CS6700 Reinforcement Learning
Report for PA #1

Varun Gumma CS21M070
Harsha Vardhan Gudivada CS21M021

1 Introduction

The assignment's solution can be divided into 1. Environment 2. Learner 3. Reinforcement Learning Algorithms 4. Hyper parameters 5. Hyper parameter search. 6. Plots

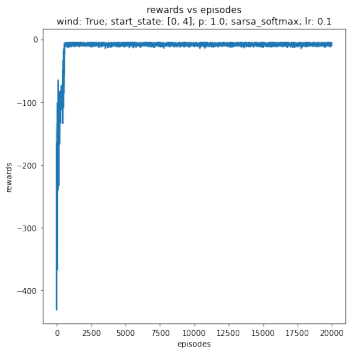
1. Environment: Code for the environment is already provided in the assignment as the `Grid World` class. For each training algorithm, the environment is instantiated 16 times, once for each configuration.
2. Learner: A Reinforcement Learner class, `RLearner` was created to use the learning algorithms, exploration strategies in learning the optimal policy for each of the 16 configurations of the environment.
3. Reinforcement Learning algorithms: SARSA and Q-Learning were implemented in the code and used during the policy learning phase.
4. Hyper parameters: Hyper parameters are like the ϵ in the $\epsilon-$greedy exploration, temperature β in the softmax exploration, learning rate α, discount factor γ. They influence the effectiveness and efficiency of policy learning. In code, they are maintained in a list of lists.
5. Hyper parameter search: A single game play is a set of 3 independent runs. Each run consists of 20000 episodes and each episode has a maximum of 100 steps. The Q-values, steps-to-completion, rewards, state-visit-counts are averages across these 3 independent runs and those were taken as unbiased estimates for the aforementioned quantities. For each configuration, we experiment with 4 different α values. For each of the 8 configurations of the environment, we use the α value which gives the highest average reward (after averaging over the 3 runs).
6. Plots: Once we have the best hyper parameters for each environment configuration, plots for rewards, steps-to-completion, state-visit-counts and Q-values & optimal actions are made for the best hyper parameter set.

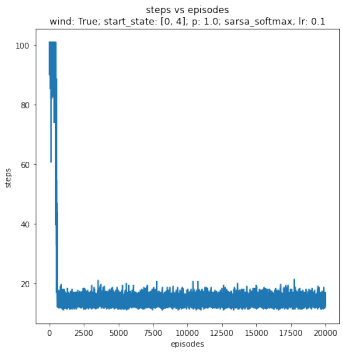
Some thoughts behind the hyper parameter search: It was observed that the rewards obtained were proportional to γ, and decreasing its value (say to 0.1 or 0.4) gave lower values. To get best results, the agent always preferred the largest discount value, and it was meaningless to change it much. Hence, we fixed it to the standard value of 0.9. Also, we decayed the value of β (for SARSA) and ϵ (for epsilon-greedy) to aid convergence. We decayed the values after every episode and set a floor at 0.001. Therefore, the decay equations are as follows:

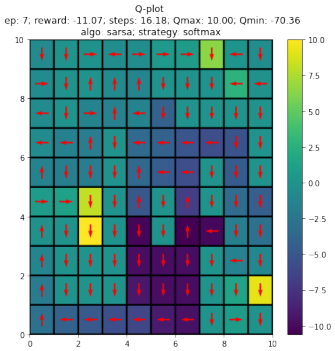
$$\beta_t = \max(10^{-3}, 0.99 \cdot \beta_{t-1})$$
$$\epsilon_t = \max(10^{-3}, 0.99 \cdot \epsilon_{t-1})$$

