Defining and Solving Reinforcement Learning Task

Hrushikesh Poola
hpoola@buffalo.edu

June 14, 2023

Abstract

This report contains three parts each part highlighting details about incremental development of training a Reinforcement Learning model using two algorithms - SARSA and Q-Learning. $Part-I$ of the report covers the implementation of a building an Environment. $Part-II$ and $Part-III$ is about using two techniques to learn about the environment built in $Part-I$. $Part-II$ uses SARSA algorithm to learn about the optimal policy of the environment. $Part-III$ uses Q-Learning algorithm to learn the optimal policy of the same environment used in $Part-II$

1 Part-I : Building Environment

1.1 About the Environment

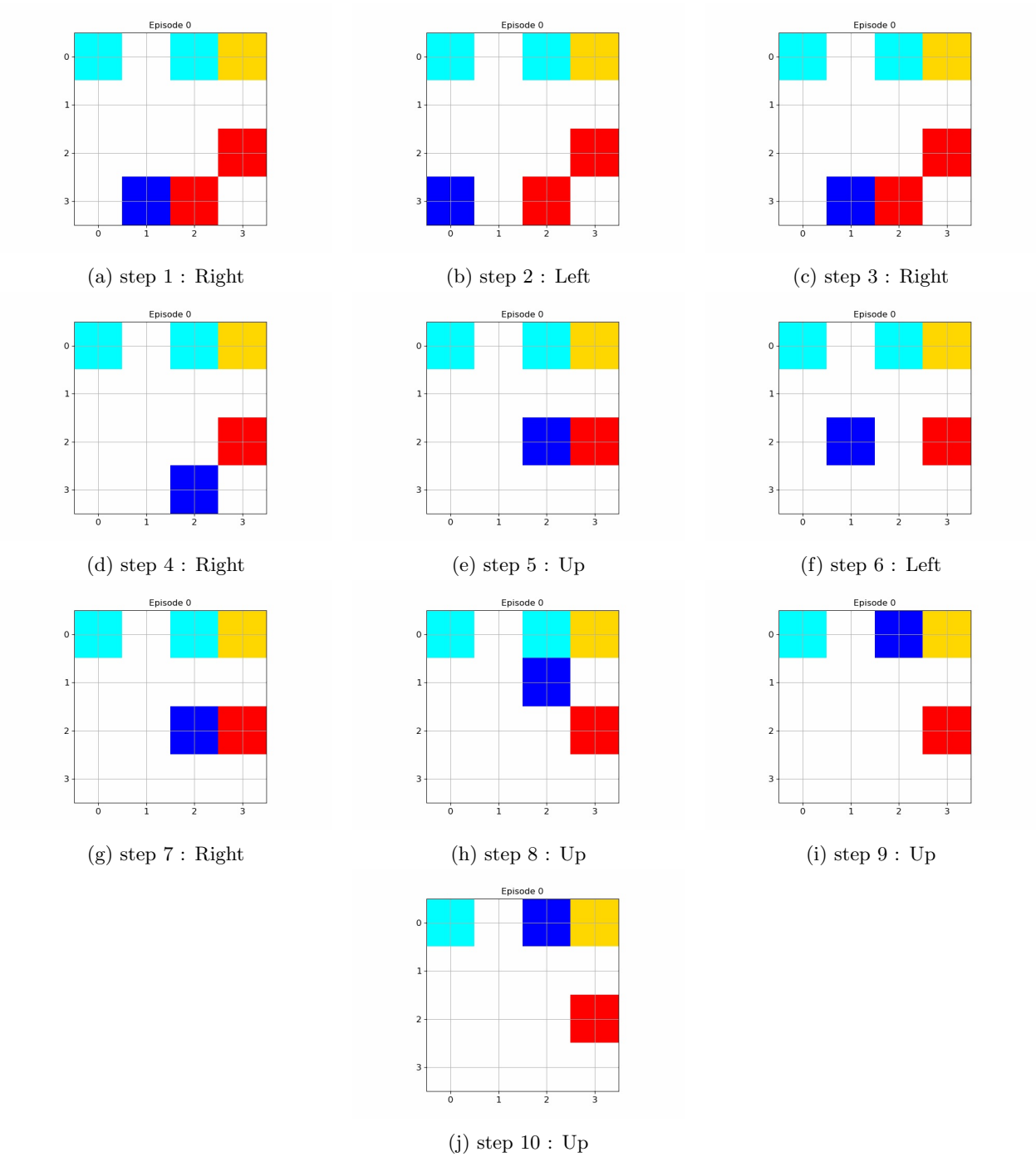
- In this problem, we choose to pick Grid based enviromnent with 16 states and 4 actions for each state.
- states of the environment is defined by reward at a particular cell of environment and position of the agent
- Main objective of the problem is that the agent should be able to explore and exploit and by maximizing the reward and reach the goal position of the environment defined.
- In the enviroment defined there are five rewards
  - goal position with +10 reward units
  - two positions with positive reward units (+5, +6). On optimal path the agent should be able to pick these rewards and maximize the overall reward of the episode.
  - two positions with negative reward units (-5 and -6). Agent should try to avoid these positions(as these positions might effect the overall reward for agent to reach goal position).

