Contents

Objective 1
Environment 1
Learning Algorithms : 2
Parameters: 3
Evaluation Metrics : 4
Hyperparameter Tuning : 4
Computation: 5
Plots 5
Softmax Iteration-2 54
Inferences: 79
Conclusion: 79

CS6700 - Reinforcement Learning
Programming Assignment 1
Team: **Leon Davis(ED21B038), Anirud N(CE21B014)**

Objective

This assignment explores the two Temporal Difference algorithms - Q Learning and Sarsa. We will be solving several variants of the Grid world problem. For each of this variant, we find the optimal hyperparameters for training, for each of the algorithms - Sarsa and Q Learning, and for each of the exploration strategies: Epsilon Greedy and Softmax.

Environment

Grid world
Variables:

```
num_cols = 10
num_rows = 10
obstructions = np.array([[0,7],[1,1],[1,2],[1,3],[1,7],[2,1],[2,3],
                         [2,7],[3,1],[3,3],[3,5],[4,3],[4,5],[4,7],
                         [5,3],[5,7],[5,9],[6,3],[6,9],[7,1],[7,6],
                         [7,7],[7,8],[7,9],[8,1],[8,5],[8,6],[9,1]])
bad_states = np.array([[1,9],[4,2],[4,4],[7,5],[9,9]])
restart_states = np.array([[3,7],[8,2]])
goal_states = np.array([[0,9],[2,2],[8,7]])
```

Rewards: -1 for normal states,-100 for restart states,-6 for bad states, +10 for goal states.

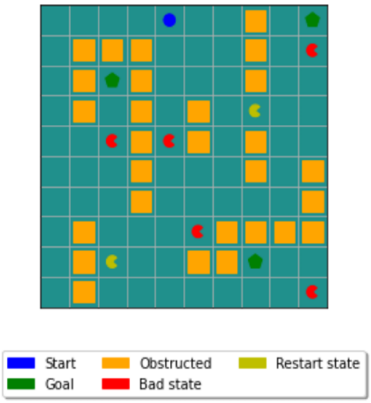


Figure 2: An example grid with start point at (0, 4)

Variants of stochasticity in the environments:

1. Start = [3,6], `wind=False`(clear), p = 1.0(deterministic step).
2. Start = [3,6], `wind=False`(clear), p = 0.7(stochastic step).
3. Start = [3,6], `wind=True`(windy), p = 1.0(deterministic step).
4. Start = [0,4], `wind=False`(clear), p = 1.0(deterministic step).
5. Start = [0,4], `wind=False`(clear), p = 0.7(stochastic step).
6. Start = [0,4], `wind=True`(windy), p = 1.0(deterministic step).

State transition probability:
The agent transitions to the next state determined by the direction of the action chosen with a probability of $p \in [0, 1]$.

Bias:
We also define a parameter called $b \in [0, 1]$. The agent transitions to the state West of the chosen action with probability $(1 - p) \times b$, and to the East of the chosen action with probability $(1 - p) \times (1 - b)$.

Wind:
The environment may also have a wind blowing that can push the agent one additional cell to the right after transitioning to the new state with a probability of 0.4.

Learning Algorithms :

In this assignment, we have run these six experiments for two Learning algorithms, ie

1) **Sarsa** (On policy)
2) **Q-learning**(Off policy)

Sarsa (On policy) :
In this algorithm, the learning agent learns and updates the value function based on the present action determined by the currently employed policy
The update equation for Sarsa is given by :
$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$

Q-Learning (Off Policy):
In this algorithm, the learning agent acquires knowledge of the value function based on the action derived from an alternative policy.
The update equation for Q learning is given by :
$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$

Parameters:

1)Temperature(τ):
The temperature parameter is a value that scales the input to the softmax function before applying the softmax operation. In this case, the Q-values are divided by a temperature parameter before applying the softmax operation. A higher temperature encourages more exploration, while a lower temperature favours exploitation.

2)Epsilon(ϵ):
- With probability $1-\epsilon$, the agent selects the action that is estimated to be the best based on its current knowledge or policy. This is the exploitation part of the strategy.
- With probability ϵ, the agent selects a random action from the set of possible actions. This is the exploration part of the strategy, allowing the agent to discover potentially better actions or to gain more information about the environment.

The value of ϵ determines the balance between exploration and exploitation. A high value of ϵ encourages more exploration, which can be beneficial early in the learning process when the agent's knowledge about the environment is limited.

3)Learning rate(α):
The learning rate (α) controls the magnitude of the updates to the Q-values. A higher learning rate allows for faster updates but may lead to instability or divergence, while a lower learning rate provides more stability but slower convergence.

4)Discount factor(γ):
The discount factor (γ) is a parameter used to discount future rewards. It determines the importance of future rewards relative to immediate rewards.
- A discount factor 0 means that the agent only considers immediate rewards and ignores all future rewards.

- A discount factor of 1 means that the agent considers all future rewards equally without discounting.

Evaluation Metrics :

To evaluate and compare various hyperparameters for each of the experiments, The following metrics were considered:

1) Average rewards over all the episodes of training (maximize)
2) The average number of steps over all episodes of training (minimize)
3) Combination of 1) and 2)

Average rewards were chosen over intermediate steps because the Sarsa algorithm, while training, takes conservative steps to achieve the goal state, so it might have a more significant number of steps when compared to Q learning.

It was realized that a combination of 1 and 2 would require normalizing average rewards, average number of steps, and importance factors associated with each for a weighted addition to obtaining the final objective function.
However, such normalisation would require us to estimate the distribution of each of these, so it was decided to go with 1) as an evaluation metric, as we need more information about the distributions of average rewards and average steps to normalize.

So, we have chosen **Average Rewards** as evaluation metrics.

Hyperparameter Tuning :

We have used **Bayesian Optimization** to find the optimal parameters for each experiment.
This was chosen over the Brute Force grid search algorithm because the later searches only at discrete points mentioned. It doesn't estimate the function mapping between the evaluation metrics and the hyperparameters.

Bayesian Optimization takes a few random samples of hyperparameters and tries to fit a curve, then takes more iterations to validate and improve the curve to fit the cover at maximal and minimal. It intelligently selects points to explore in the parameter space rather than exhaustively searching over a grid.
Bayesian Optimization was also faster and more accurate when compared to the Brute Force approach, and it offered more flexibility to search over an extensive range of values.

It can be structured as follows
Objective Function:
The objective function measures the model's performance (or some criterion) with respect to these hyperparameters. Here, the objective function is Average rewards.
Bayesian Optimization Process:
The optimization is formulated as a probabilistic function. A surrogate probabilistic model is created based on the limited available data points (objective function evaluations). This provides the probabilistic estimate of the objective function.
Acquisition Function:
This allows the algorithm to explore different points as it decides what points to estimate next.

After evaluating each point, the surrogate function is updated with the new data point. And the Acquisition function is reevaluated.

This process continues until the specified number of iterations. The more the number of iterations, more closer the estimated curve of hyperparameters vs objective function is with the actual unknown function between the same.
In this algorithm, we have set the number of iterations to be 29, where the first nine iterations are for exploration, i.e., random sample points drawn, followed by 20 iterations following the Bayesian model, as around these number of iterations, the max value of objective function across various combinations of hyperparameters estimated seemed close to the required.
This took several hours of training to get the final values.
Reference: https://github.com/bayesian-optimization/BayesianOptimization
(we used the pre-existing library to perform the optimization)

Computation:

Each experiment was run for 5001 training episodes and averaged over five runs.
The plots were reported, and the hyperparameter tuning was done by averaging over five runs of the same experiment.
Each experiment took several hours of training, so we parallelized the optimization to speed up the task.
In other words, we parallelly ran 24 tasks to find the optimal hyperparameters for each task
1. Start = [3,6], `wind=False`(clear), p = 1.0(deterministic step).
2. Start = [3,6], `wind=False`(clear), p = 0.7(stochastic step).
3. Start = [3,6], `wind=True`(windy), p = 1.0(deterministic step).
4. Start = [0,4], `wind=False`(clear), p = 1.0(deterministic step).
5. Start = [0,4], `wind=False`(clear), p = 0.7(stochastic step).
6. Start = [0,4], `wind=True`(windy), p = 1.0(deterministic step).

