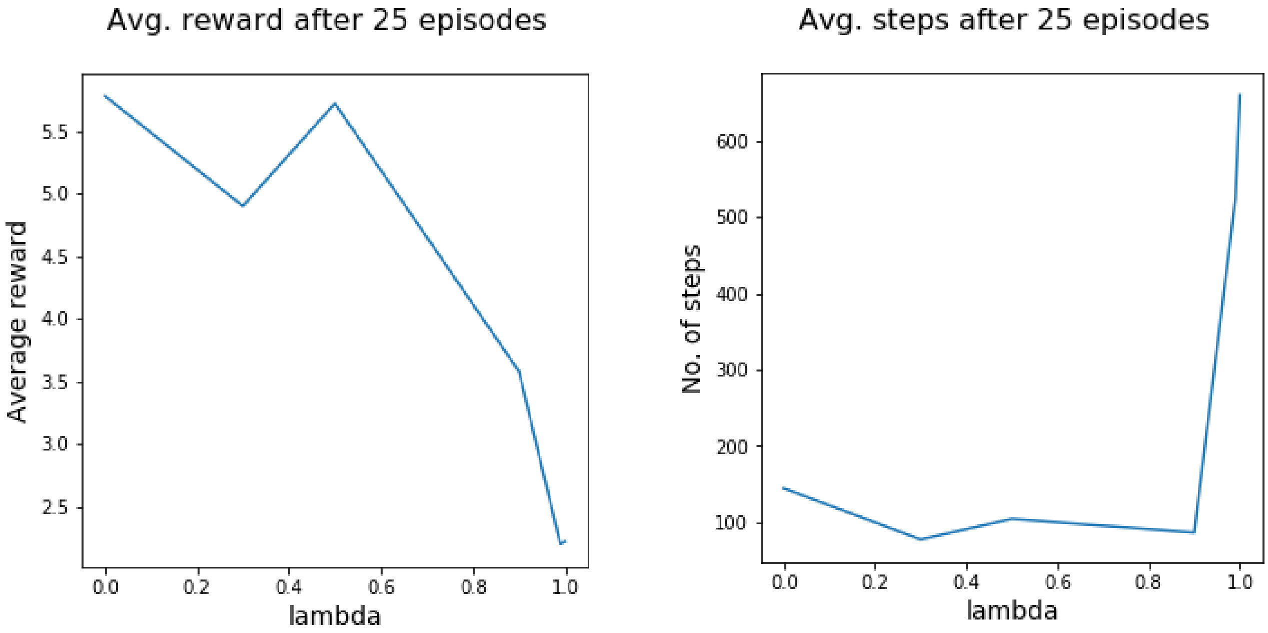


Fig 15: Performance of SARSA(λ) in arriving at B

Case C

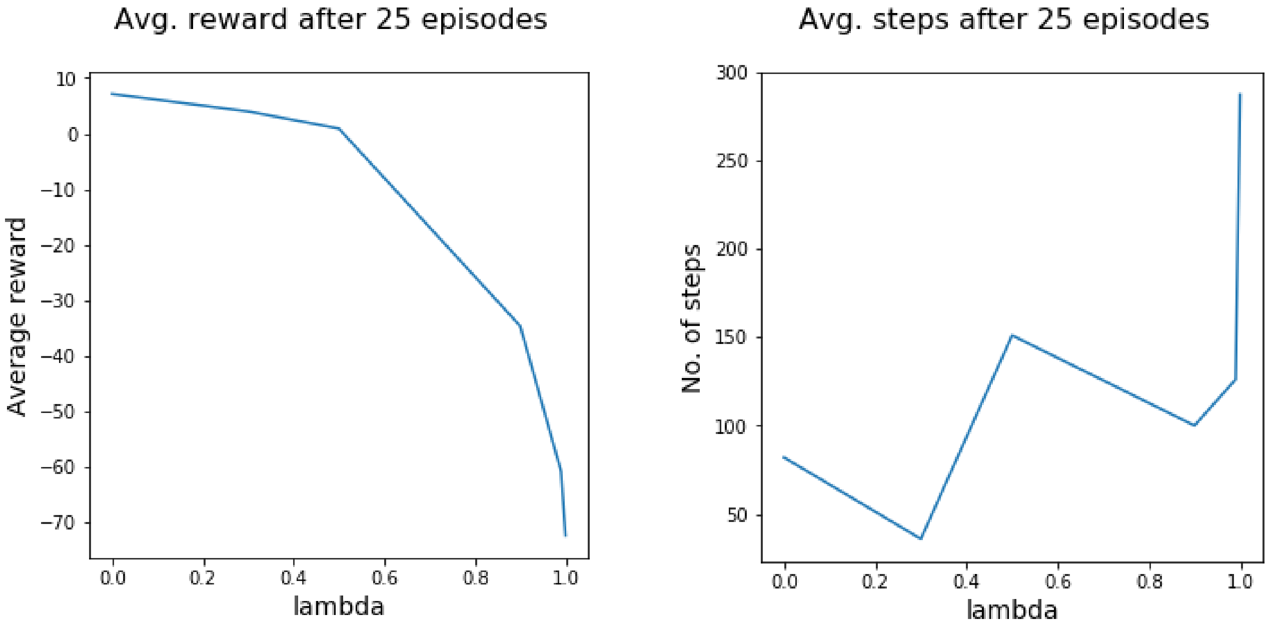


Fig 16: Performance of SARSA(λ) in arriving at C

Problem 2

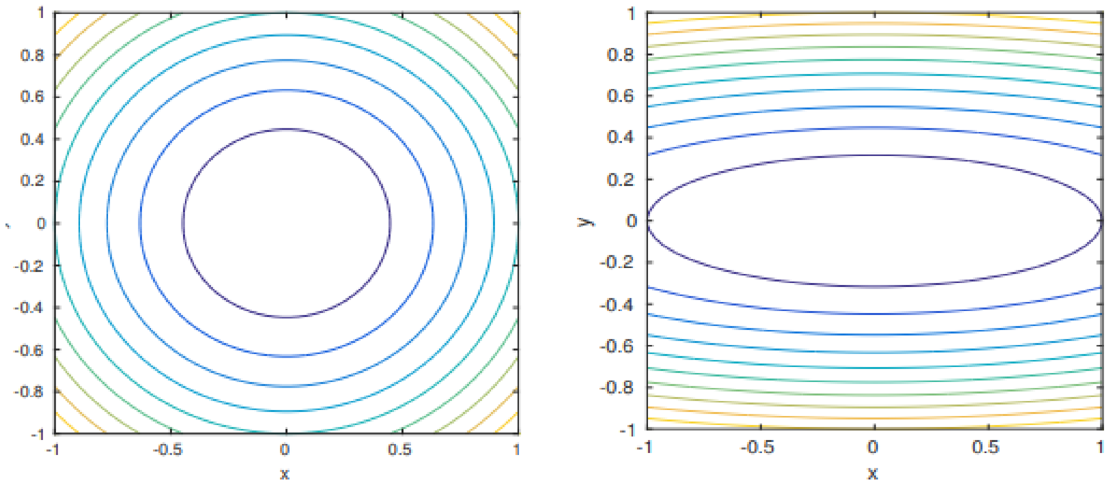


Fig 17: chakra and vishamC worlds

a) Hyperparameter tuning

The effect of varying the hyperparameters for both chakra and visham environments is as follows:

- Batch Size: If the size of a batch is small, learning of a policy requires more number of iterations although the number of episodes required for convergence more or less remained the same.
- Learning rate: A higher learning rate moves the solution faster towards the optimality but the downside is that it produces oscillations which affects convergence.