1.2 Adding Rewards to the Environment

As part of creating the enviromment added rewards in different positions of the environment for the agent to collect and learn about the optimal policy of the Environment.

1.3 Visualization of the Environment

Task is to randomly sample an action from all possible actions of agent and try to explore the environment. Visualization of the same is attached in gif format (under task1 folder in same folder as notebook file).



(a) step 1 : Right

(b) step 2 : Left

(c) step 3 : Right

(d) step 4 : Right

(e) step 5 : Up

(f) step 6 : Left

(g) step 7 : Right

(h) step 8 : Up

(i) step 9 : Up

(j) step 10 : Up

Figure 1: Episode with 10 Timesteps with different Actions at each timestep(Agent : Blue Color)

1.4 Safety in AI

- Environment is defined with a limited number of positions for agent to access. This will ensure that the agent will know only about the defined positions of the environment (restricting the scope)
- By avoiding the actions that leads agent to go out of the environment.This is achieved by omitting the action that leads agent to move out of environment.
- Agent action is picked only from set of allowed actions defined.This will ensure that agent will not pick any arbitrary action.
- By both exploring and exploiting, we can ensure that agent is not biased to certain paths in the environment during initial episodes, once the agent explored enough it will try to choose the optimal path(in our case maximum overall reward path) to reach goal state.

---

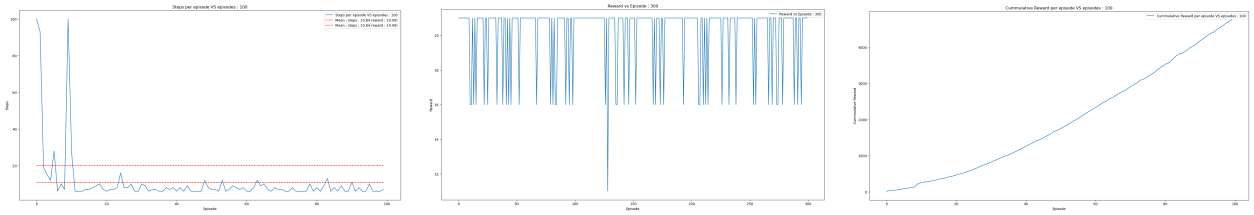
2 Part-II : SARSA

2.1 About Algorithm

In the algorithm we use state action table and epsilon greedy policy to learn about the environment by both exploring and exploiting.After each action the state action table will be updated based on the current position, action, next position, next action and reward.

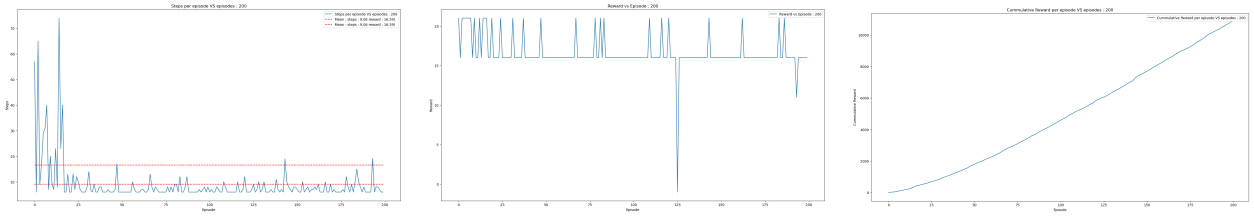
2.2 Experiments based on Episodes

By changing number of episodes with epsilon decay factor as 1e-6. plotted steps vs episodes, reward vs epidsoes graphs.