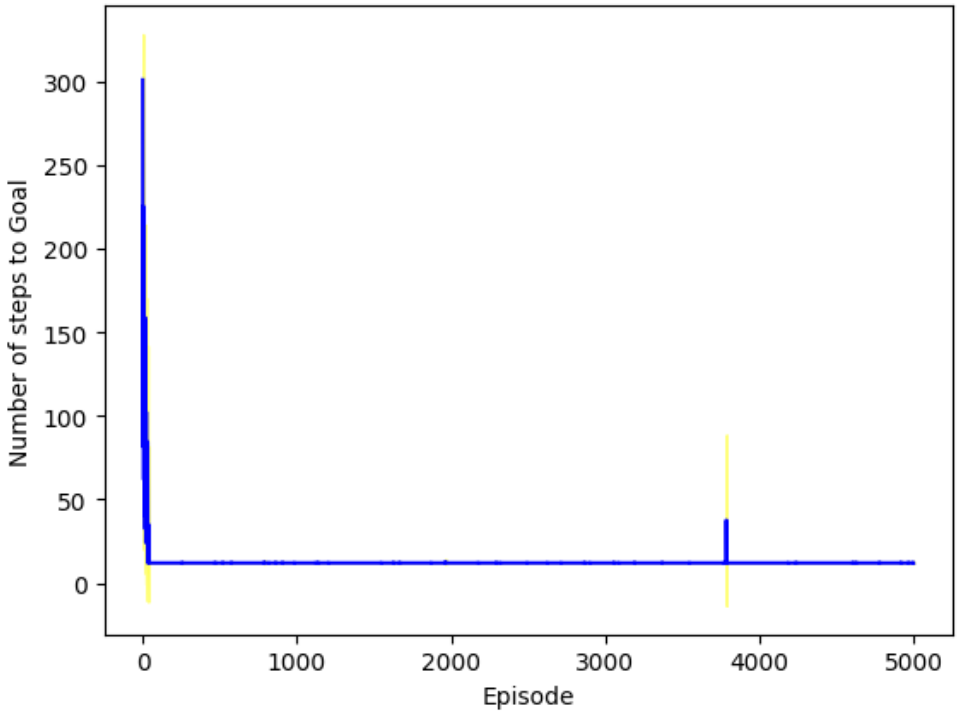
Each of these six tasks was run for both Sarsa and learning algorithms and over epsilon greedy and softmax algorithms - totally summing up to 24 tasks of finding the optimal hyperparameters for different algorithms(sarsa and q learning) and different exploration-exploitation strategies (epsilon greedy and softmax). All these 24 task implementations, i.e., files, are included in the zip file.

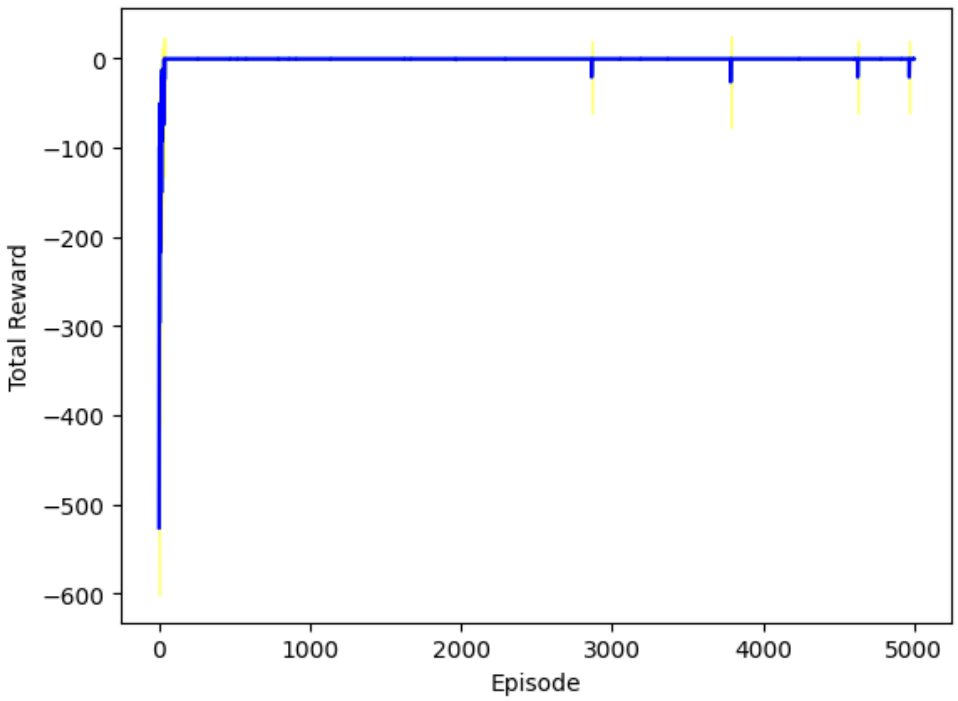
Plots

The **dark blue lines** in the reward vs. episodes and the number of steps vs. episodes indicate the mean values, whereas the **yellow lines** in the plot indicate the range of values within one standard deviation of the mean. (mean + or - standard deviation)

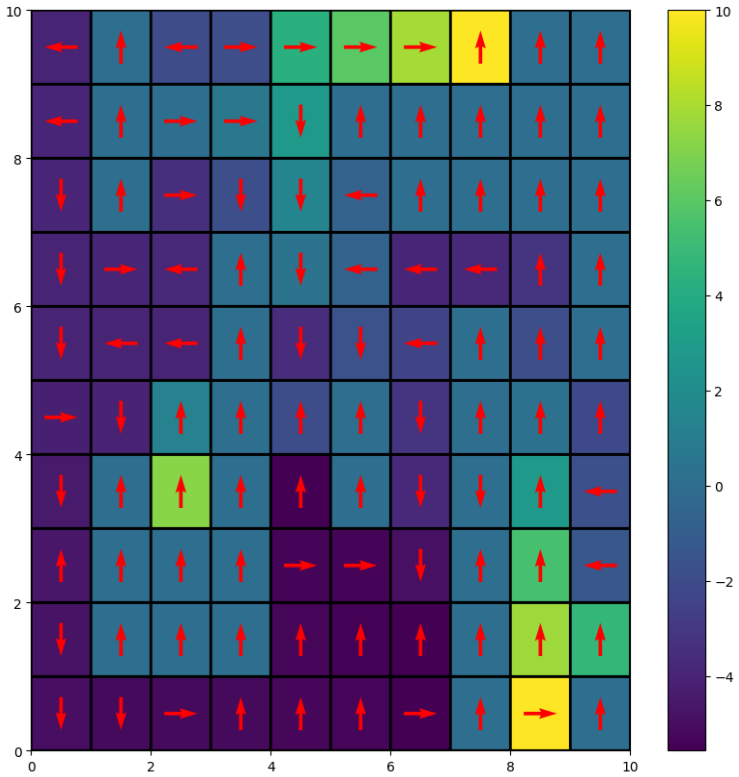
Plots for SARSA

Wind=False **P**=1 **Epsilon Greedy** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Epsilon**= 0.001