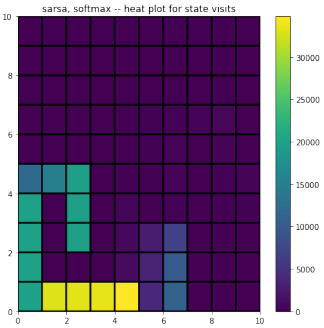
2 Plots

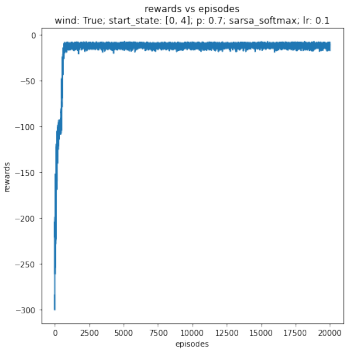
2.1 SARSA Softmax

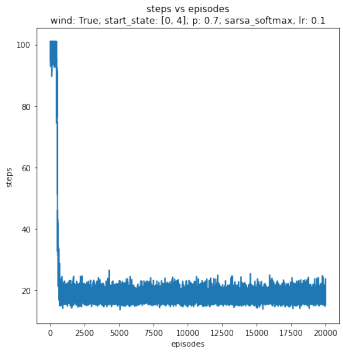


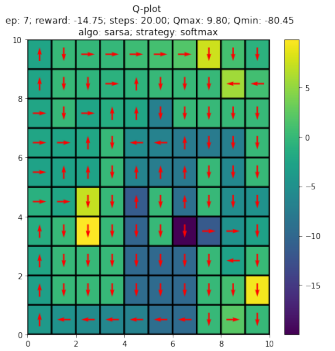


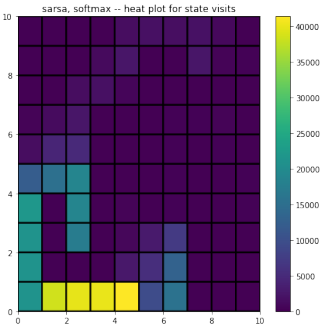


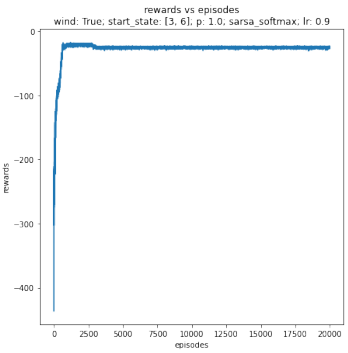


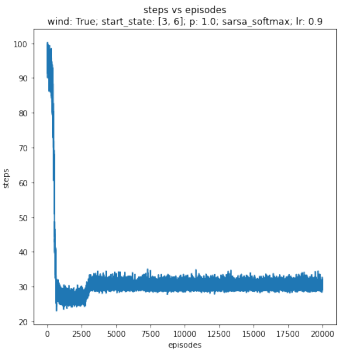


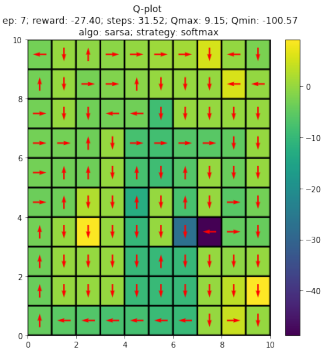


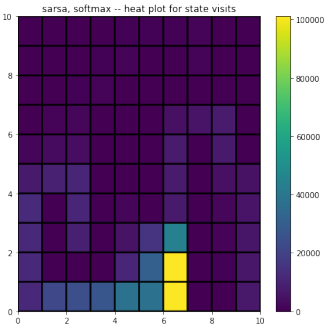


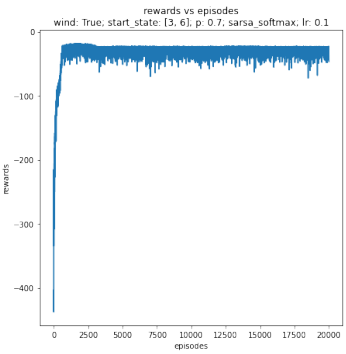


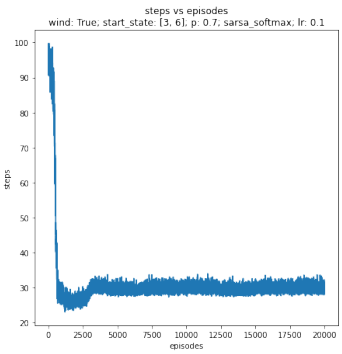


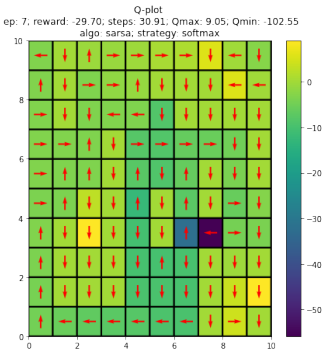


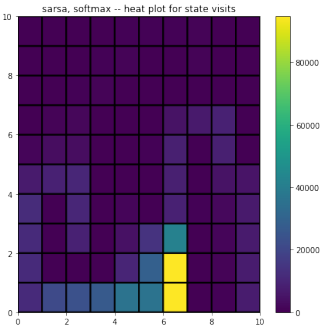


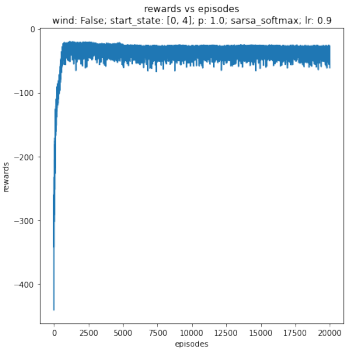


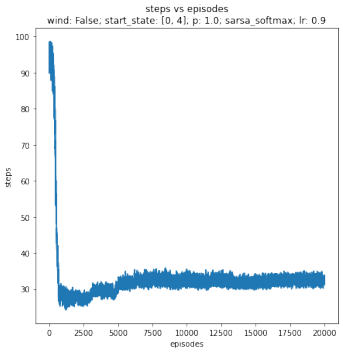


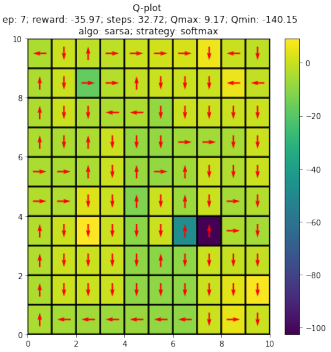


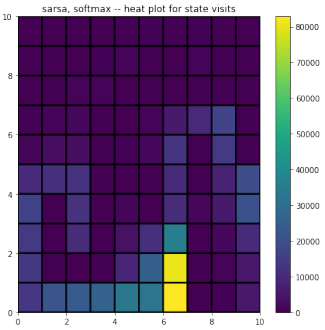


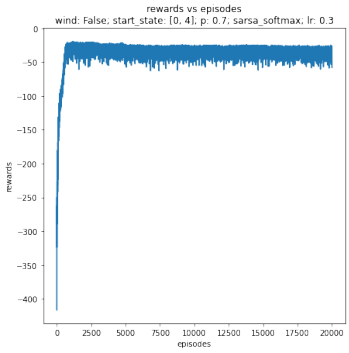


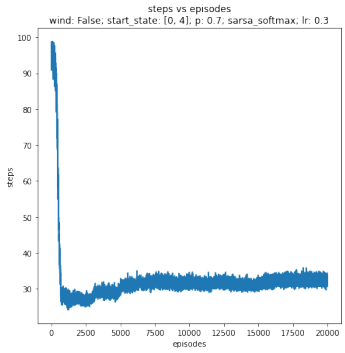

rewards vs episodes
wind: False, start_state: [0, 4], p: 1.0, sarsa_softmax, lr: 0.9
rewards
episodes

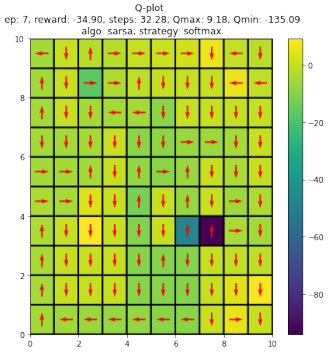

steps vs episodes
wind: False, start_state: [0, 4], p: 1.0, sarsa_softmax, lr: 0.9
steps
episodes

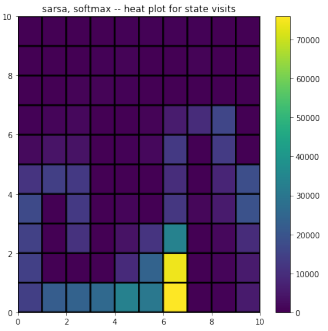

Q-plot
ep: 7, reward: -35.97, steps: 32.72, Qmax: 9.17, Qmin: -140.15
algo: sarsa; strategy: softmax

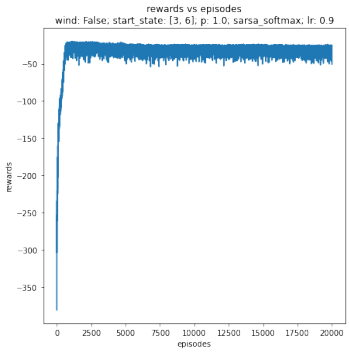

sarsa, softmax -- heat plot for state visits

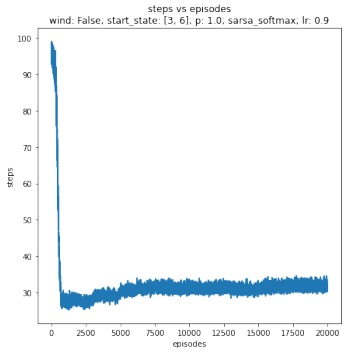

rewards vs episodes
wind: False, start_state: [0, 4], p: 0.7, sarsa_softmax, lr: 0.3
rewards
episodes

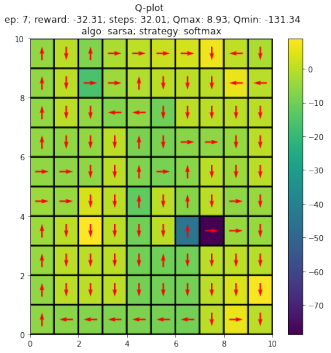

steps vs episodes
wind: False, start_state: [0, 4], p: 0.7, sarsa_softmax, lr: 0.3
steps
episodes

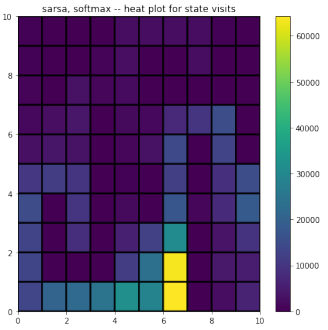

Q-plot
ep: 7, reward: -34.90, steps: 32.28, Qmax: 9.18, Qmin: -135.09
algo: sarsa; strategy: softmax

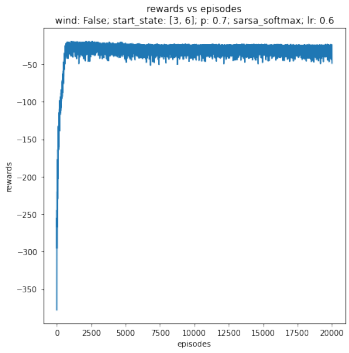

sarsa, softmax -- heat plot for state visits

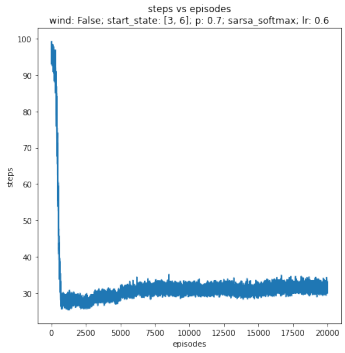

rewards vs episodes
wind: False, start_state: [3, 6], p: 1.0, sarsa_softmax, lr: 0.9
rewards
episodes

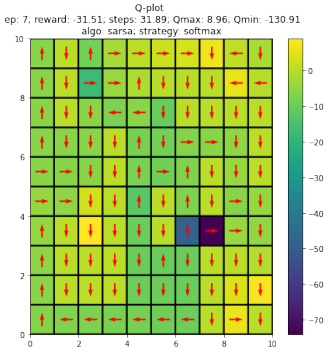

steps vs episodes
wind: False, start_state: [3, 6], p: 1.0, sarsa_softmax, lr: 0.9
steps
episodes

