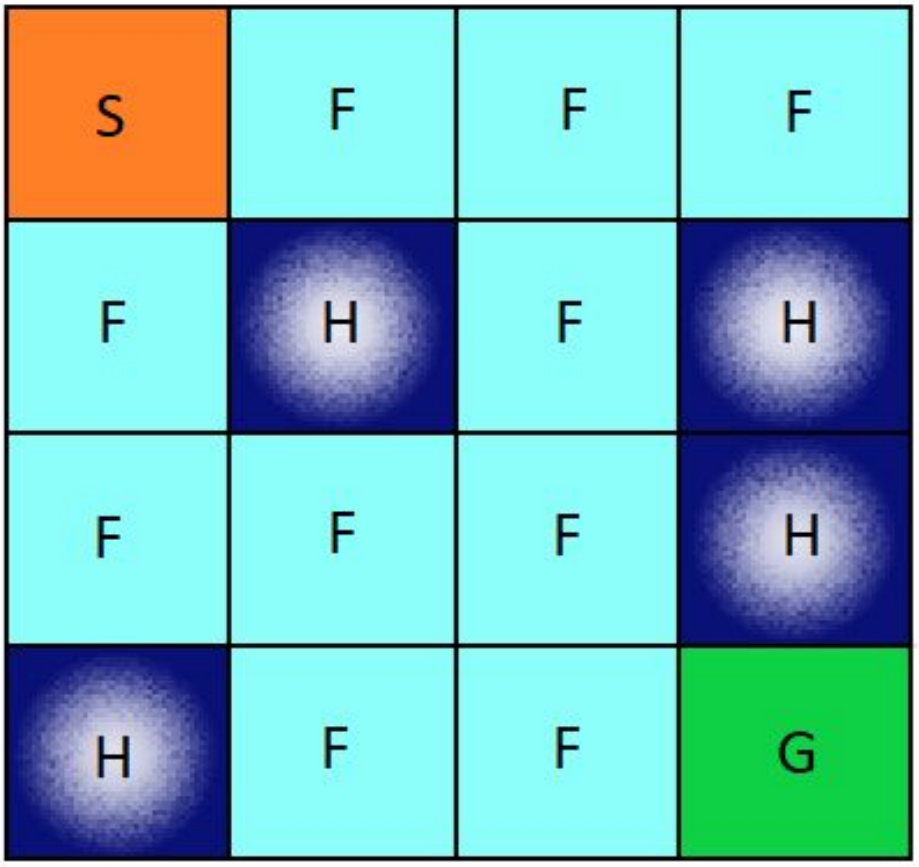
MACHINE LEARNING

COMP3611

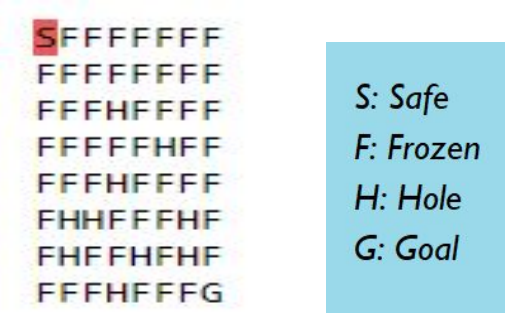


Sudhanshu Kumar Singh

10.12.2018

INTRODUCTION

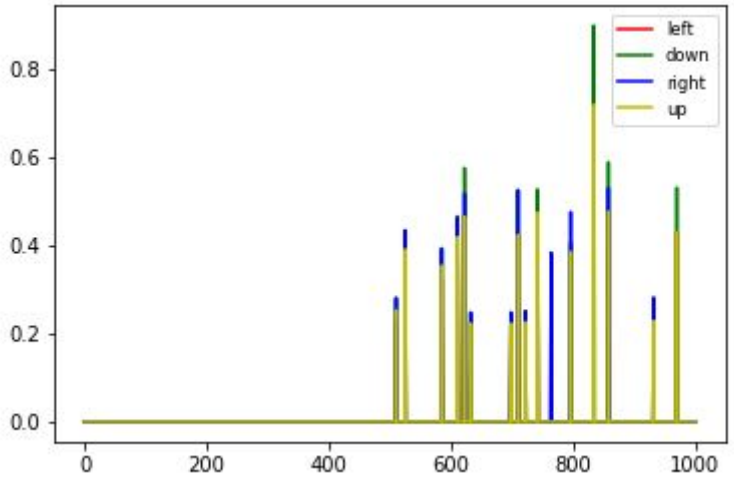
The Frozen Lake environment is a 8×8 grid which contain four possible areas — Safe (S), Frozen (F), Hole (H) and Goal (G). The agent moves around the grid until it reaches the goal or the hole. If it falls into the hole, it has to start from the beginning and is rewarded the value 0. The process continues until it learns from every mistake and reaches the goal eventually.

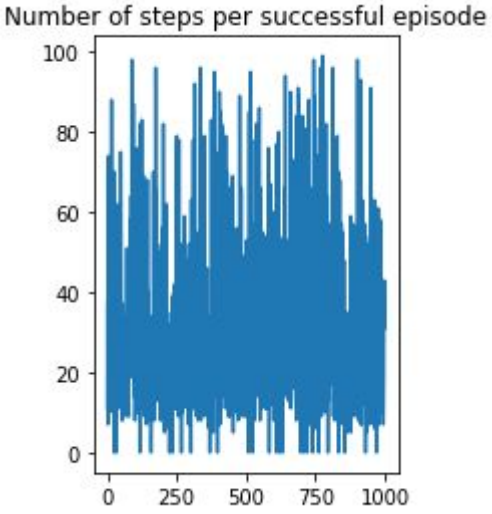


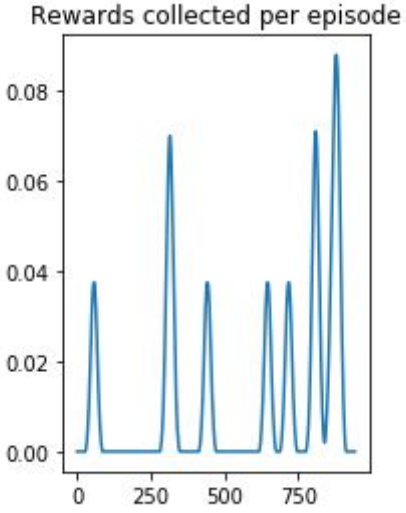
You are given a template of the code structure which you will use for implementing this work. The “main” function initializes the World and the Agent. The main function includes the main loop for running training episodes. Also, there are plotting functions provided to help you visualize the results. You will see that there are three plots that we expect you to show. The first plot contains the q-value of the available actions in the initial state over training episodes as recorded by “world.q_Sinit_progress”. The second plot contains the evolution of the number of steps per successful episode over training episodes. A successful episode is an episode in which the agent reaches the goal (i.e., not just any terminal state) as recorded by “episodeStepsCnt_progress”. The third plot contains the evolution of the return per episode as recorded by “r_total_progress”. The last plot is made smoother using the running_mean function. It is for you to implement the code that updates these values

On- Vs Off-Policy:

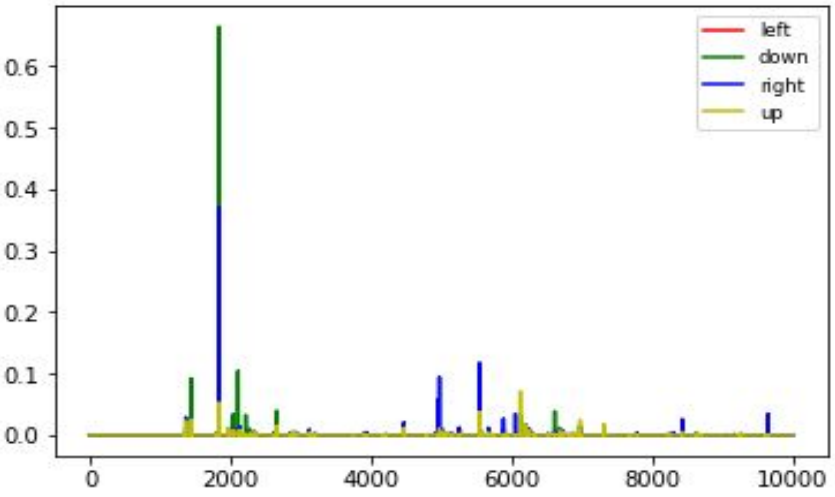
RL Algorithm: Q- learning, epsilon= 0.8

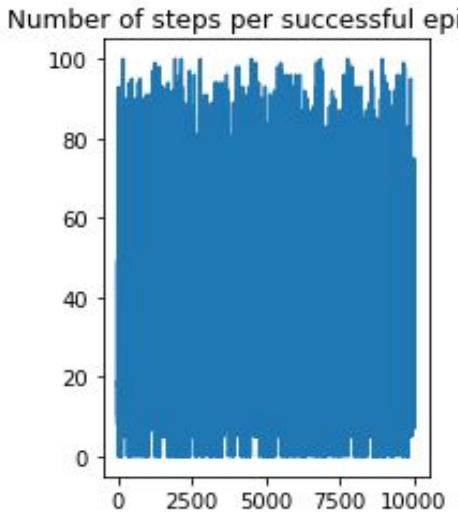


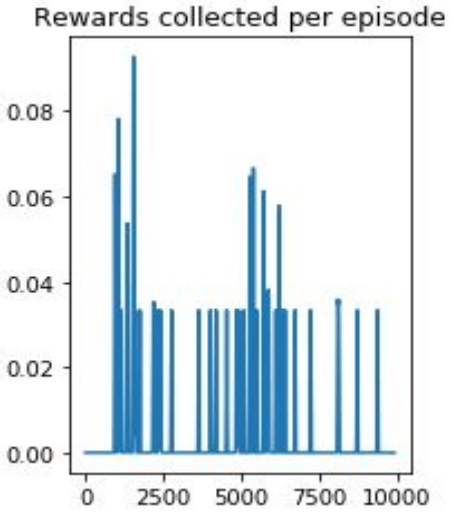




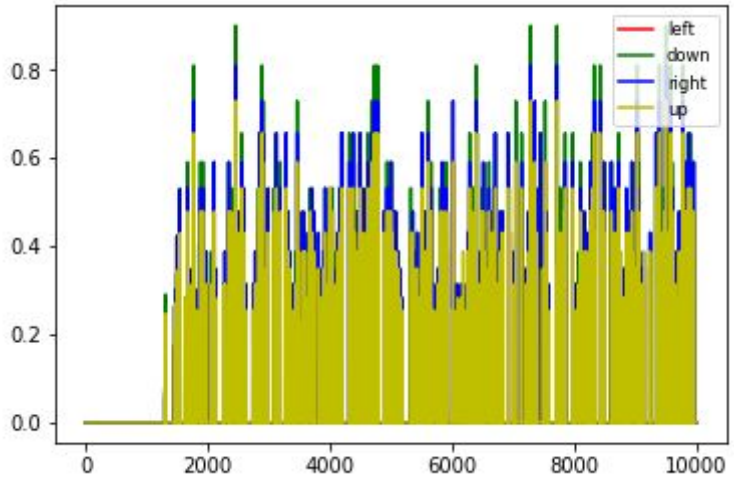
RL Algorithm: SARSA- learning, epsilon= 0.8