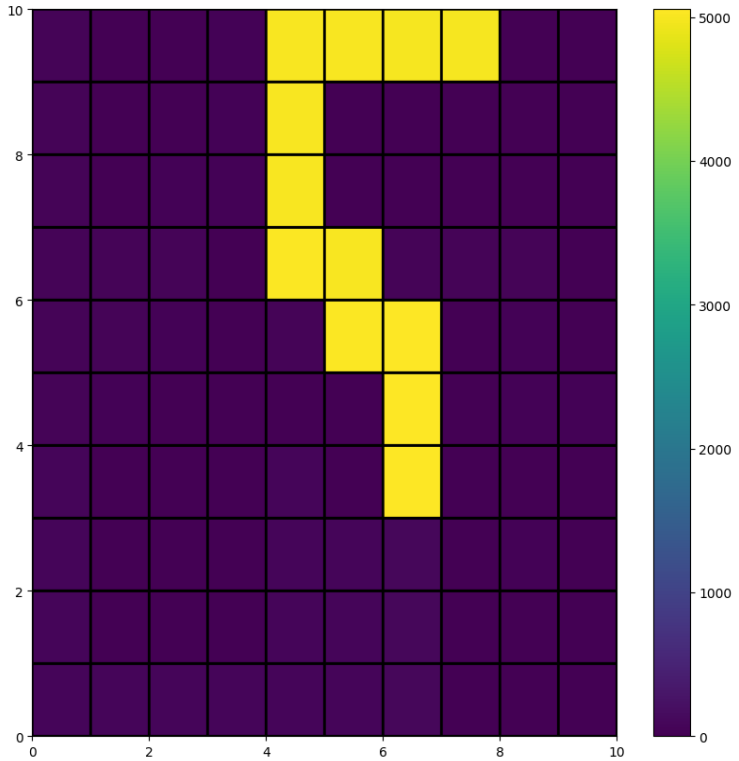




Episode 5000: Reward: -1.000000, Steps: 12.00, Qmax: 10.00, Qmin: -47.49

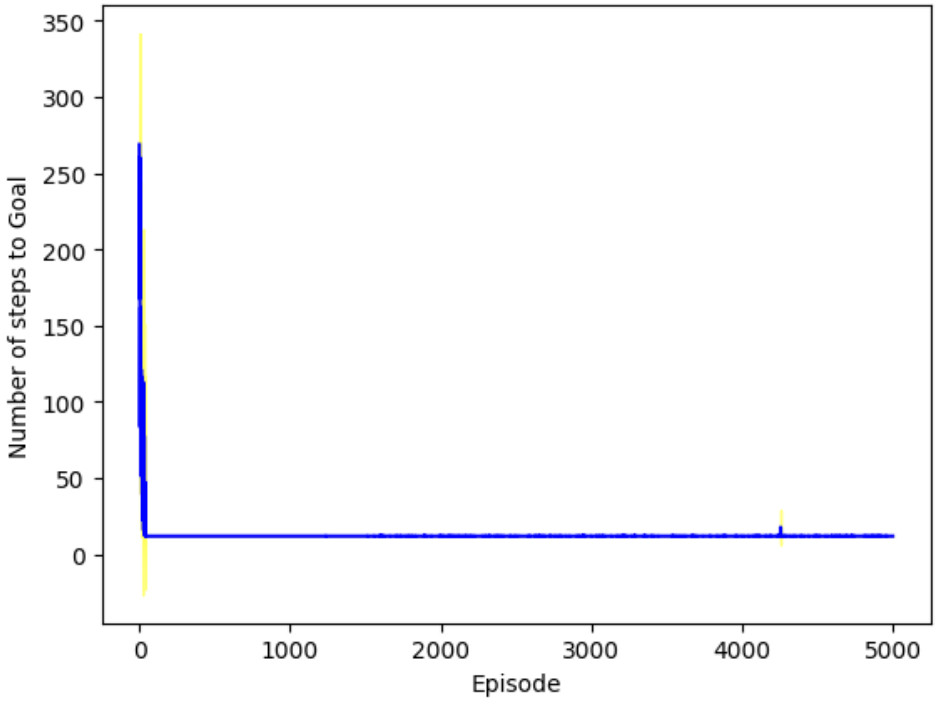


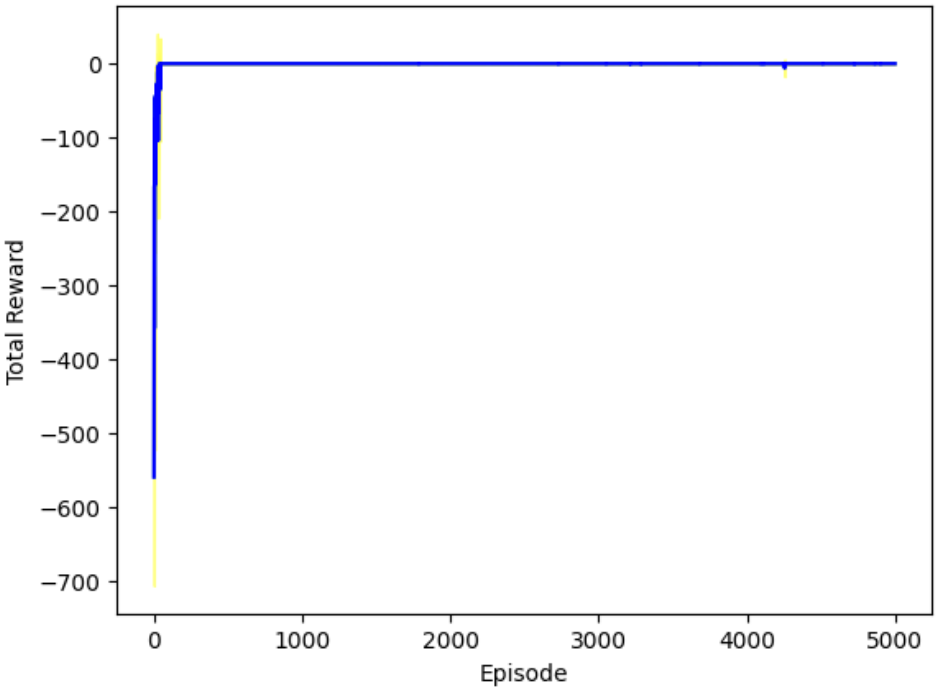
State Visit Count Plot



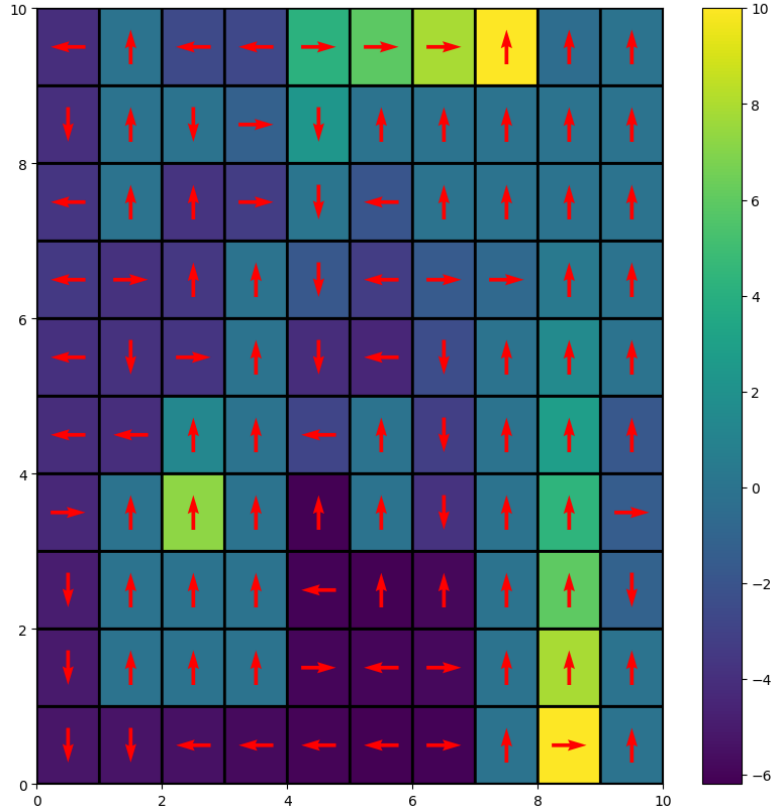
Plots for SARSA

Wind=False **P**=1 **Softmax** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.214