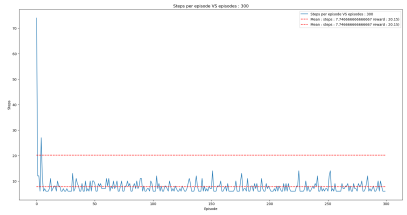
(a) steps graph (b) reward graph (c) Cumulative Reward graph

Figure 2: Graphs with 100 episodes

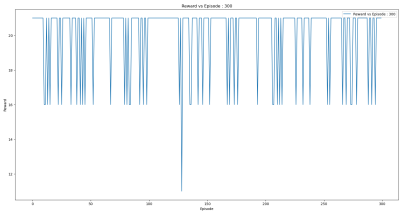


(a) steps graph (b) reward graph (c) Cumulative Reward graph

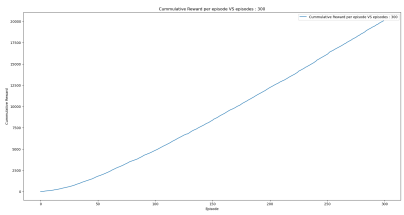
Figure 3: Graphs with 200 episodes



(a) steps graph

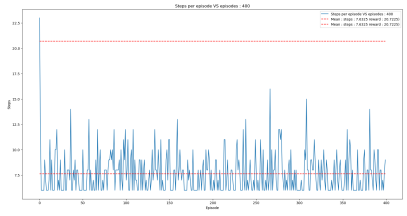


(b) reward graph

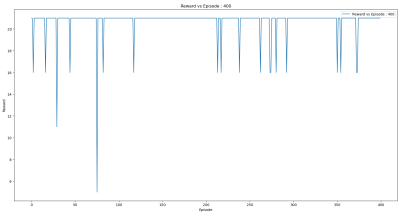


(c) Cumulative Reward graph

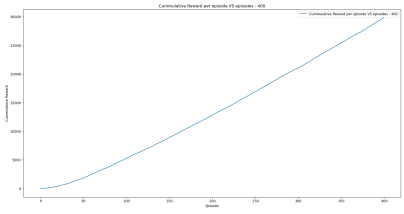
Figure 4: Graphs with 300 episodes



(a) steps graph

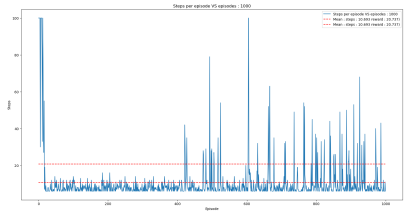


(b) reward graph

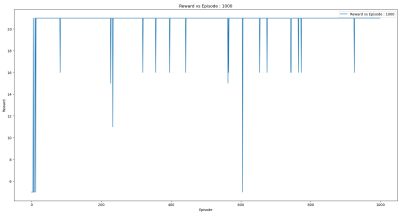


(c) Cumulative Reward graph

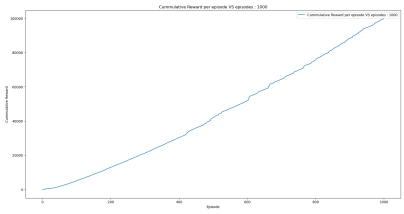
Figure 5: Graphs with 400 episodes



(a) steps graph

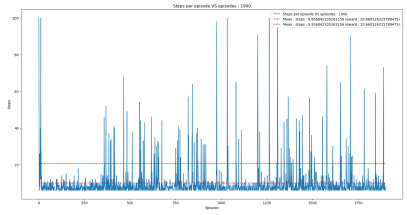


(b) reward graph

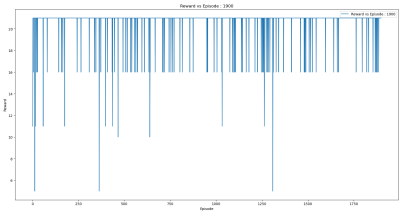


(c) Cumulative Reward graph

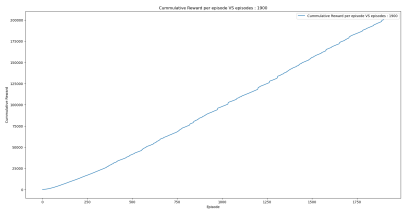
Figure 6: Graphs with 1000 episodes



(a) steps graph



(b) reward graph



(c) Cumulative Reward graph

Figure 7: Graphs with 1900 episodes