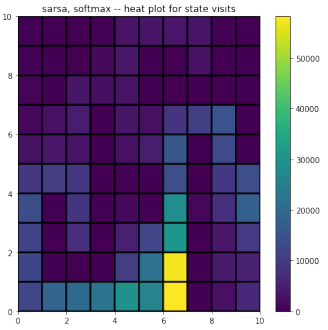

Q-plot
ep: 7, reward: -32.31, steps: 32.01, Qmax: 8.93, Qmin: -131.34
algo: sarsa; strategy: softmax


sarsa, softmax -- heat plot for state visits

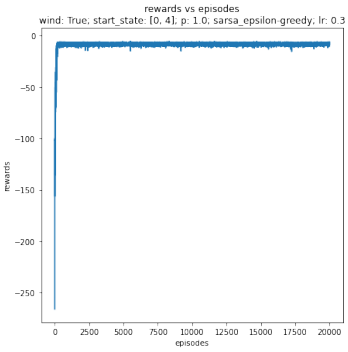

rewards vs episodes
wind: False, start_state: [3, 6], p: 0.7, sarsa_softmax, lr: 0.6
rewards
episodes

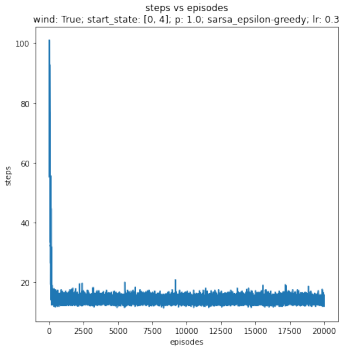

steps vs episodes
wind: False, start_state: [3, 6], p: 0.7, sarsa_softmax, lr: 0.6
steps
episodes

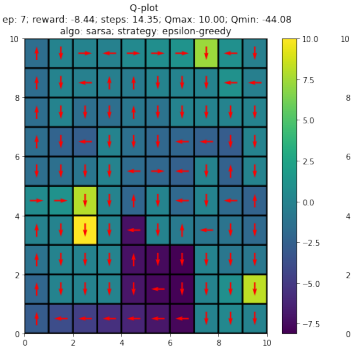

Q-plot
ep: 7, reward: -31.51, steps: 31.89, Qmax: 8.96, Qmin: -130.91
algo: sarsa; strategy: softmax