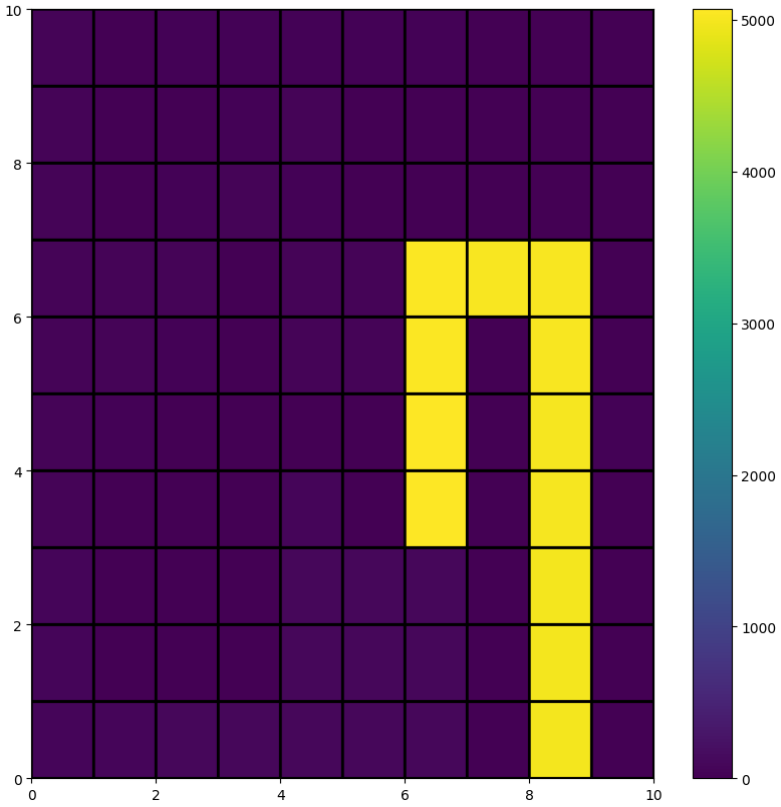




Episode 5000: Reward: -1.000000, Steps: 12.00, Qmax: 10.00, Qmin: -47.49

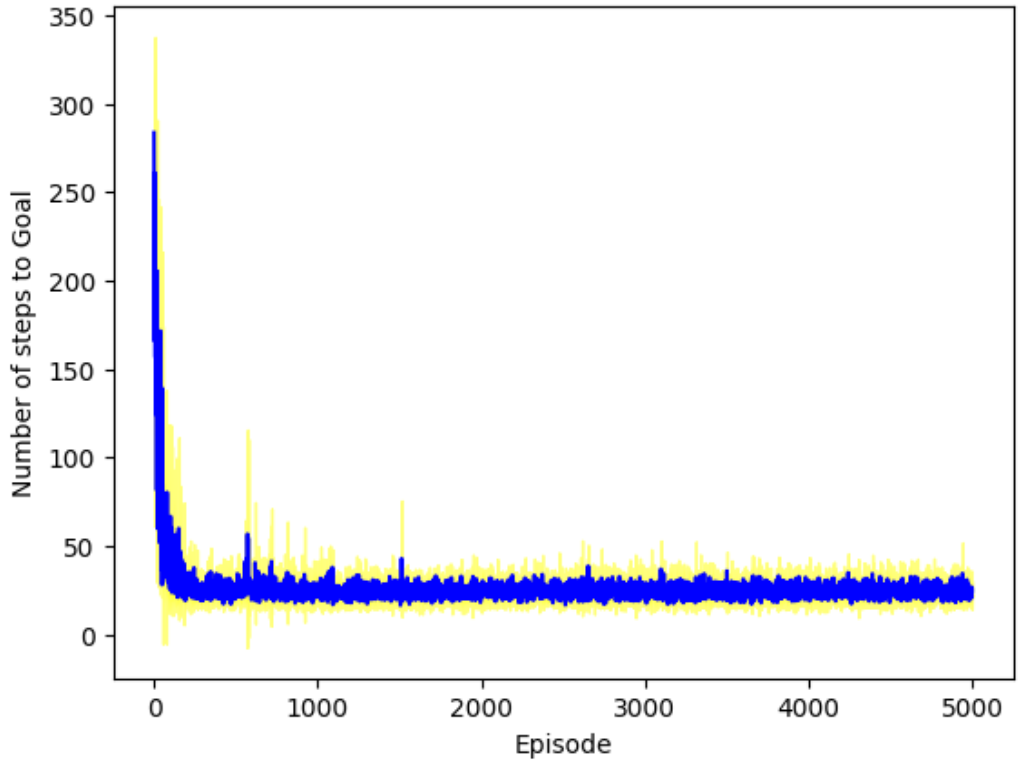


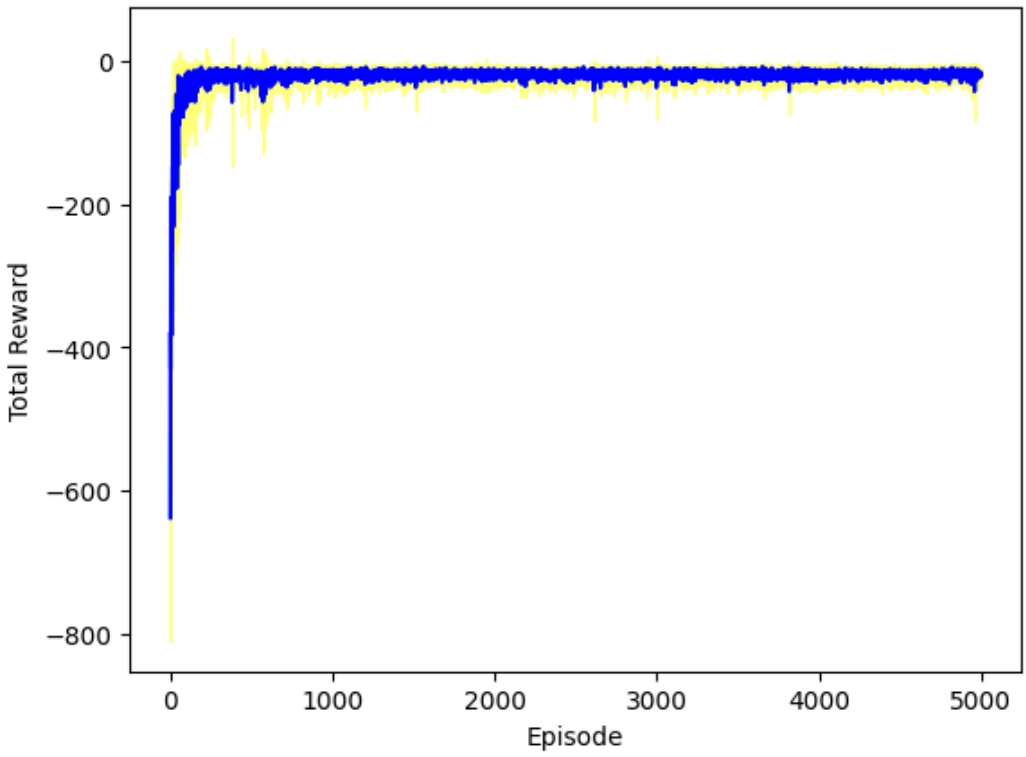
State Visit Count Plot



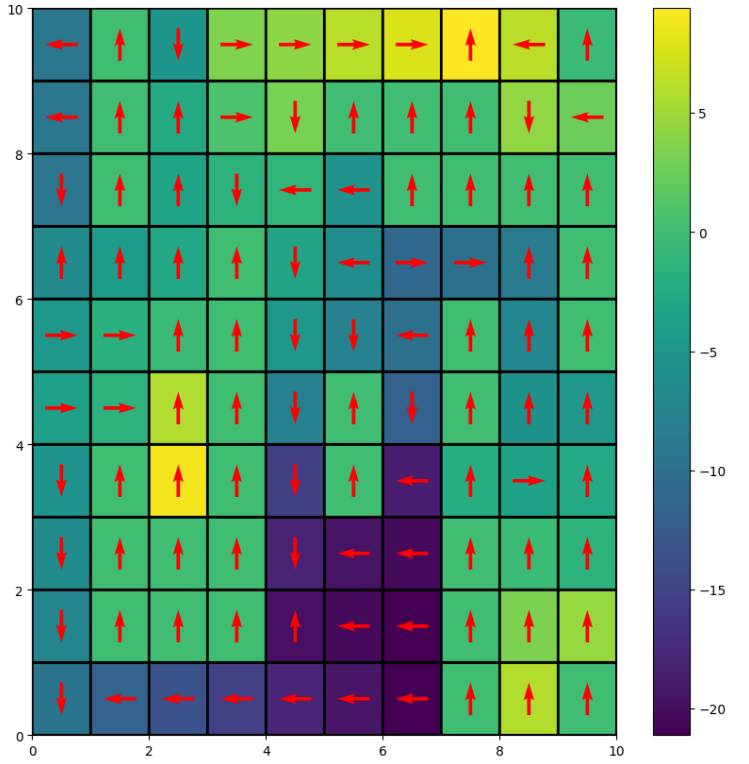
Plots for SARSA

Wind=False **P**=0.7 **Epsilon Greedy** **Start State**=[3,6]
Alpha= 0.222 **Gamma**=0.999 **Epsilon**= 0.0001