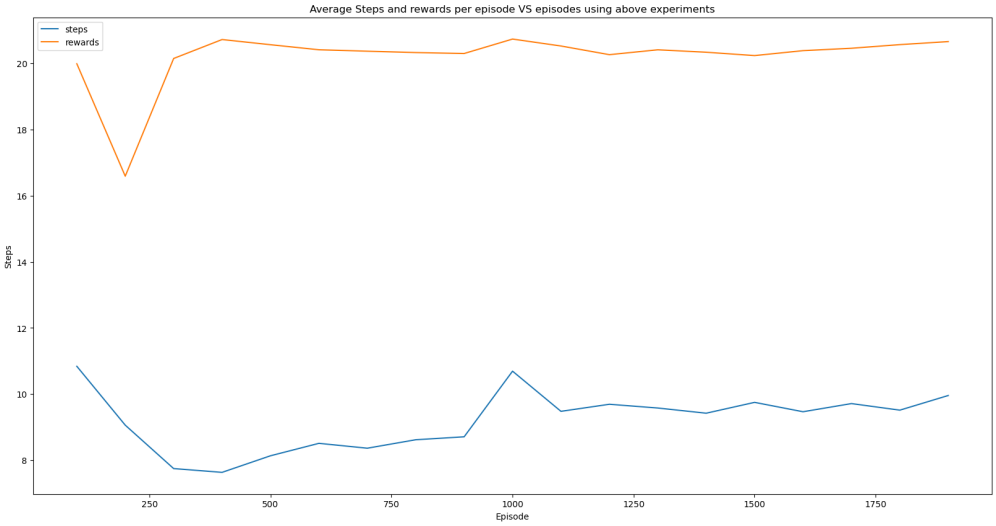


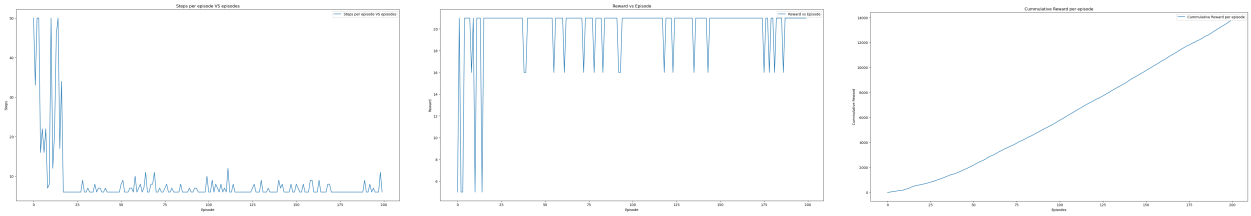
Figure 8: Graph with mean values of above experiments

2.3 Stats from optuna

2.3.1 Hyperparameters with maximum expected reward

- alpha = 0.11713750290022812
- episodes = 200
- epsilon = 0.12253351733401509
- gamma = 0.9289409987872245
- steps = 50

2.3.2 Graphs based on above parameters



(a) Steps graph (b) Reward graph (c) Cumulative Reward graph

Figure 9: Graph with hyperparameters given by optuna with max reward for 200 trails

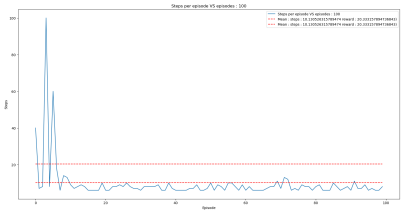
3 Part-III : Q-Learning

3.1 About Algorithm

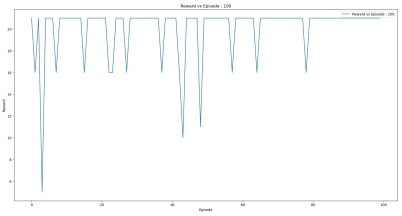
In the algorithm we use state action table and epsilon greedy policy to learn about the environment by both exploring and exploiting.After each action the state action table will be updated based on the current position, action, next position and reward.

3.2 Experiments based on Episodes

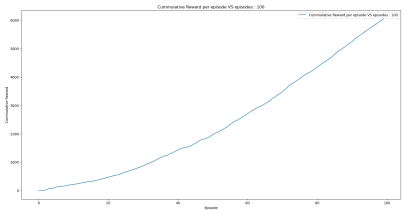
By changing number of episodes with epsilon decay factor as 1e-6. plotted steps vs episodes, reward vs epidsoes graphs.