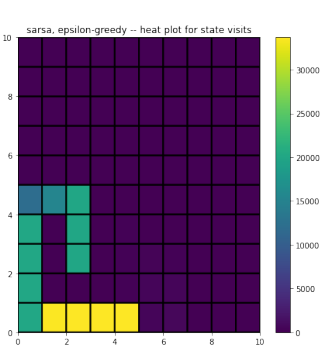

sarsa, softmax -- heat plot for state visits

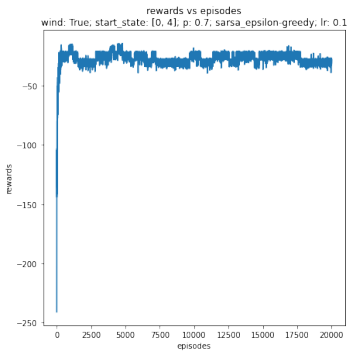
2.2 SARSA Epsilon-greedy

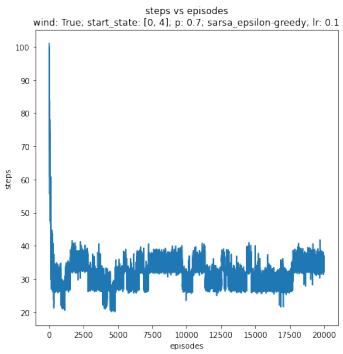


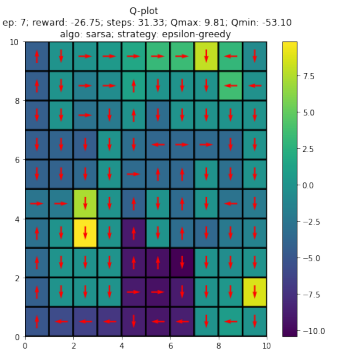


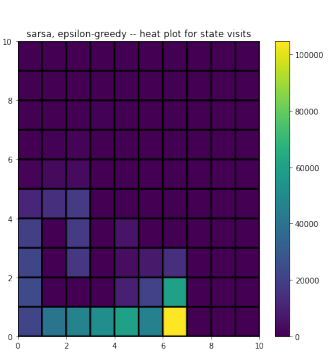


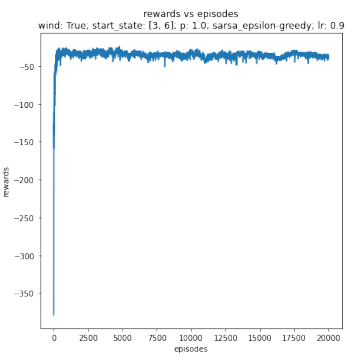


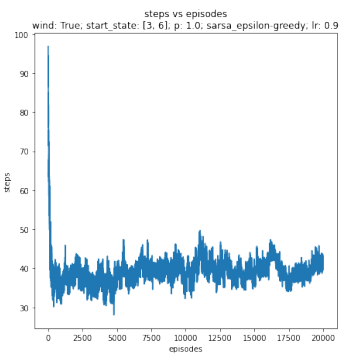


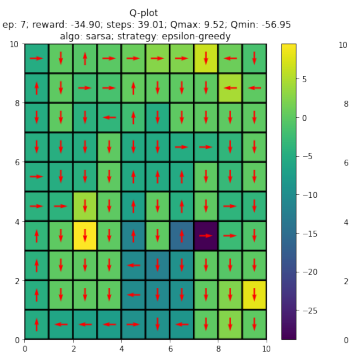


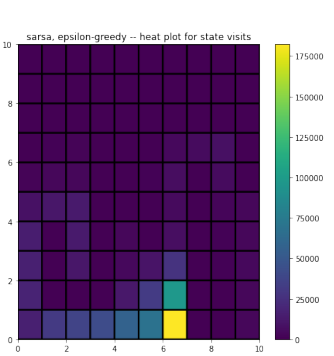


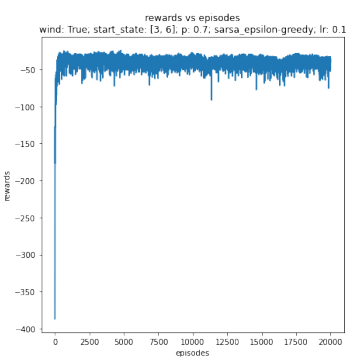


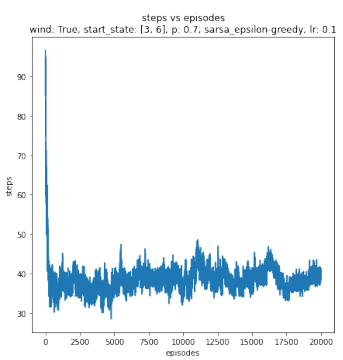


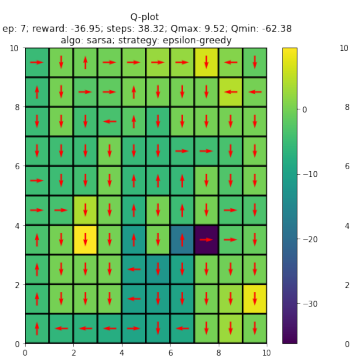


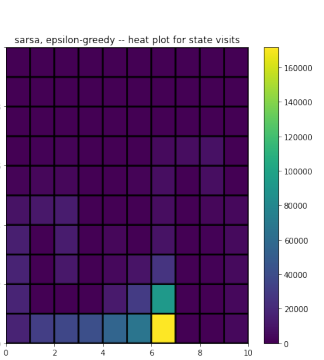


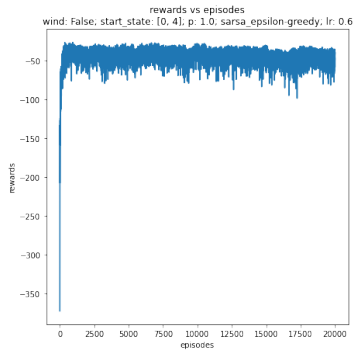


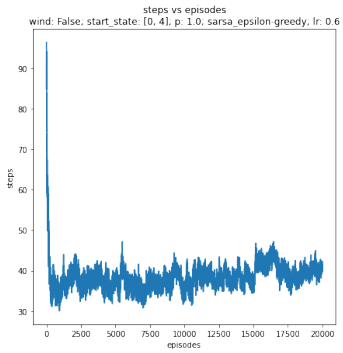


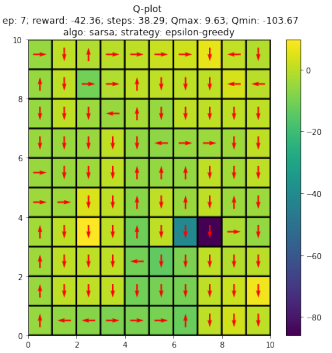


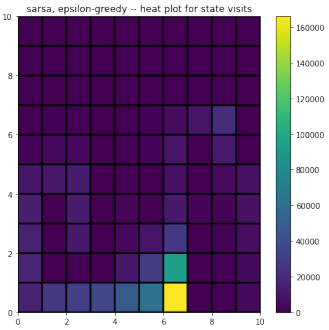


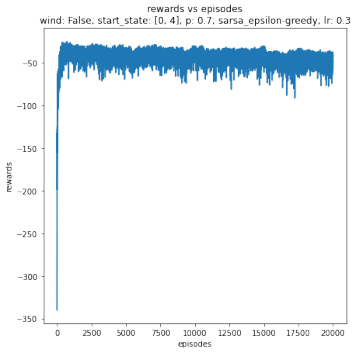


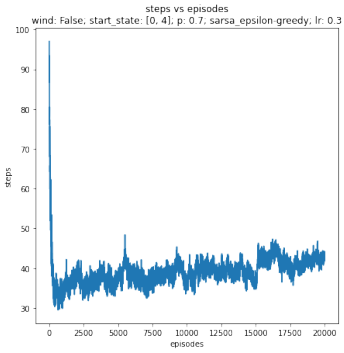

rewards vs episodes
wind: False, start_state: [0, 4], p: 1.0, sarsa_epsilon-greedy, lr: 0.6
rewards
episodes

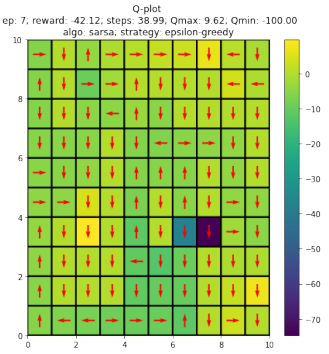

steps vs episodes
wind: False, start_state: [0, 4], p: 1.0, sarsa_epsilon-greedy, lr: 0.6
steps
episodes

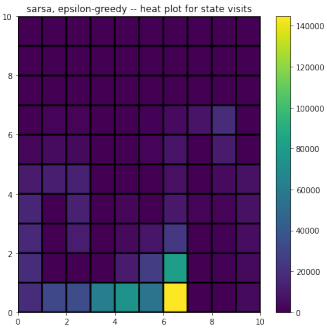

Q-plot
ep: 7, reward: -42.36, steps: 38.29, Qmax: 9.63, Qmin: -103.67
algo: sarsa, strategy: epsilon-greedy

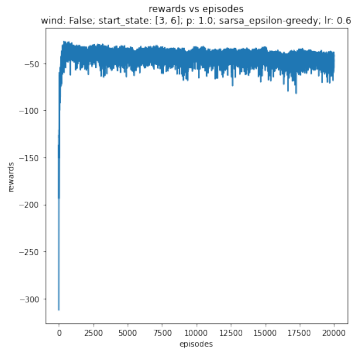

sarsa, epsilon-greedy -- heat plot for state visits

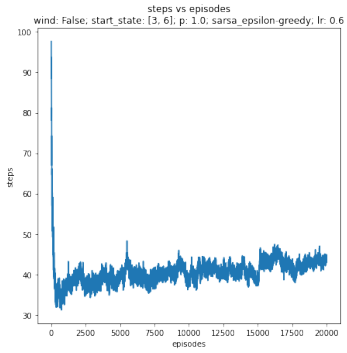

rewards vs episodes
wind: False, start_state: [0, 4], p: 0.7, sarsa_epsilon-greedy, lr: 0.3
rewards
episodes

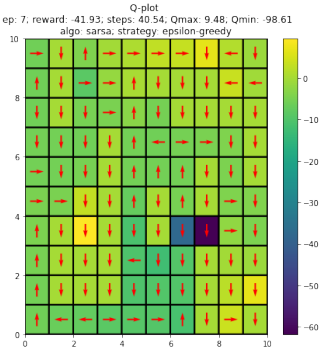

steps vs episodes
wind: False, start_state: [0, 4], p: 0.7, sarsa_epsilon-greedy, lr: 0.3
steps
episodes

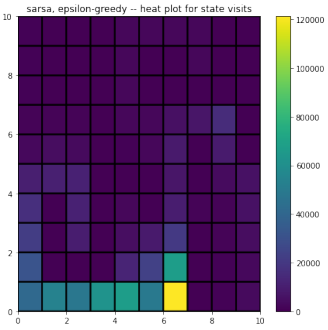

Q-plot
ep: 7, reward: -42.12, steps: 38.99, Qmax: 9.62, Qmin: -100.00
algo: sarsa, strategy: epsilon-greedy

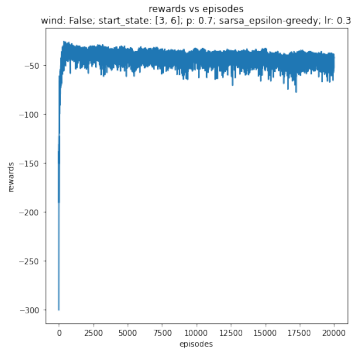

sarsa, epsilon-greedy -- heat plot for state visits

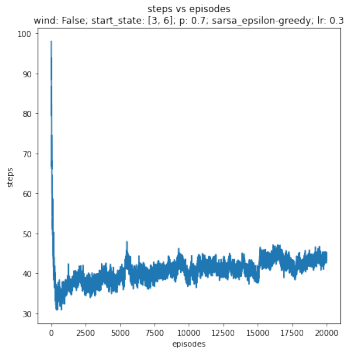

rewards vs episodes
wind: False, start_state: [3, 6], p: 1.0, sarsa_epsilon-greedy, lr: 0.6
rewards
episodes

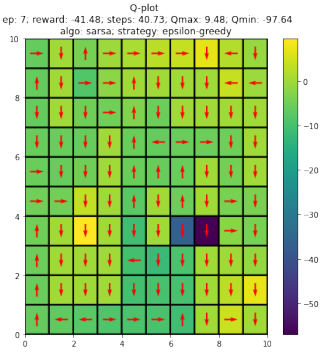

steps vs episodes
wind: False, start_state: [3, 6], p: 1.0, sarsa_epsilon-greedy, lr: 0.6
steps
episodes

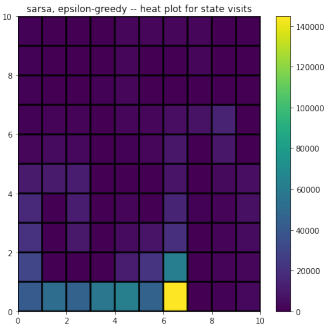

Q-plot
ep: 7, reward: -41.93, steps: 40.54, Qmax: 9.48, Qmin: -98.61
algo: sarsa, strategy: epsilon-greedy


sarsa, epsilon-greedy -- heat plot for state visits

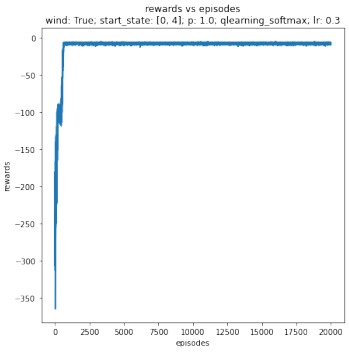

rewards vs episodes
wind: False, start_state: [3, 6], p: 0.7, sarsa_epsilon-greedy, lr: 0.3
rewards
episodes

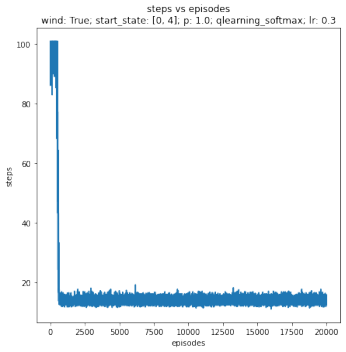

steps vs episodes
wind: False, start_state: [3, 6], p: 0.7, sarsa_epsilon-greedy, lr: 0.3
steps
episodes