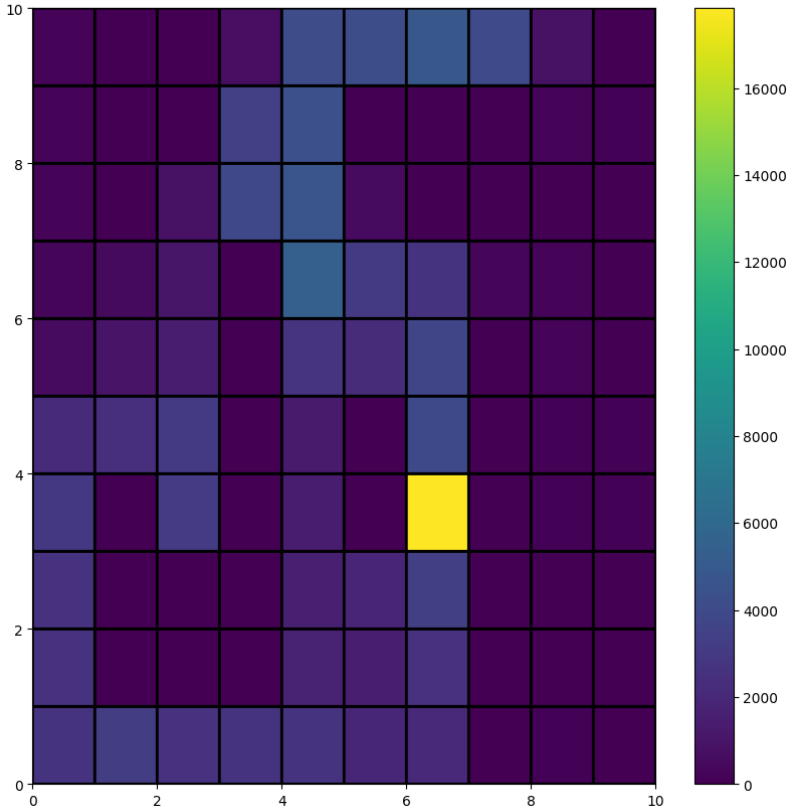




Episode 5000: Reward: -21.030303, Steps: 26.58, Qmax: 9.40, Qmin: -39.70

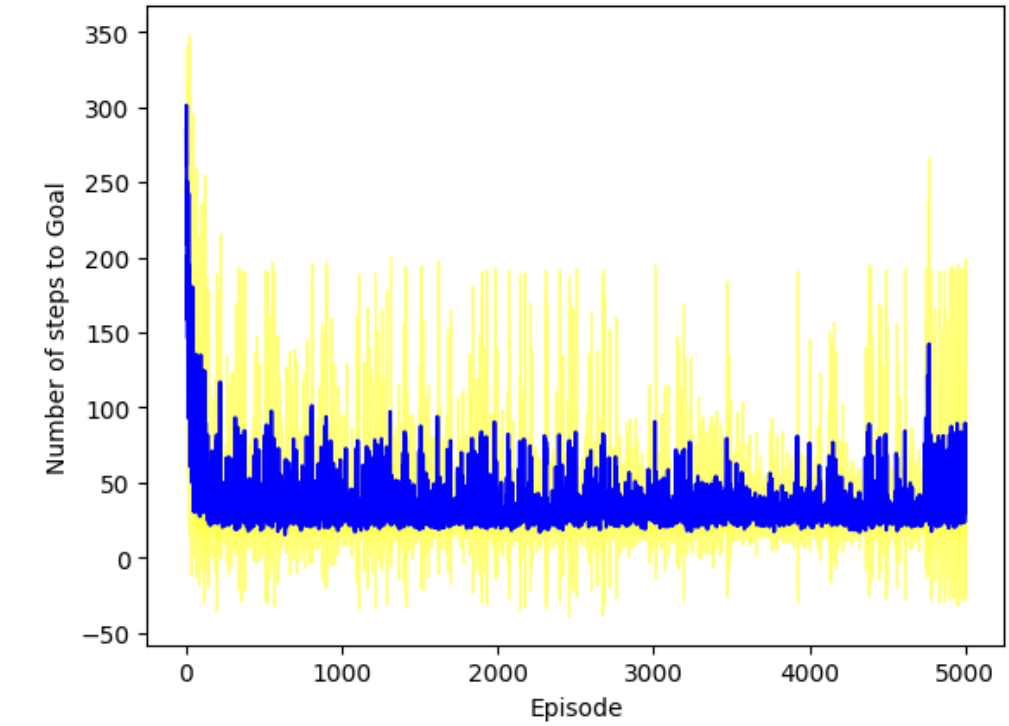


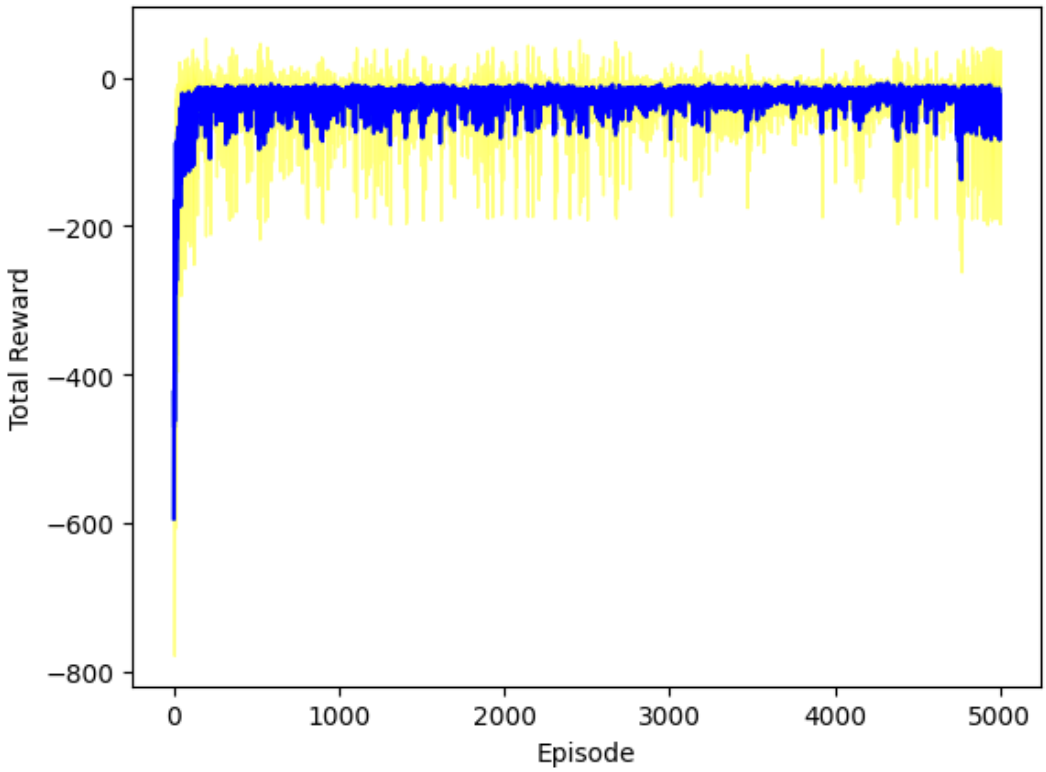
State Visit Count Plot



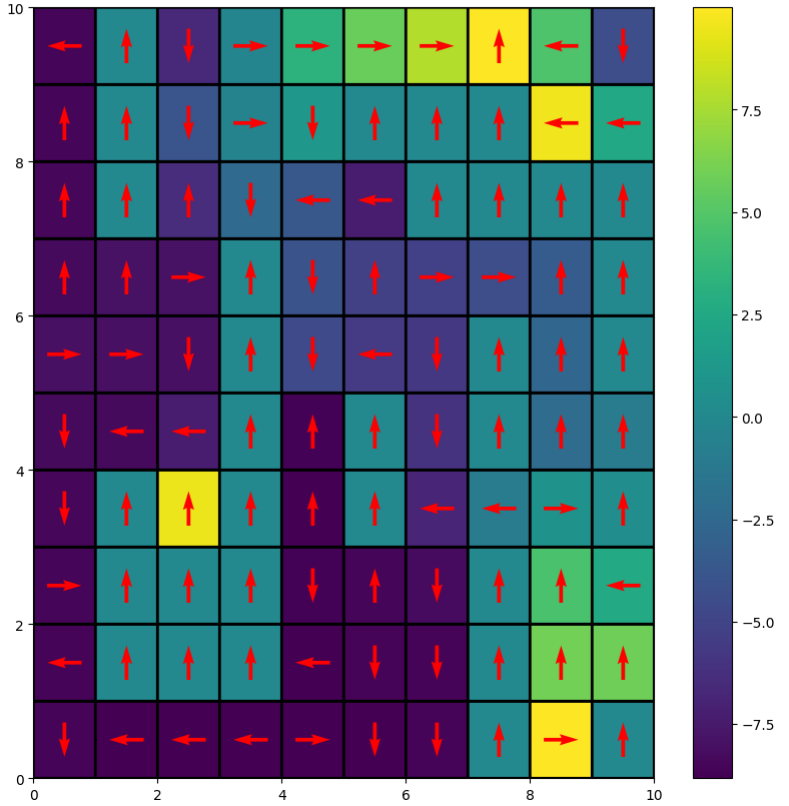
Plots for SARSA

Wind=False **P**=0.7 **Softmax** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.214