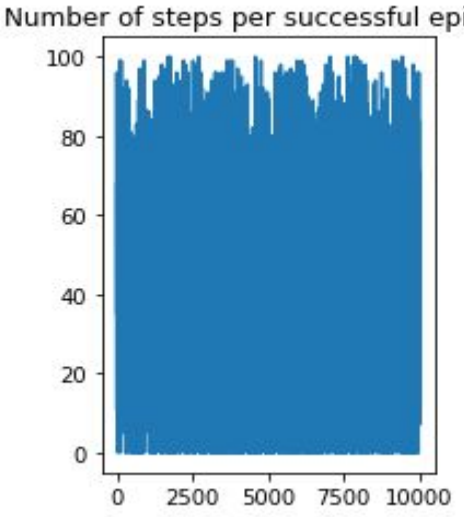


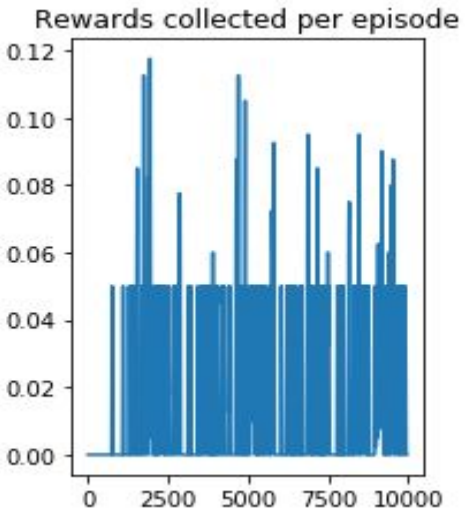




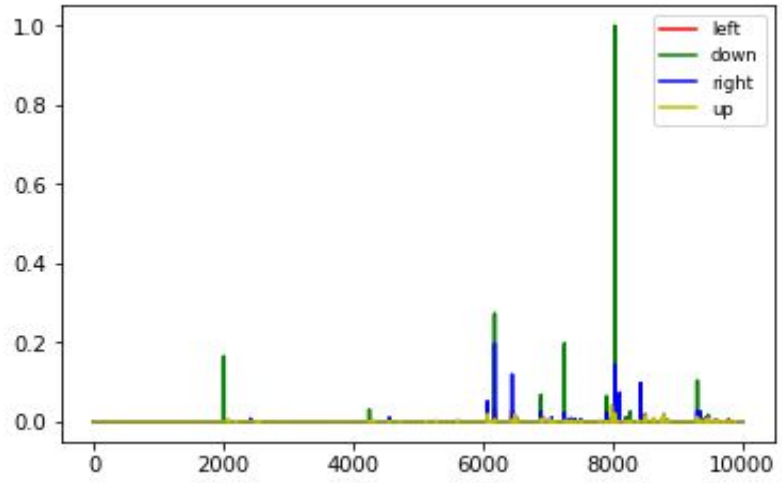
RL Algorithm: Q- learning, epsilon= 0.1

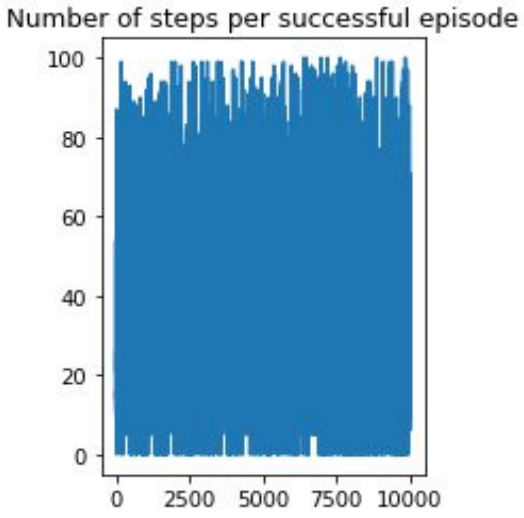


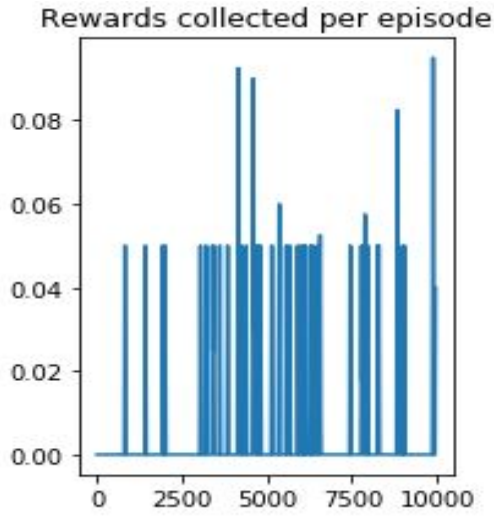




RL Algorithm: SARSA- learning, epsilon= 0.1







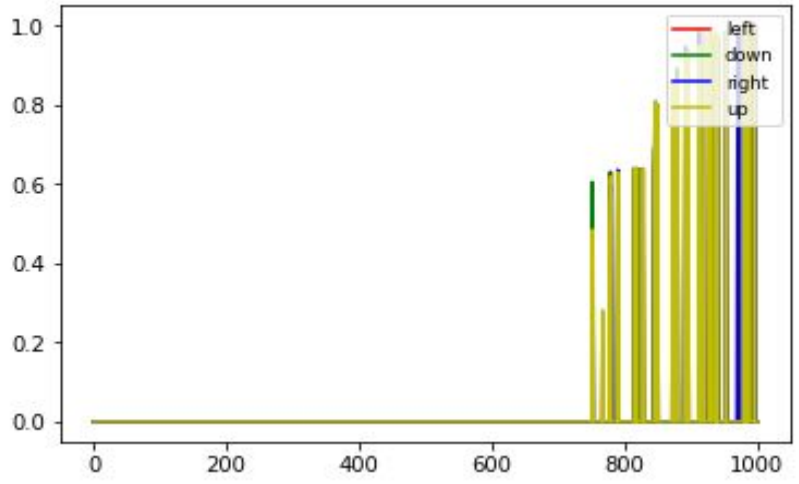
So, the Q- learning RL algorithm is off- policy learning algorithm and SARSA-learning is on- policy learning algorithm. Q- learning is off- policy because the next action *a'* is chosen to maximize the next state's