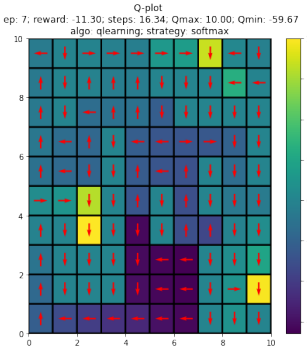

Q-plot
ep: 7, reward: -41.48, steps: 40.73, Qmax: 9.48, Qmin: -97.64
algo: sarsa, strategy: epsilon-greedy

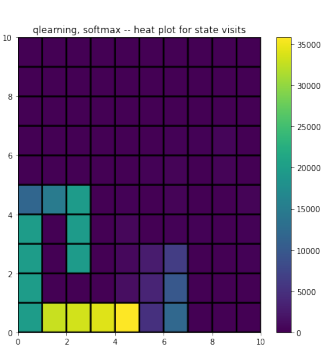

sarsa, epsilon-greedy -- heat plot for state visits

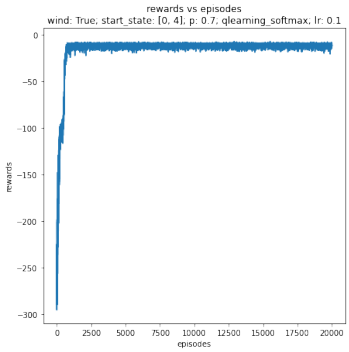
2.3 Q-Learning Softmax

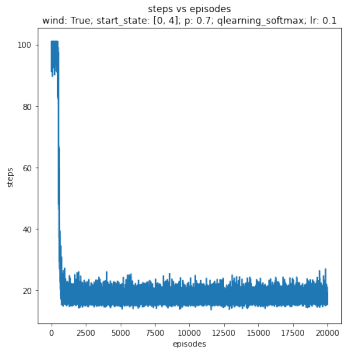


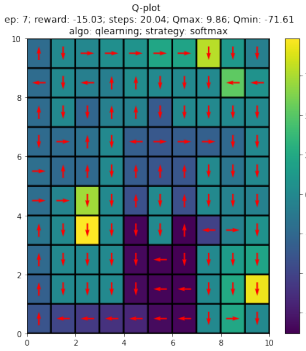


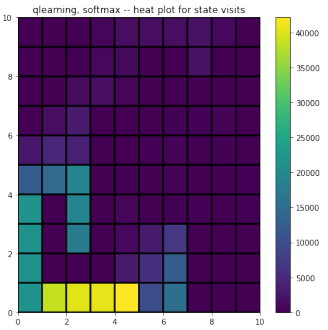


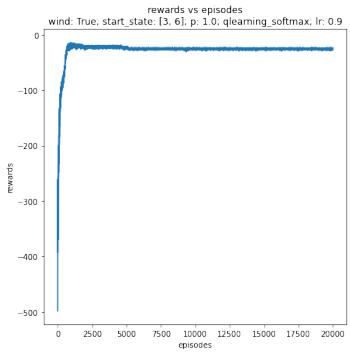


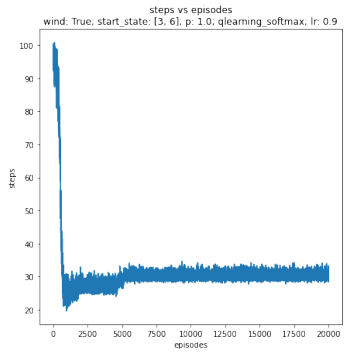


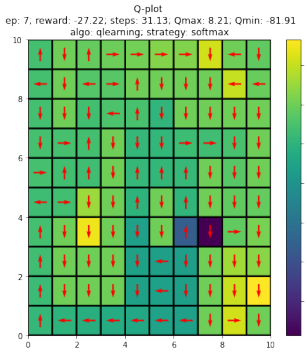


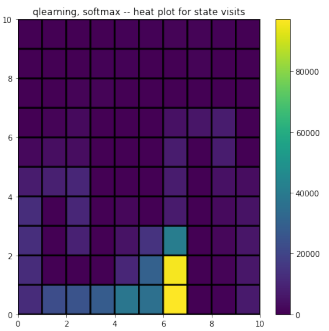


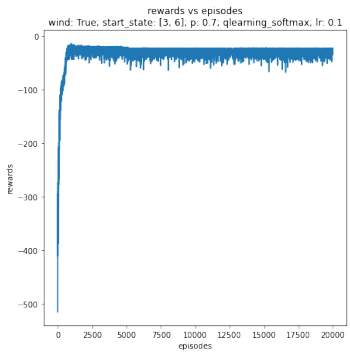


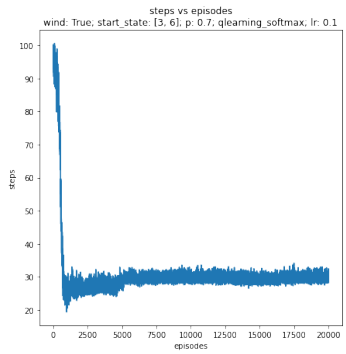


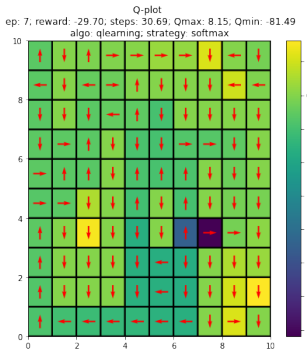


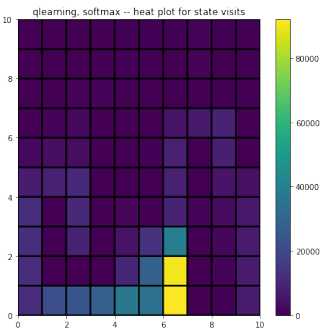


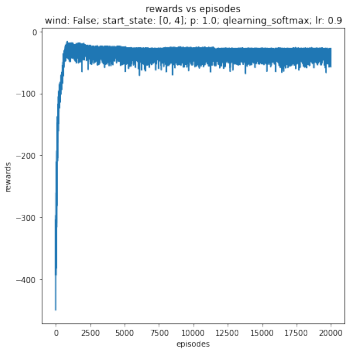


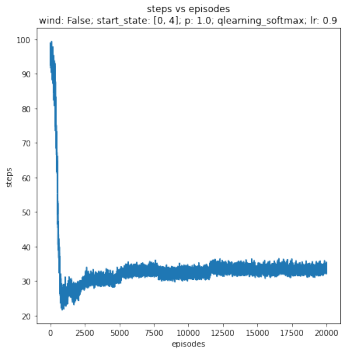


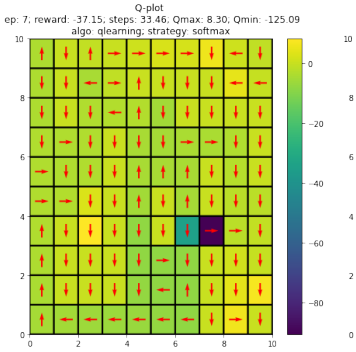


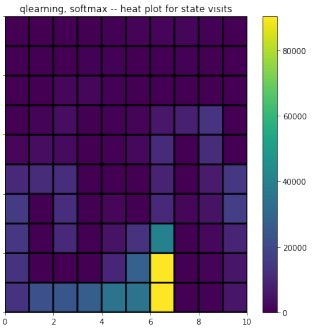


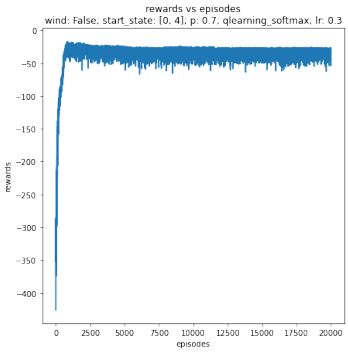


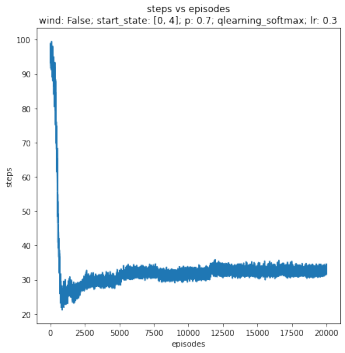

rewards vs episodes
wind: False, start_state: [0, 4], p: 1.0, qlearning_softmax, lr: 0.9
rewards
episodes