- Discount factor: When the discount factor γ was increased, more episodes were needed. But at the same time, lesser number of iterations were required for convergence.

b) Value function

$$f = \omega^T X, \text{ where } \quad \omega = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \end{bmatrix} \text{ and } \quad X = \begin{bmatrix} 1 \\ x^2 \\ y^2 \end{bmatrix}$$

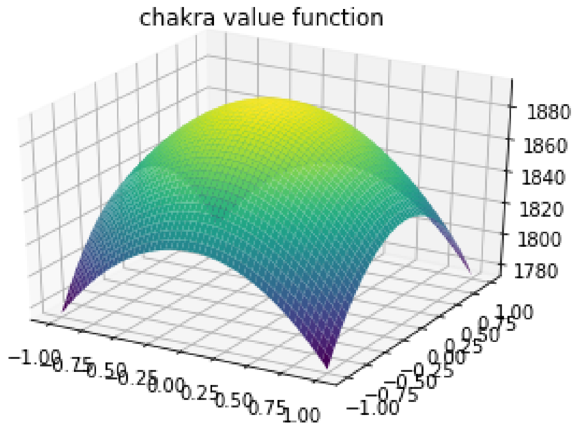


FIg 18: Value function visualized for the chakra world

c) Trajectories

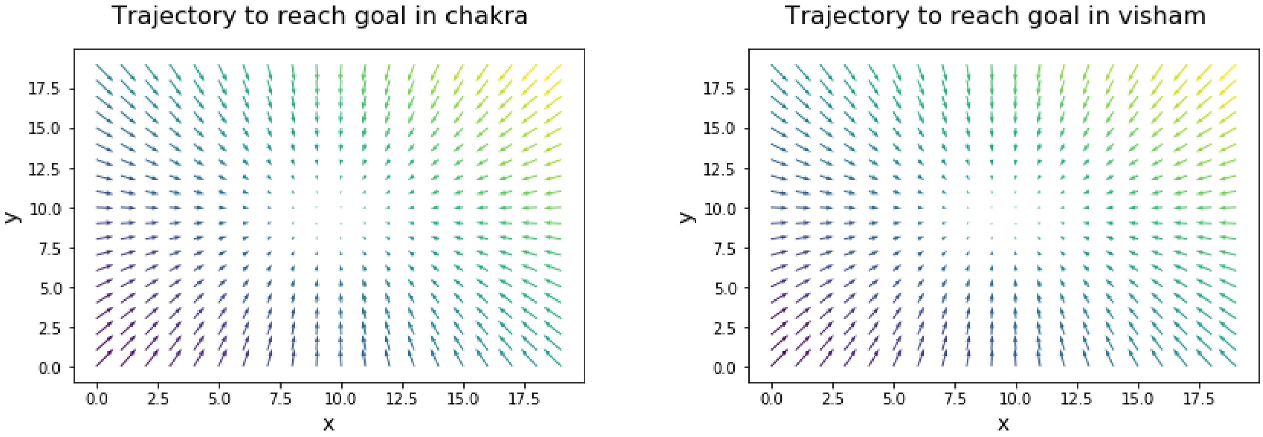


Fig 19: Policy Gradient trajectories for chakra and vishamC environments

Discussion

Mean value of distribution of actions was used to visualize the trajectory. It is evident from the above figures that the agent movies in a radial fashion towards the goal. Also, the agent tries to reach the goal as much possible along the x-direction in the vishamC world as the reward formulation is skewed.