Q-value instead of following the current policy. As a result, Q-learning belongs to the off-policy category. SARSA is on-policy agent because it learns the Q-value based on the action performed by the current policy instead of the changing the policy by greedy approach.

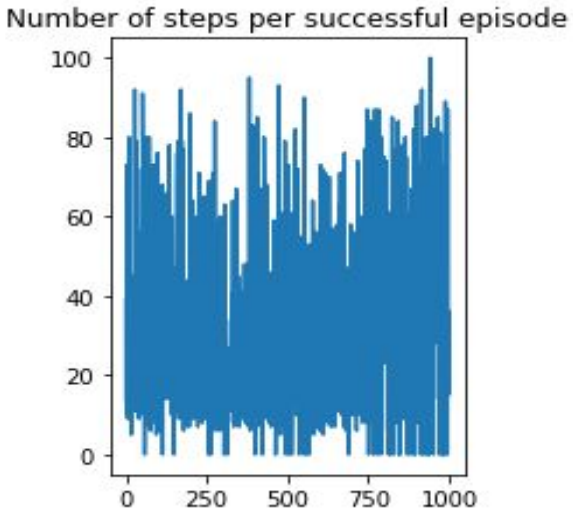
As epsilon (exploration constant) values becomes smaller, then Q- learning will approach to on-policy behaviour (same as SARSA) and may converge to local maxima. But when epsilon increases, agent tries to find other possible path for achieving the goal.