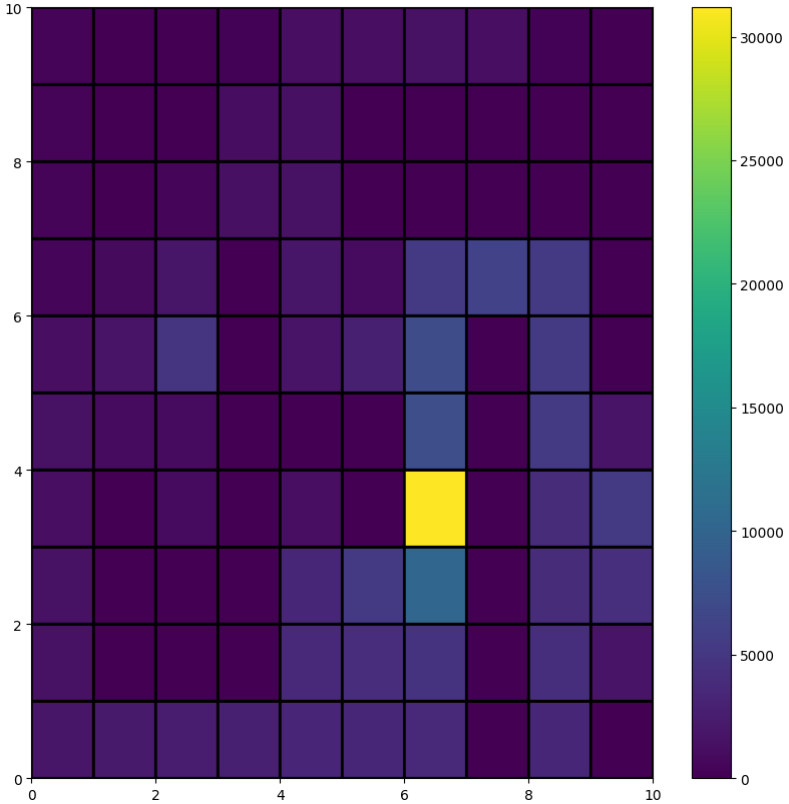




Episode 5000: Reward: -25.171717, Steps: 34.25, Qmax: 10.00, Qmin: -52.36

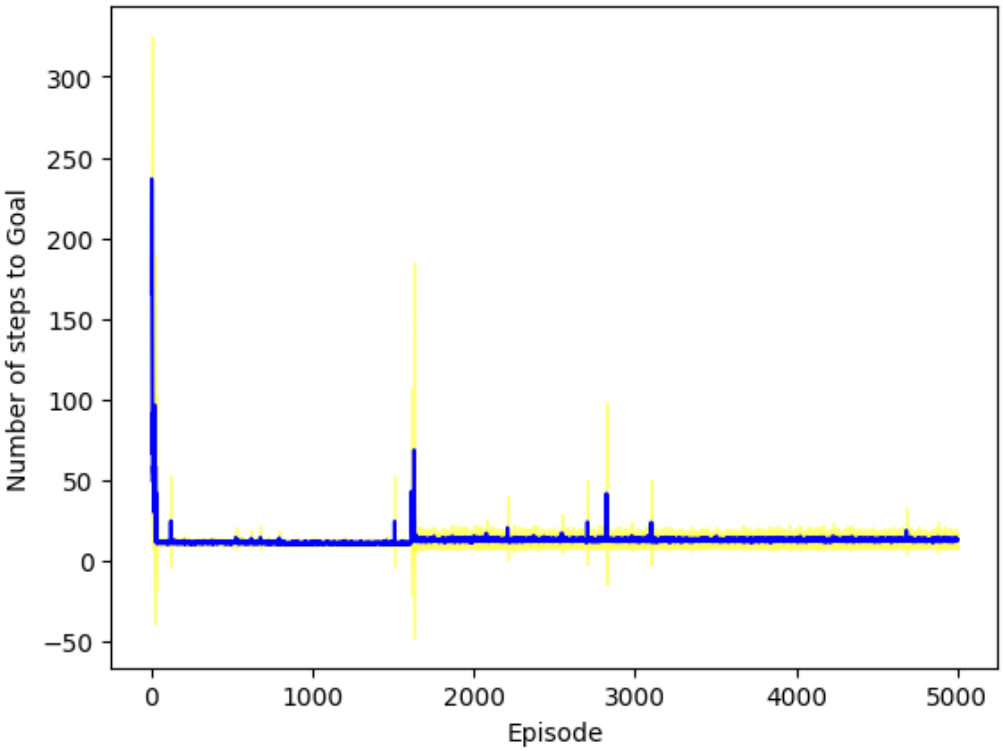


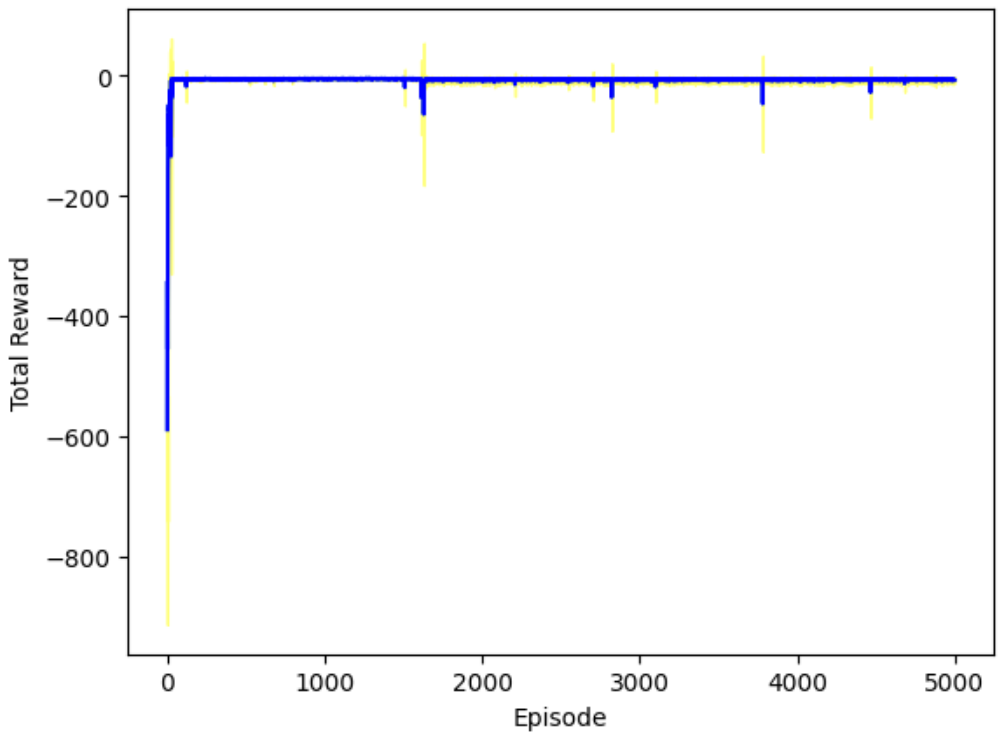
State Visit Count Plot



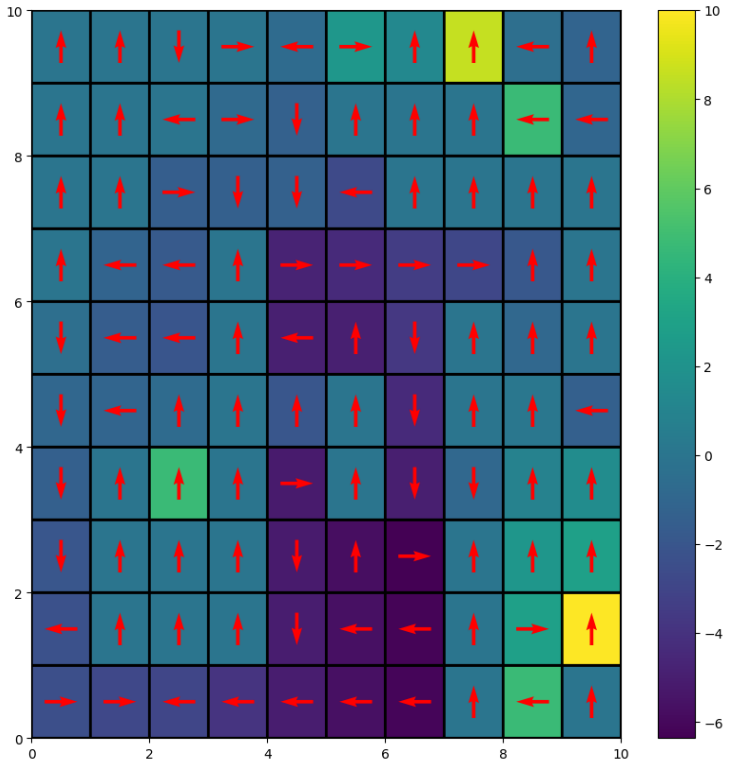
Plots for SARSA

Wind=True **P**=1 **Epsilon Greedy** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Epsilon**= 0.0001