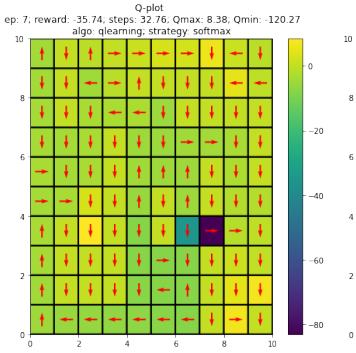

steps vs episodes
wind: False, start_state: [0, 4], p: 1.0, qlearning_softmax, lr: 0.9
steps
episodes

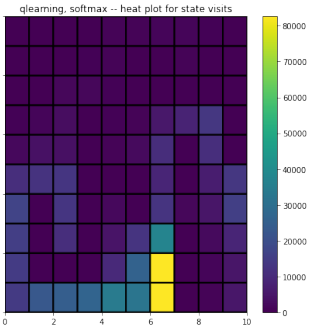

Q-plot
ep: 7, reward: -37.15, steps: 33.46, Qmax: 8.30, Qmin: -125.09
algo: qlearning, strategy: softmax

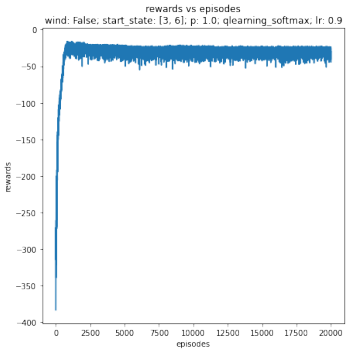

qlearning, softmax -- heat plot for state visits

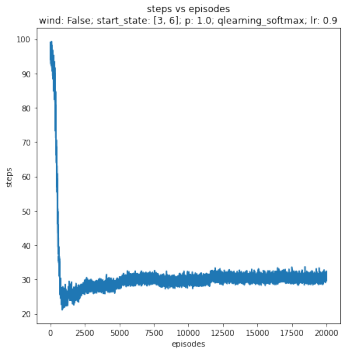

rewards vs episodes
wind: False, start_state: [0, 4], p: 0.7, qlearning_softmax, lr: 0.3
rewards
episodes

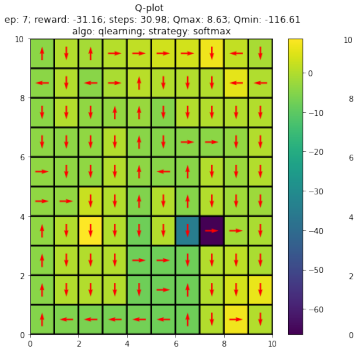

steps vs episodes
wind: False, start_state: [0, 4], p: 0.7, qlearning_softmax, lr: 0.3
steps
episodes

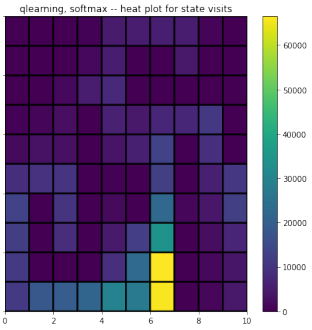

Q-plot
ep: 7, reward: -35.74, steps: 32.76, Qmax: 8.38, Qmin: -120.27
algo: qlearning, strategy: softmax

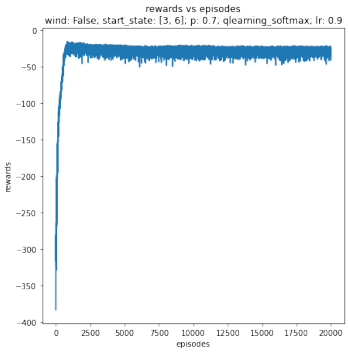

qlearning, softmax -- heat plot for state visits

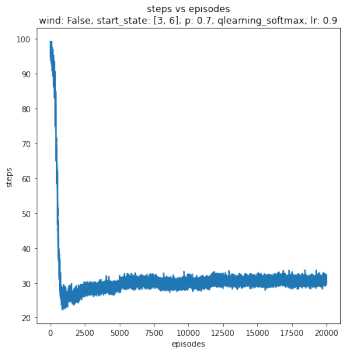

rewards vs episodes
wind: False, start_state: [3, 6], p: 1.0, qlearning_softmax, lr: 0.9
rewards
episodes

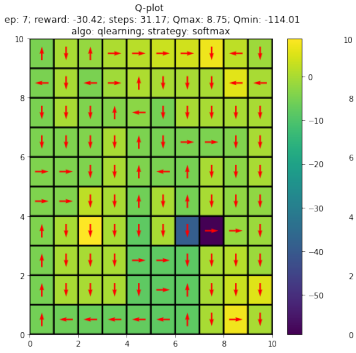

steps vs episodes
wind: False, start_state: [3, 6], p: 1.0, qlearning_softmax, lr: 0.9
steps
episodes

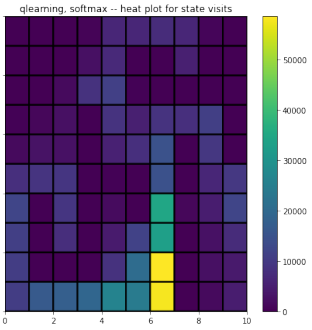

Q-plot
ep: 7, reward: -31.16, steps: 30.98, Qmax: 8.63, Qmin: -116.61
algo: qlearning, strategy: softmax


qlearning, softmax -- heat plot for state visits

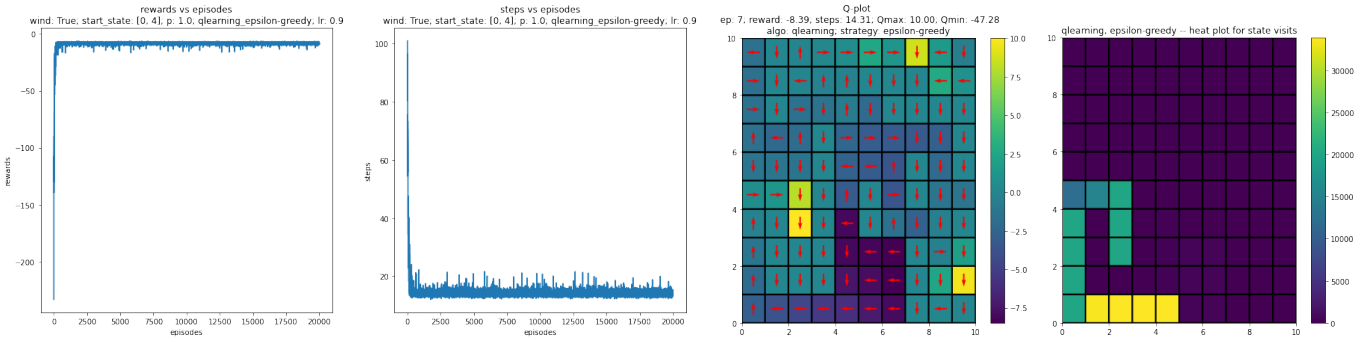

rewards vs episodes
wind: False, start_state: [3, 6], p: 0.7, qlearning_softmax, lr: 0.9
rewards
episodes