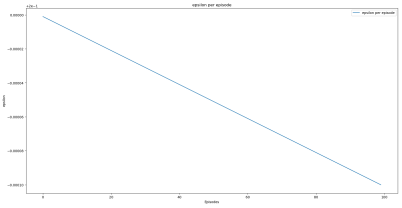
(a) steps graph



(b) reward graph

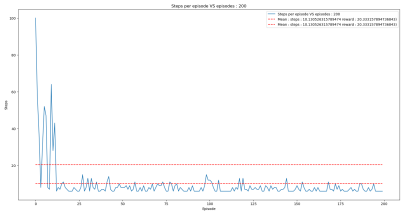


(c) Cumulative Reward graph

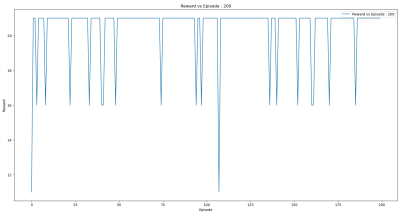


(d) Epsilon decay

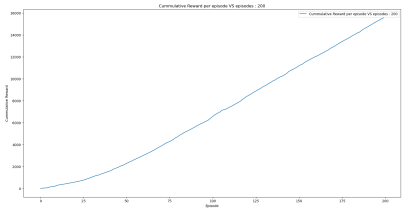
Figure 10: Graphs with 100 episodes



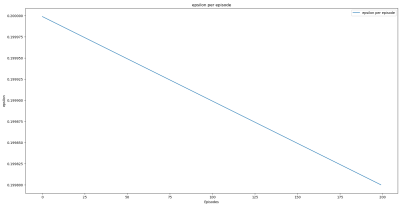
(a) steps graph



(b) reward graph

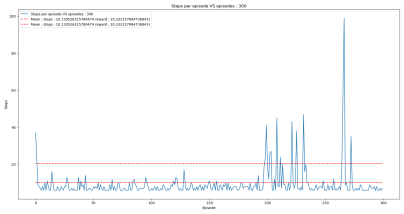


(c) Cumulative Reward graph

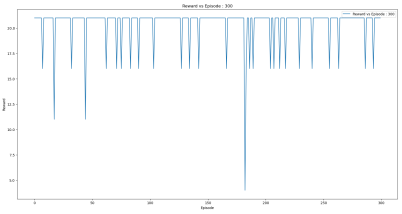


(d) Epsilon decay

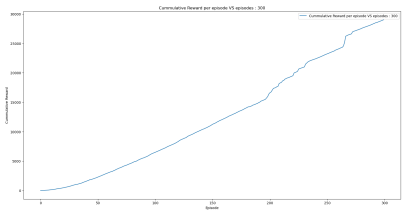
Figure 11: Graphs with 200 episodes



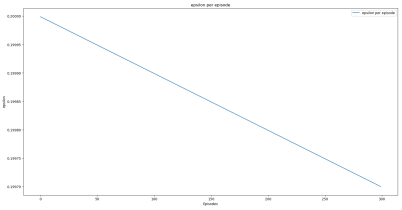
(a) steps graph



(b) reward graph

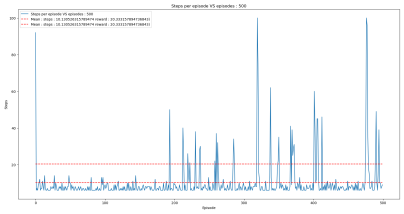


(c) Cumulative Reward graph

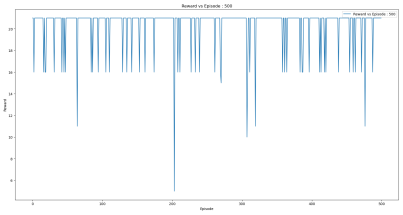


(d) Epsilon decay

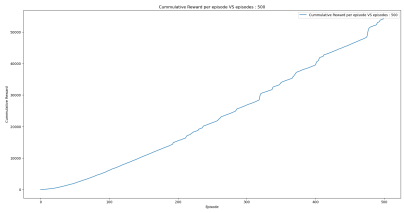
Figure 12: Graphs with 300 episodes



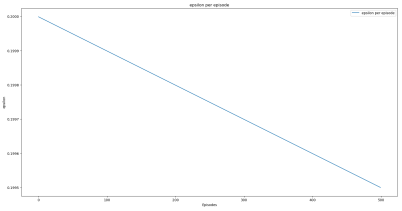