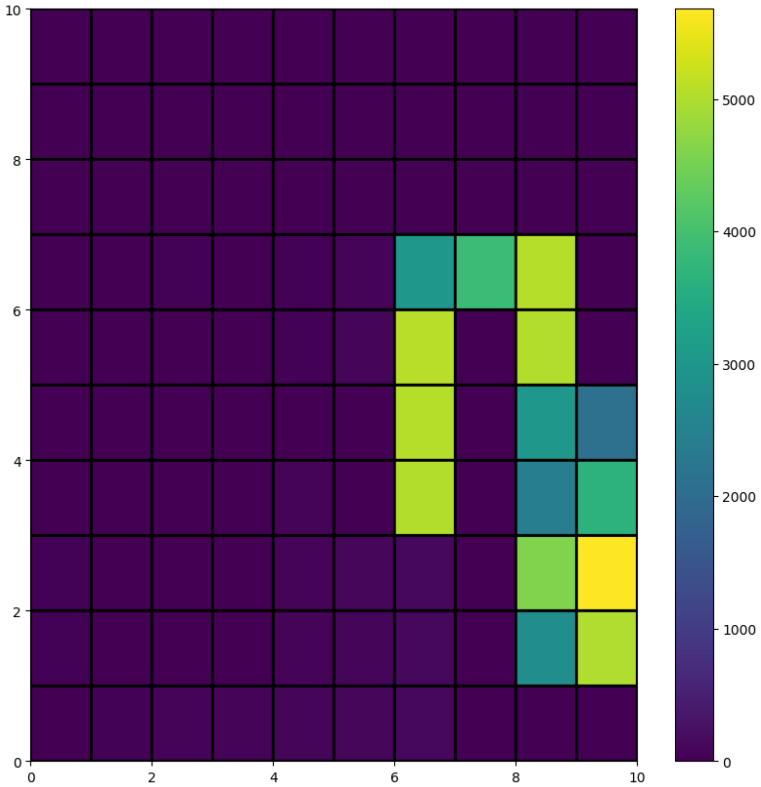




Episode 5000: Reward: -5.898990, Steps: 11.90, Qmax: 10.00, Qmin: -62.46

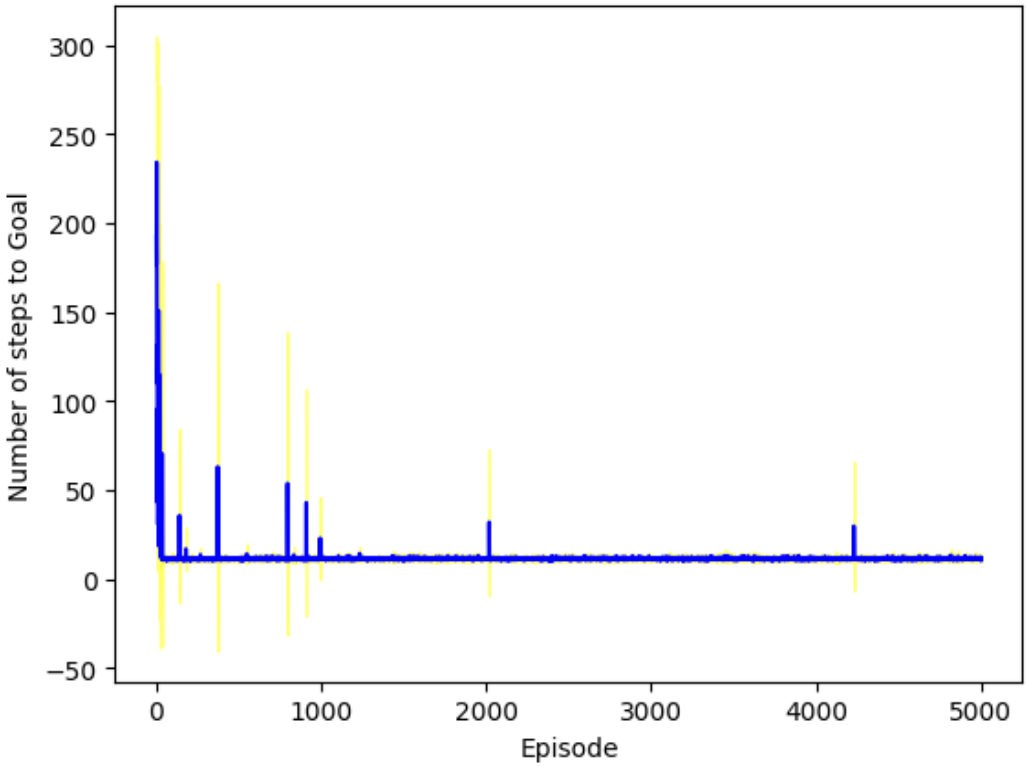


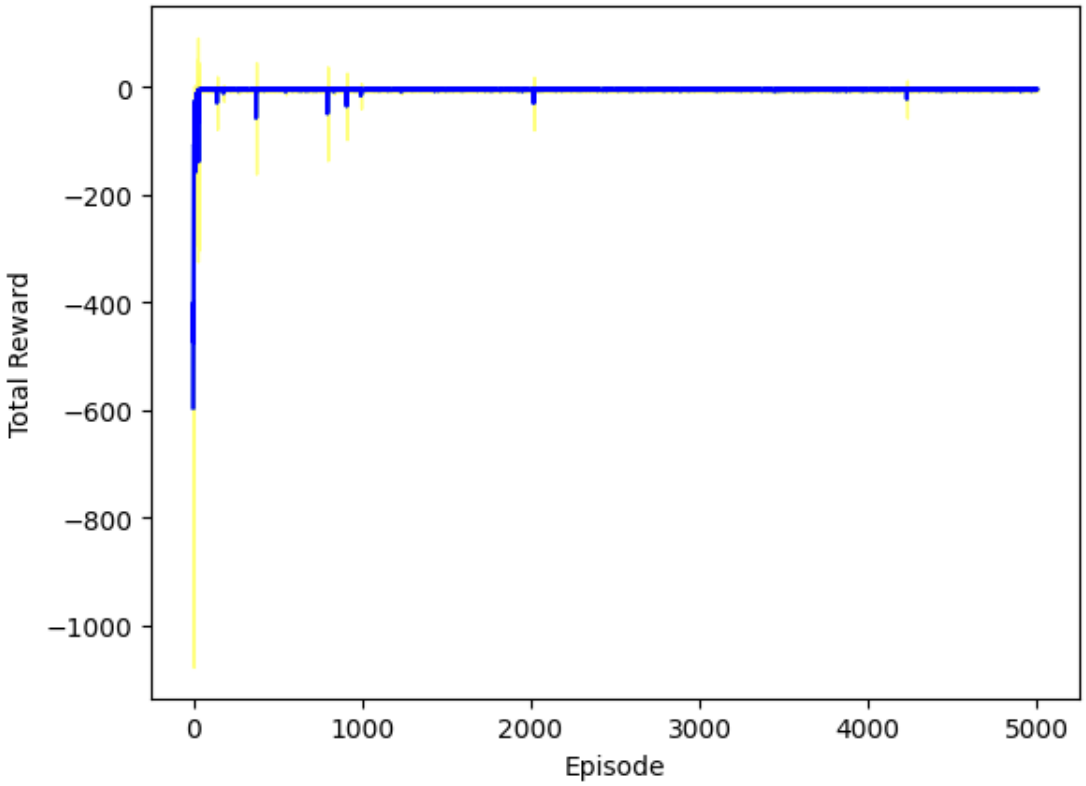
State Visit Count Plot



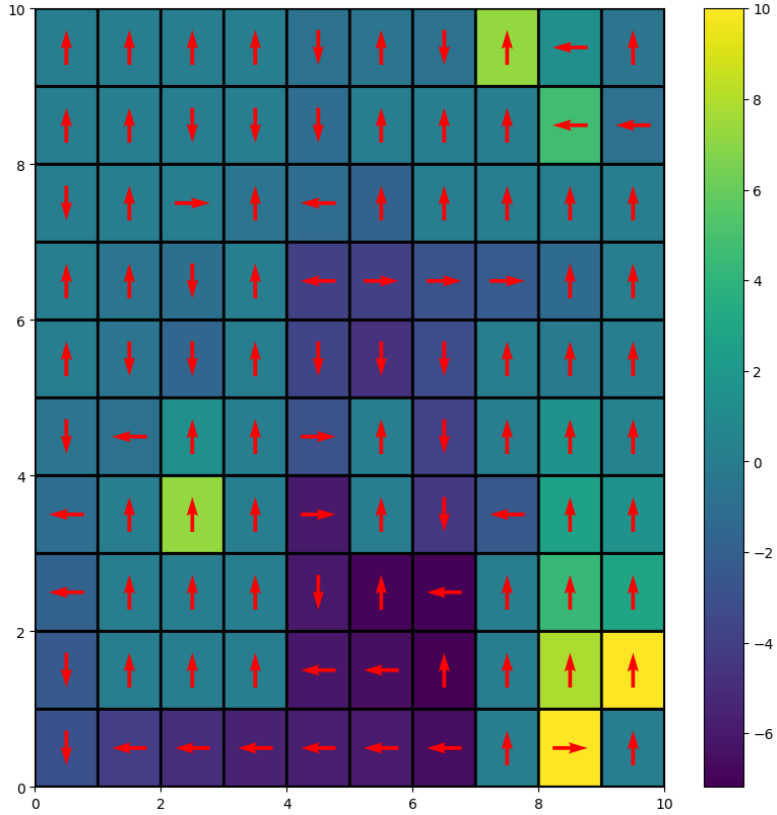
Plots for SARSA

Wind=True **P**=1 **Softmax** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.214