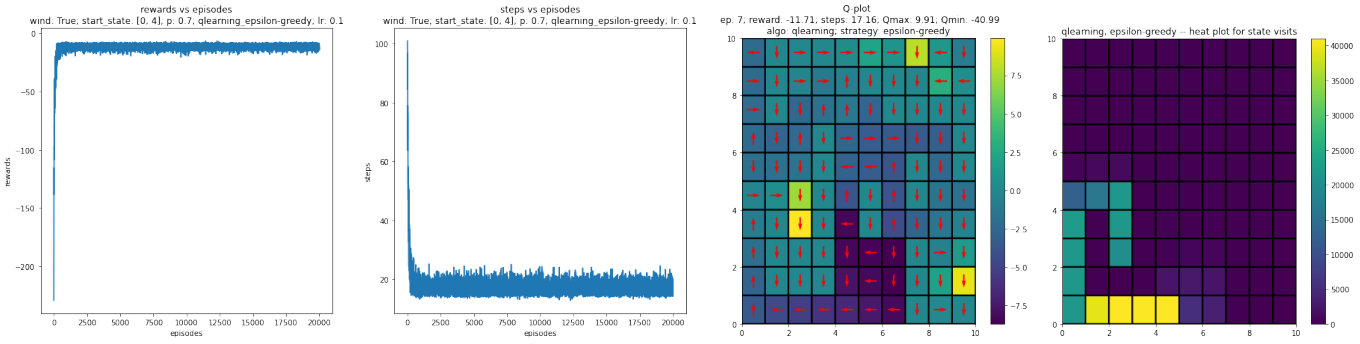

steps vs episodes
wind: False, start_state: [3, 6], p: 0.7, qlearning_softmax, lr: 0.9
steps
episodes

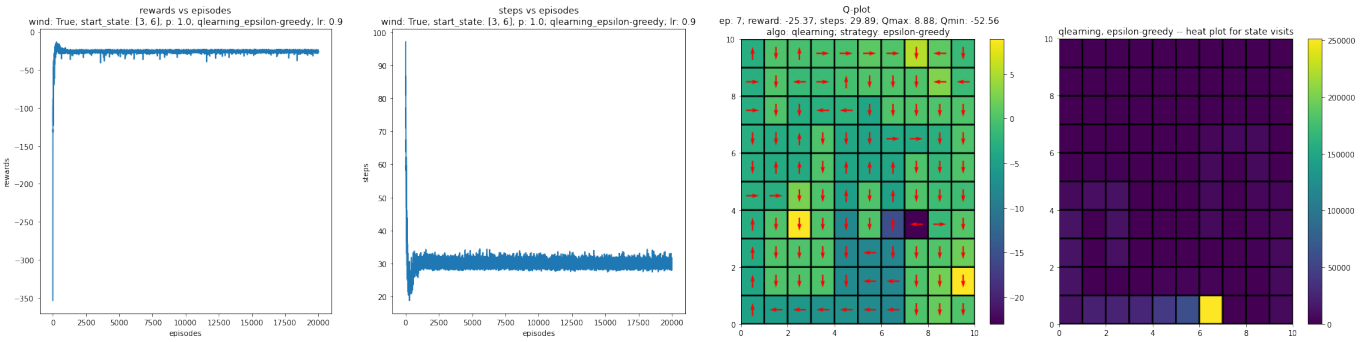

Q-plot
ep: 7, reward: -30.42, steps: 31.17, Qmax: 8.75, Qmin: -114.01
algo: qlearning, strategy: softmax

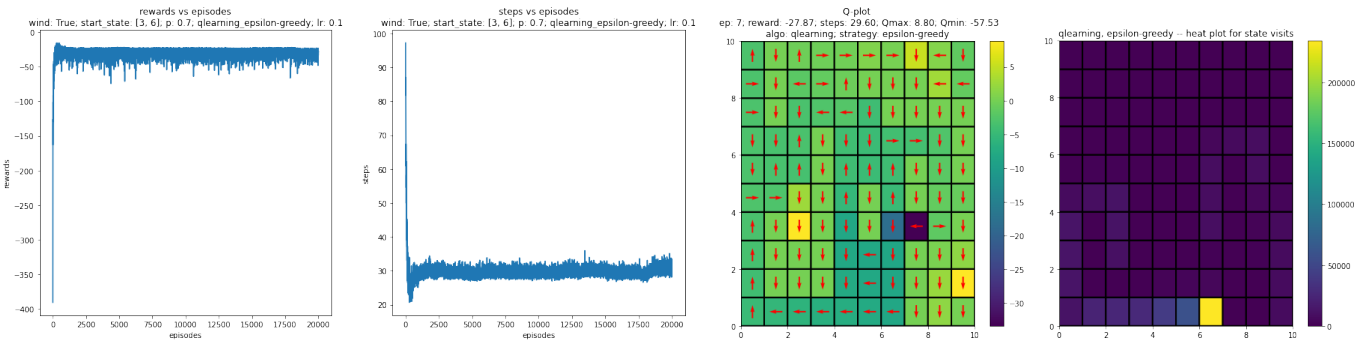

qlearning, softmax -- heat plot for state visits

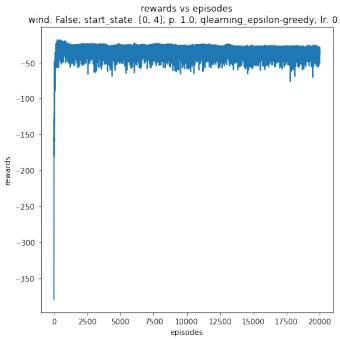
2.4 Q-Learning Epsilon-greedy

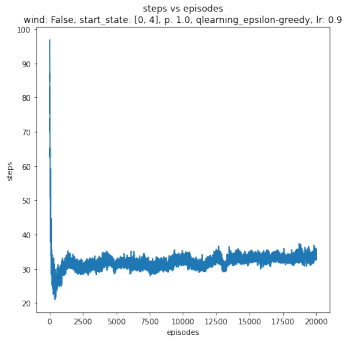


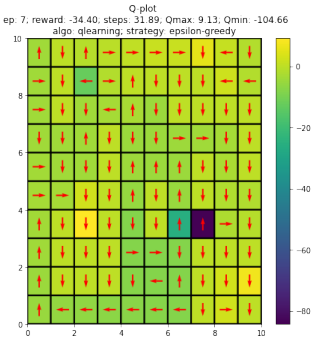


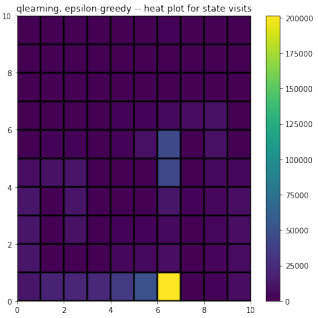


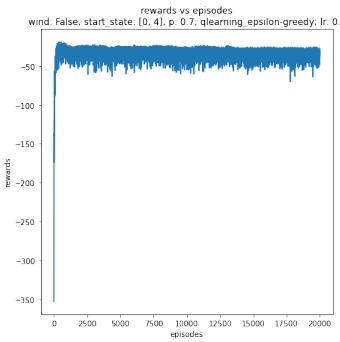


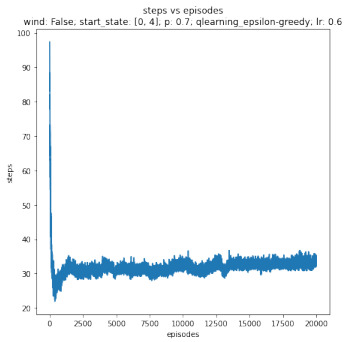

rewards vs episodes
wind: False, start_state: [0, 4], p: 1.0, qlearning_epsilon-greedy, lr: 0.9
rewards
episodes

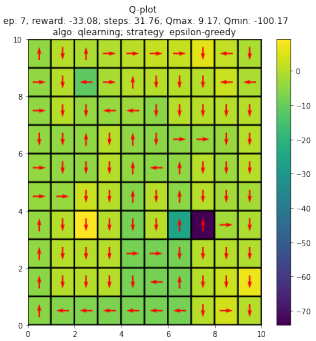

steps vs episodes
wind: False, start_state: [0, 4], p: 1.0, qlearning_epsilon-greedy, lr: 0.9
steps
episodes

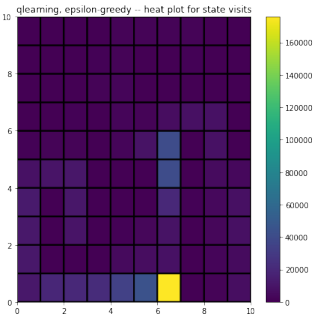

Q-plot
ep: 7; reward: -34.40; steps: 31.89; Qmax: 9.13; Qmin: -104.66
algo: qlearning; strategy: epsilon-greedy