(a) steps graph



(b) reward graph

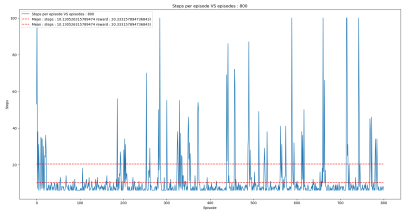


(c) Cumulative Reward graph

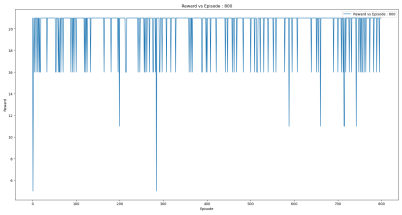


(d) Epsilon decay

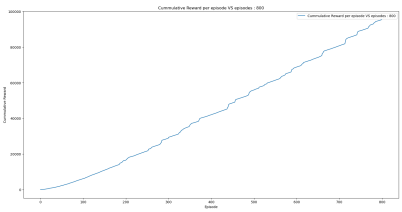
Figure 13: Graphs with 500 episodes



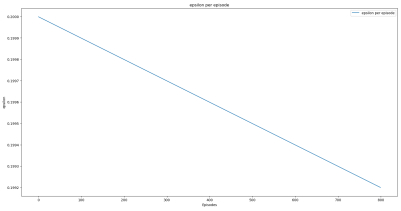
(a) steps graph



(b) reward graph

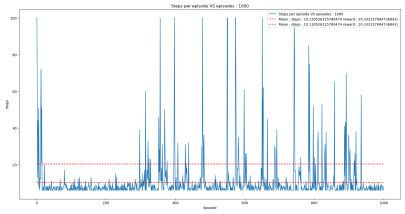


(c) Cumulative Reward graph

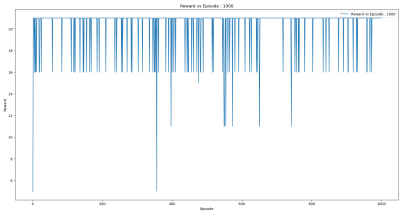


(d) Epsilon decay

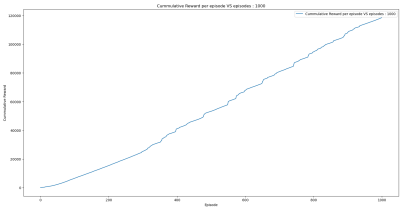
Figure 14: Graphs with 800 episodes



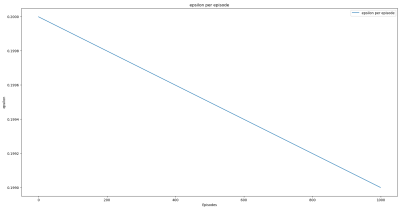
(a) steps graph



(b) reward graph

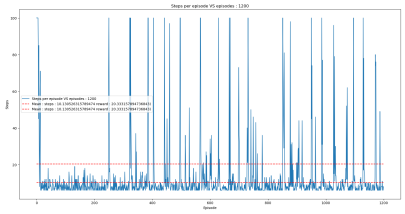


(c) Cumulative Reward graph

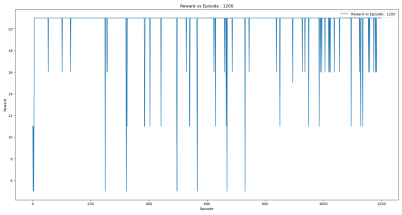


(d) Epsilon decay

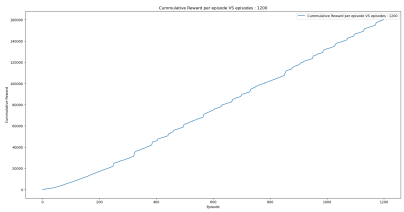
Figure 15: Graphs with 1000 episodes



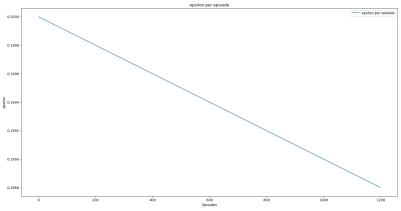
(a) steps graph



(b) reward graph