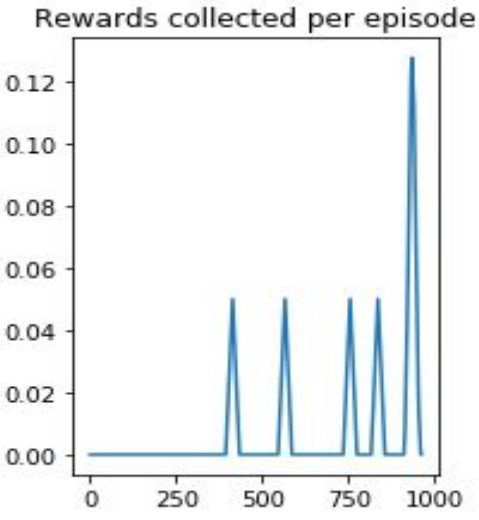
Yes, it can affect the convergence to optimal policy by not at all converging due to very less exploration constant or by getting stuck to local maxima.

The Discount Factor :

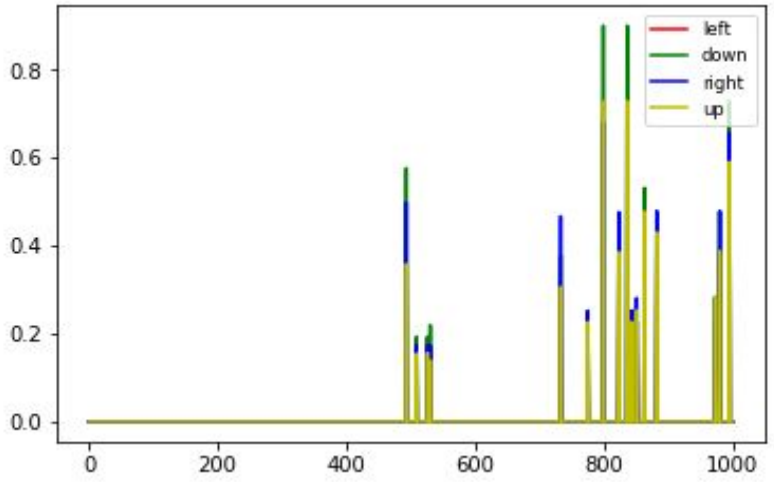
RL Algorithm: Q- learning, gamma = 1

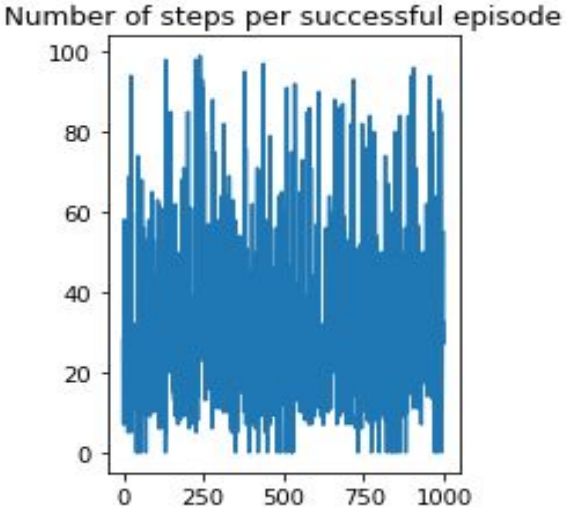


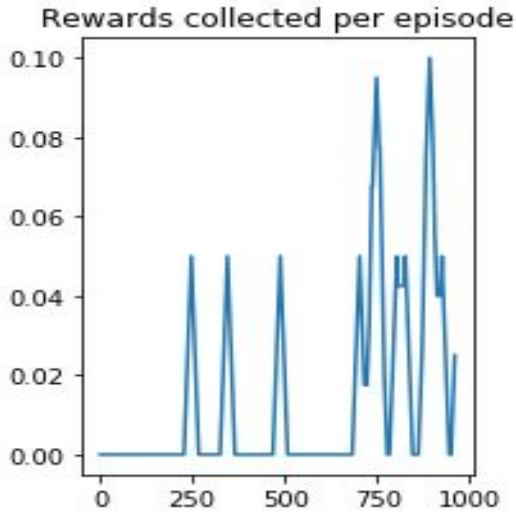




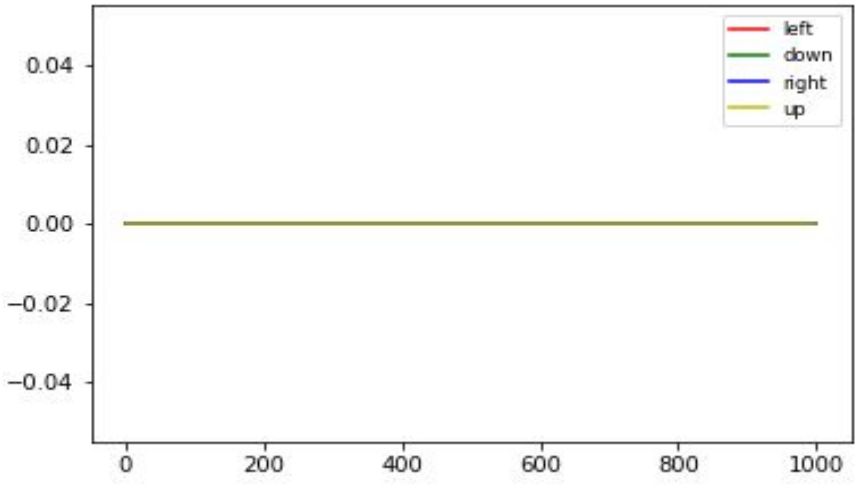
RL Algorithm: Q- learning, gamma = 0.9

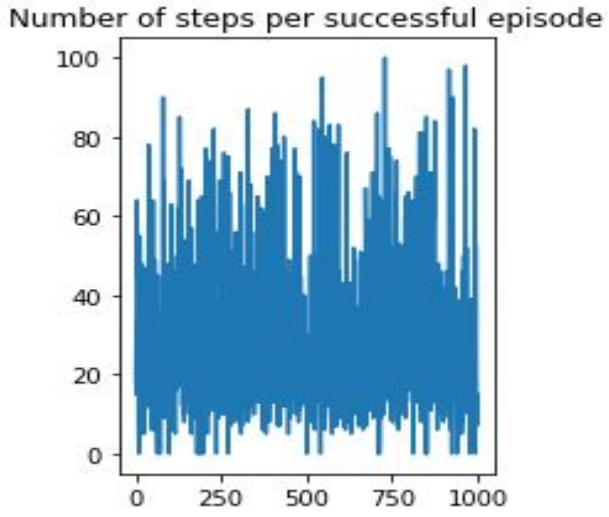


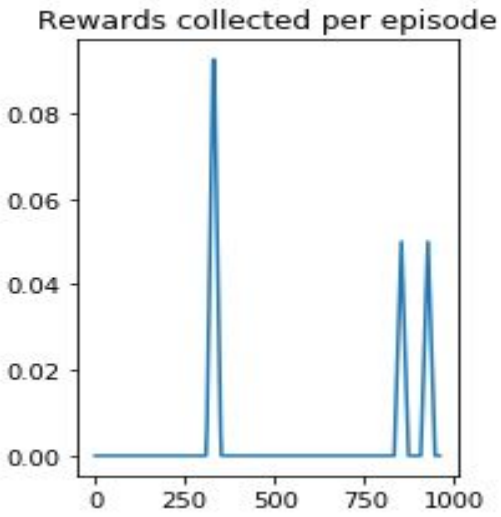




RL Algorithm: Q- learning, gamma = 0





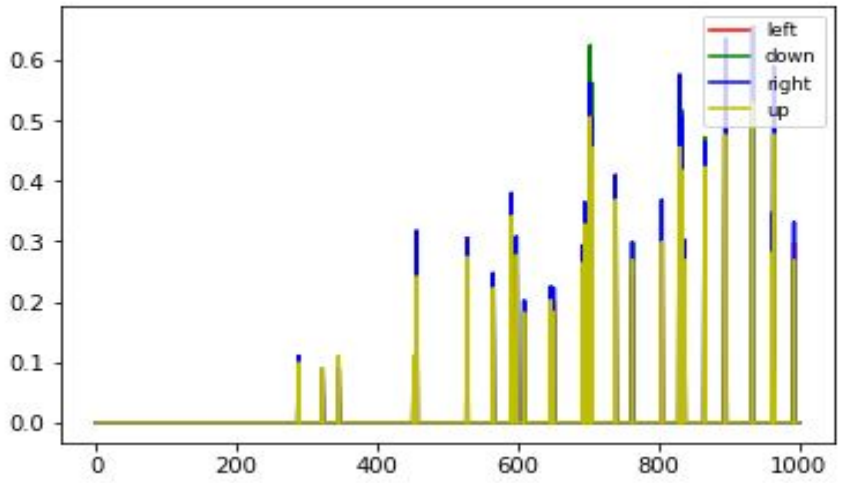


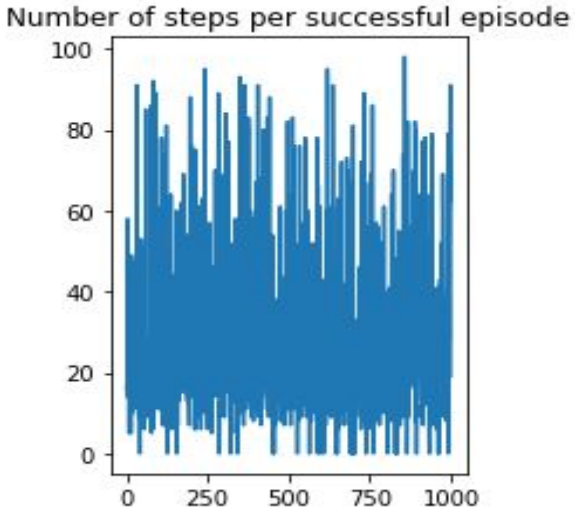
The discount factor is a measure of how far ahead in time the algorithm looks. To prioritise rewards in the distant