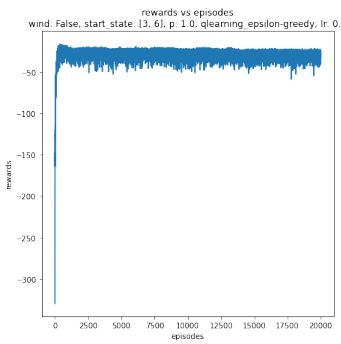

qlearning, epsilon-greedy -- heat plot for state visits

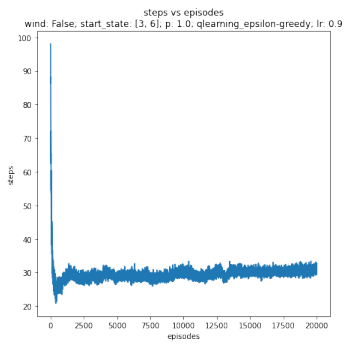

rewards vs episodes
wind: False, start_state: [0, 4], p: 0.7, qlearning_epsilon-greedy, lr: 0.6
rewards
episodes

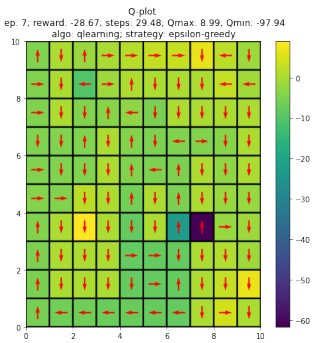

steps vs episodes
wind: False, start_state: [0, 4], p: 0.7, qlearning_epsilon-greedy, lr: 0.6
steps
episodes

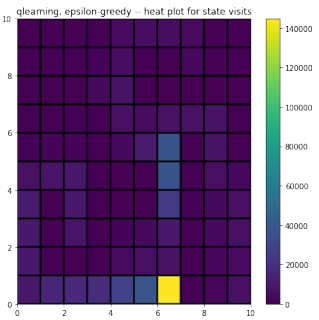

Q-plot
ep: 7; reward: -33.08; steps: 31.76; Qmax: 9.17; Qmin: -100.17
algo: qlearning; strategy: epsilon-greedy

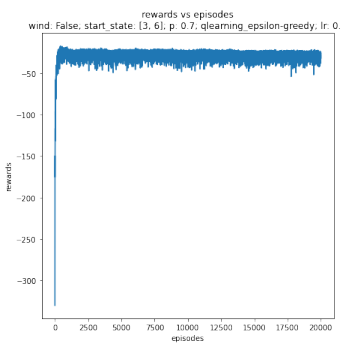

qlearning, epsilon-greedy -- heat plot for state visits

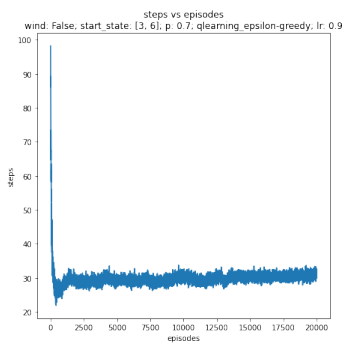

rewards vs episodes
wind: False, start_state: [3, 6], p: 1.0, qlearning_epsilon-greedy, lr: 0.9
rewards
episodes

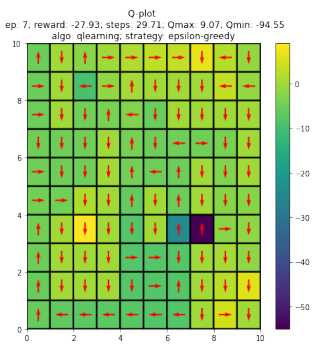

steps vs episodes
wind: False, start_state: [3, 6], p: 1.0, qlearning_epsilon-greedy, lr: 0.9
steps
episodes

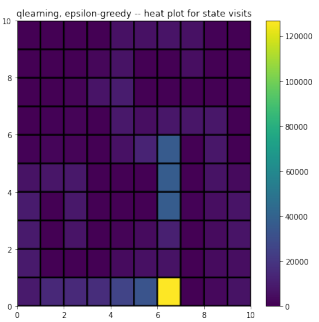

Q-plot
ep: 7; reward: -28.67; steps: 29.48; Qmax: 8.99; Qmin: -97.94
algo: qlearning; strategy: epsilon-greedy


qlearning, epsilon-greedy -- heat plot for state visits


rewards vs episodes
wind: False, start_state: [3, 6], p: 0.7, qlearning_epsilon-greedy, lr: 0.9
rewards
episodes


steps vs episodes
wind: False, start_state: [3, 6], p: 0.7, qlearning_epsilon-greedy, lr: 0.9
steps
episodes


Q-plot
ep: 7; reward: -27.93; steps: 29.71; Qmax: 9.07; Qmin: -94.55
algo: qlearning; strategy: epsilon-greedy


qlearning, epsilon-greedy -- heat plot for state visits

3 Observations

3.1 SARSA

1. If `p = 1`, irrespective of wind, start_state or strategy used, convergence is observed within an average of 40 steps.

2. If start_state is $[0, 4]$, it almost always converges to goal state $[2, 2]$ but encounters a bad-state in between. This may be because, during that trajectory the agent is "guided" on either side by obstacles and even if wind is present or we attempt to move off course because of bias, we still be on the same path.

3. For all policies `p != 1`, we take more steps to reach the goal, as 30% of the time, we move perpendicularly away from the trajectory. Because of this, we get restricted by the `max_steps` and may not converge often (as seen in state-visit-heat-map)

4. For policies starting at $[3, 6]$, we find that it's equally likely to reach goal states $[2, 2]$ or $[0, 9]$. Both these states are equi-distant from the $[3, 6]$ and surely encounter a bad state en-route.

5. For policies starting from $[3, 6]$, we find a restart state to the left of the start state. In case of `p = 1`, do not enter the restart state, and move up/down (right is blocked) and do not incur any high penalty. But when we have stochasticity, we drift to the left with a probability 0.15 and earn a high penalty. This cannot be avoided as the only two actions that can be taken in that state are up/down.

6. Epsilon-greedy strategy shows more fluctuations (w.r.t steps) than its softmax counter-part. We guess this is because of the higher epsilon value, i.e 0.2 we started with.

7. The only time the agent was able to identify the $[8, 7]$ goal state is when, we have a completely deterministic environment (no wind, and `p=1`) and a high learning_rate. This maybe because, without stochasticity we can reach goal states without "wasting" too many steps (like drifting to unwanted states). This gives us a chance to explore farther states and reach distant goal states.

8. With any strategy and environment condition, the agent learns a trajectory to goal state $[2, 2]$ very quickly, within 1000 steps.

3.2 Q-Learning

1. Policy learnt by the agent depends on the start state and the evaluated Q values. When the start state is $[0, 4]$ and there is wind, agent converged to the nearest goal state $[2, 2]$. And when there is no wind, agent could reach both $[2, 2]$ and $[0, 9]$. When the start state is $[3, 6]$, agent was able to reach all the 3 goal states depending upon the Q-values.

2. In the initial episodes of training, when the agent hasn't learnt the optimal policy yet, each episode was ended at 100 steps. This can be seen in the steps vs episodes plots.

3. Q-values of a state indicate how probable a state can lead the agent to one of the goal sates in the least number of steps. As such, states just before the 3 goal states have high Q-values. This can be observed in the Q-values plot.

4. In rewards vs episodes plot, we can see that fluctuations after convergence is much higher when the stochasticity (p) = 0.7 compared to p = 1. Similar differences in the fluctuations can also be observed in the steps vs episodes plot.

5. When the stochasicity (p) is lower, then the best learning rate hyper parameter is also lower. This is because when `p = 0.7`, there is some random exploration by the agent going on. A very high learning rate in such a case can reduce the reward earned. So, a lower learning rate turned out to be a better hyper parameter as it leads to a more careful approach.

6. In the `rewards vs episodes` and `steps vs episodes` plots, there are different stretches at different convergence levels. These may correspond to multiple goal states reached, each with a different reward and steps convergence level. We can verify the goal states reached from the heat plot for the state visits.

7. With any strategy and environment condition, the agent learns a trajectory to goal state $[2, 2]$ very quickly, within 1000 steps.