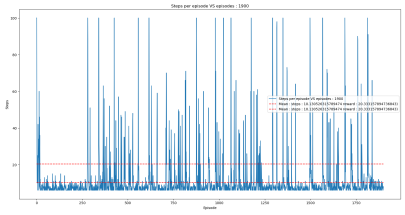


(c) Cumulative Reward graph

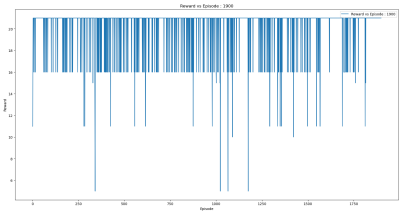


(d) Epsilon decay

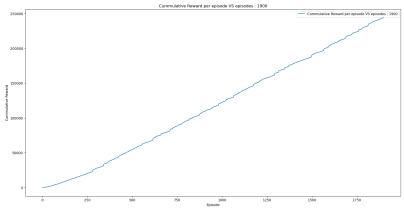
Figure 16: Graphs with 1200 episodes



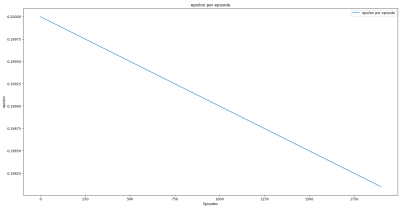
(a) steps graph



(b) reward graph



(c) Cumulative Reward graph



(d) Epsilon decay

Figure 17: Graphs with 1900 episodes

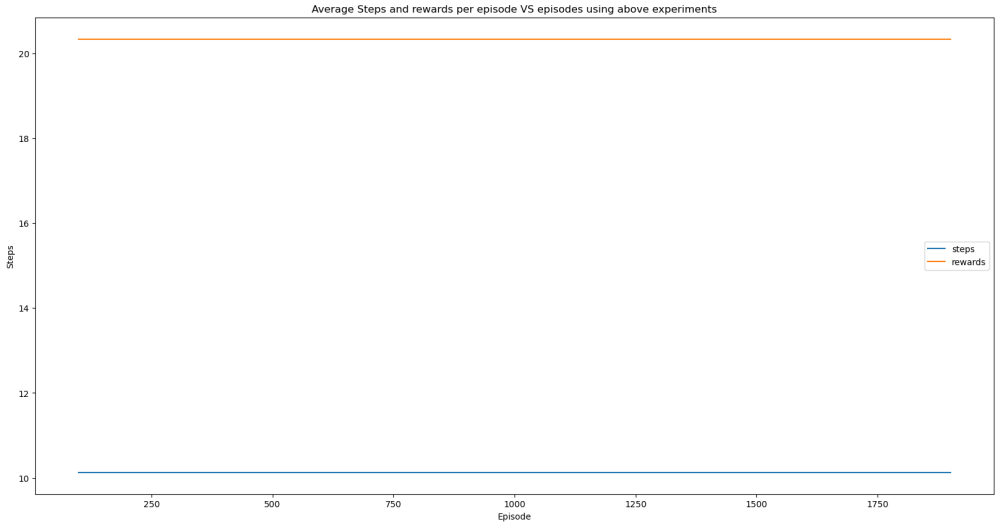


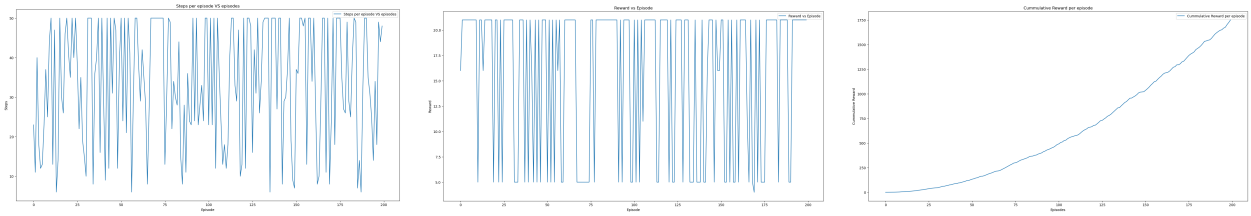
Figure 18: Graph with mean values of above experiments

3.3 Stats from optuna

3.3.1 Hyperparameters with maximum expected reward

- alpha = 0.0015656820648213832
- episodes = 200
- epsilon = 0.19661118556986412
- gamma = 0.8486582251694117
- steps = 50

3.3.2 Graphs based on above parameters



(a) Steps graph (b) Reward graph (c) Cumulative Reward graph

Figure 19: Graph with hyperparameters given by optuna with max reward for 200 trails