future, keep the value closer to one. A discount factor closer to zero on the other hand indicates that only rewards in the immediate future are being considered, implying a shallow lookahead. It is nearly always arbitrarily chosen to be near the 0.9 point. So, a discount factor of 1 may never converge (but it converging here in 10000 episodes), while for discount factor of 0 it gets trapped into nearby states. So as we increase gamma, the learned policy will increase and will take future actions into account more effectively. The Value function also increases as gamma increases but then convergence of the agent may fail.

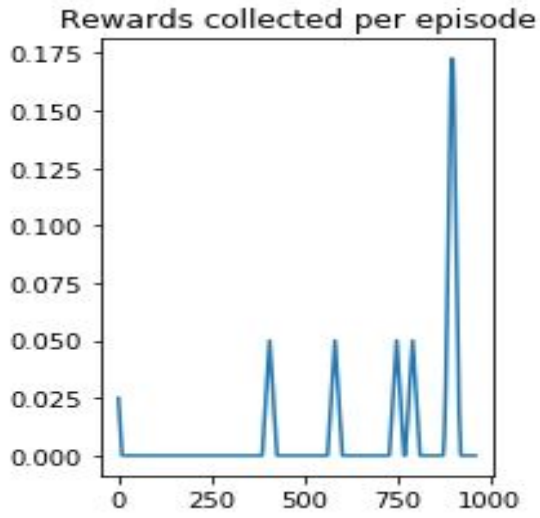
The Environment Stochasticity :

In a Stochastic environment ,when the agent repeats doing the same action in a given state, the new state and received reward may not be the same each time.

While in Deterministic environment ,if the agent while in a given state repeats a given action, it will always go the same next state and receive the same reward value.







So, from the Reward collected per episode we can see that the agent learned its optimal policy very late (around 900 episodes later) and the reasons for this is the stochastic environment. Due to ever changing state-action pair, the stability of model was difficult, which the agent learns very late. The highest learning rate is 0.9, around which it learns very fast. For lower values of learning rate the agent learns at very end or sometimes don't learn at all. For higher than 0.95 values of learning rate, the model diverges and the agent do not learns anything.