4 Comparision

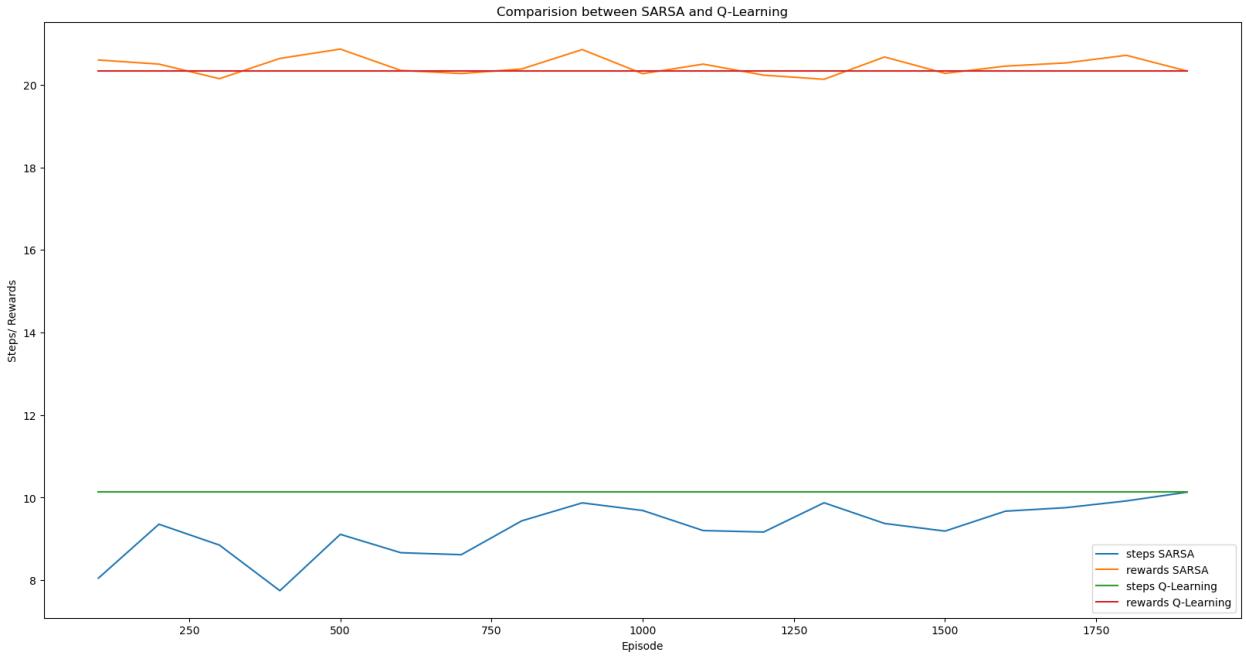


Figure 20: Comparision between SARSA and Q-Learning

- In SARSA, state action table is updated based on current state, action, reward, next state and next action, whereas in Q-Learning instead of next action we pick the maximum value of all possible actions in next state to update the state action value of current state.
- Both SARSA and Q-Learning chooses next action based on epsilon greedy policy, but only difference is SARSA picks the value of next state action pair whereas Q-Learning picks the maximum of all possible actions from next state values from state action value table.
- The same can be seen from the graph above the average steps and rewards in case on Q-Learning is almost constant as the state action table is updated using the optimal value at each teimestep, whereas SARSA is depending on the action it is taking to update the state action table.
- Observation from the above graph is that SARSA converges to optimal policy based on the action it is taking at each timestep whereas Q-Learning is moving towards optimal path independent of the action taken. But both SARSA and Q-Learning uses state action pair table to move towards optimal policy.
- Difference in picking the optimal policy can be clealy seen from the timesteps of different episodes using SARSA and Q-Learning. These can be visualized from the gif files from folder task2 and task3. In **task2**(folder images) inital episodes the agent is trying to explore and exploit but the weightage of exploiting is dependant on the action taken and agent eventually learns the optimal policy, whereas **task3**(folder images) even in inital episodes agent is trying to pick the optimal policy as the policy is derived from state action table and the state action table is updated using the max value of next state in Q-Learning.

5 References

- https://pypi.org/project/gymnasium/
- https://github.com/optuna/optuna
- https://matplotlib.org/stable/api/animation$_{a}pi.html$
- https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html
- https://docs.python.org/3/library/glob.html
- https://docs.python.org/3/library/os.html
- https://gymnasium.farama.org/api/spaces/