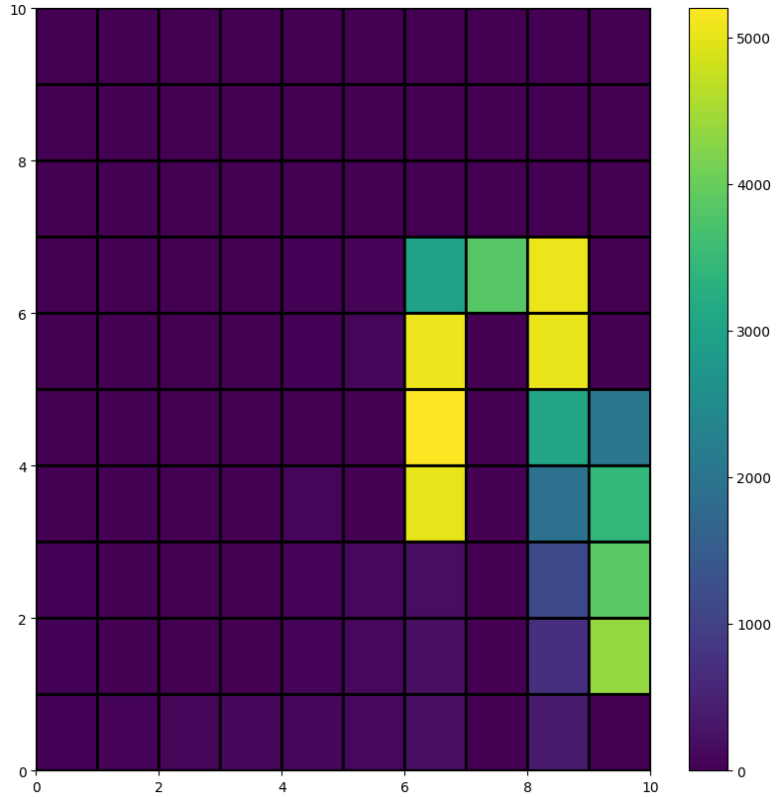




Episode 5000: Reward: -3.858586, Steps: 10.46, Qmax: 10.00, Qmin: -48.61

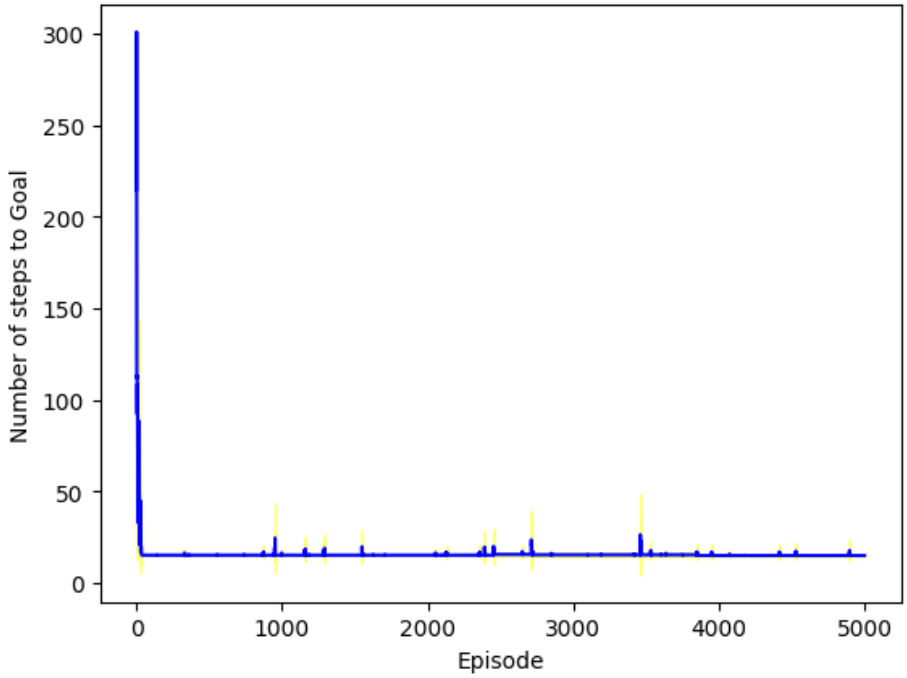


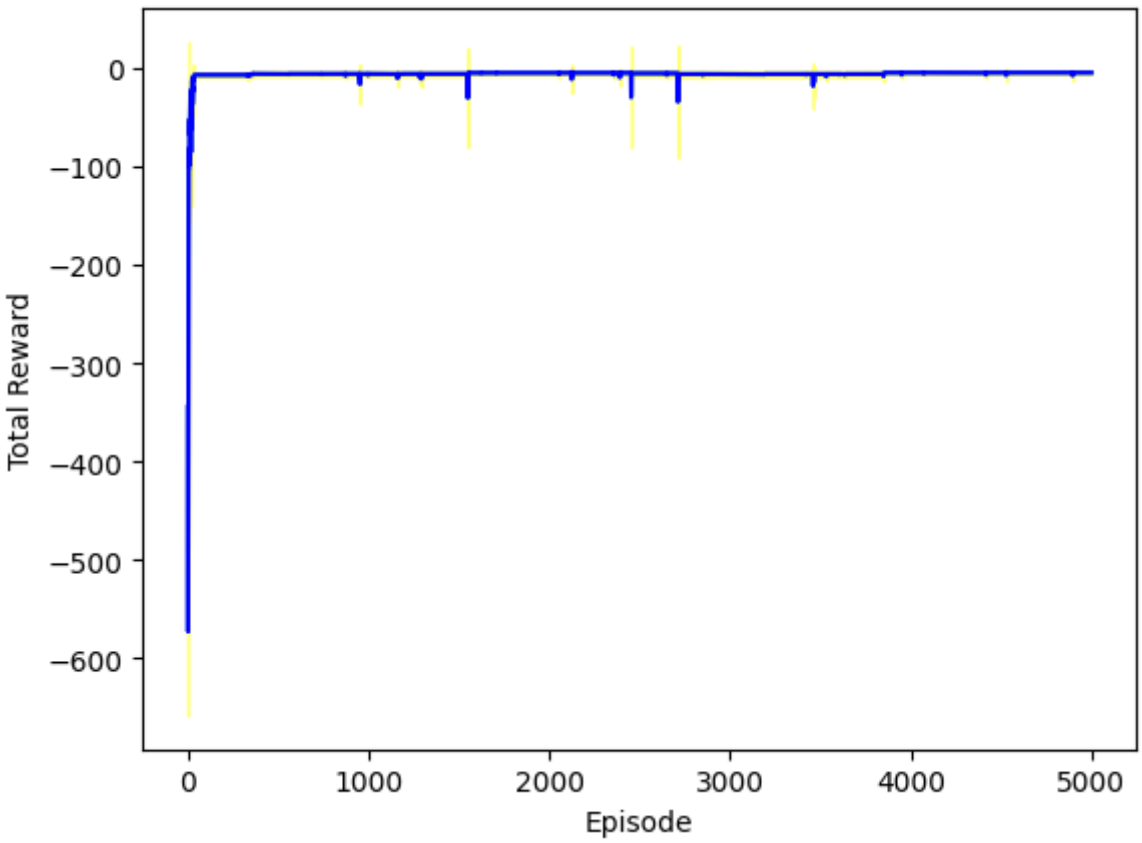
State Visit Count Plot



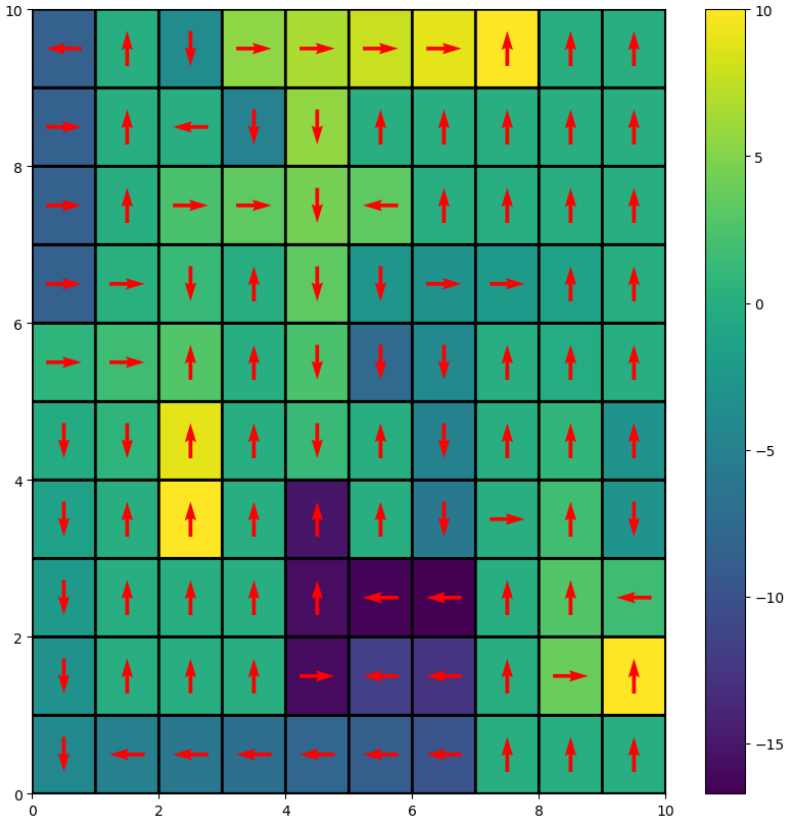
Plots for SARSA

Wind=False **P**=1 **Epsilon Greedy** **Start State**=[0,4]
Alpha= 0.998 **Gamma**=0.985 **Epsilon**= 0.0001

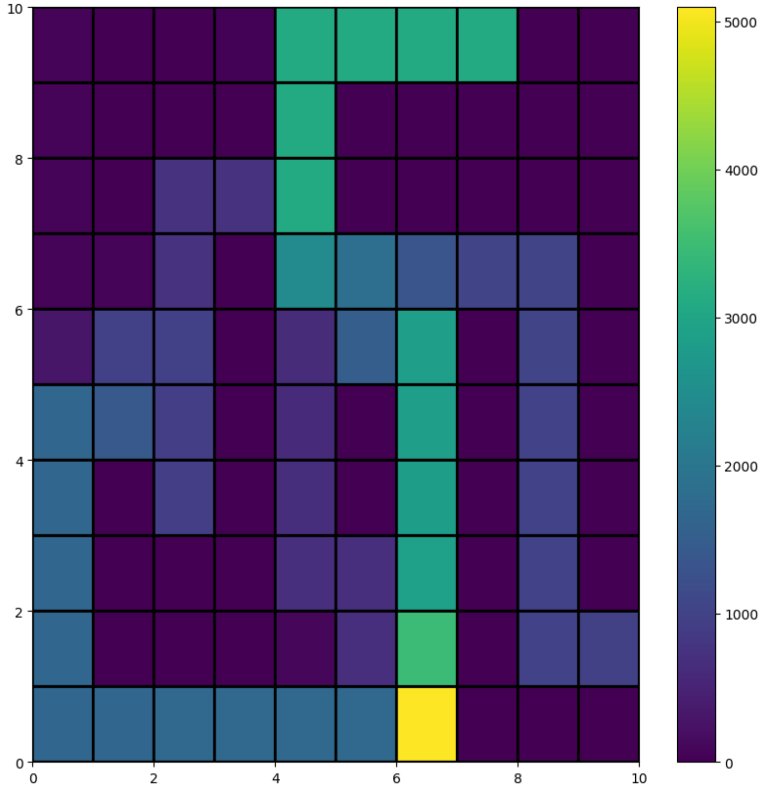




Episode 5000: Reward: -10.000000, Steps: 16.00, Qmax: 10.00, Qmin: -100.00

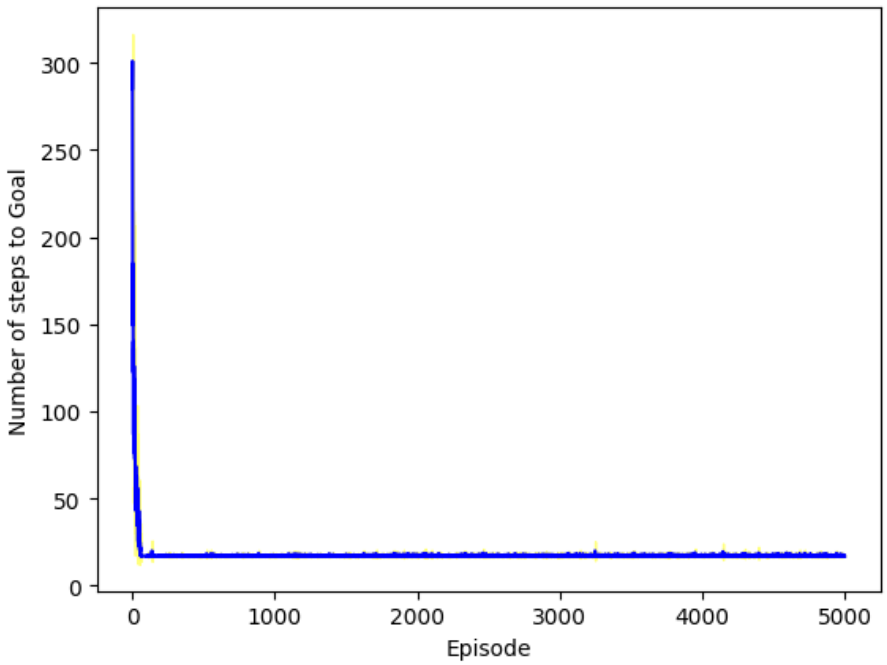


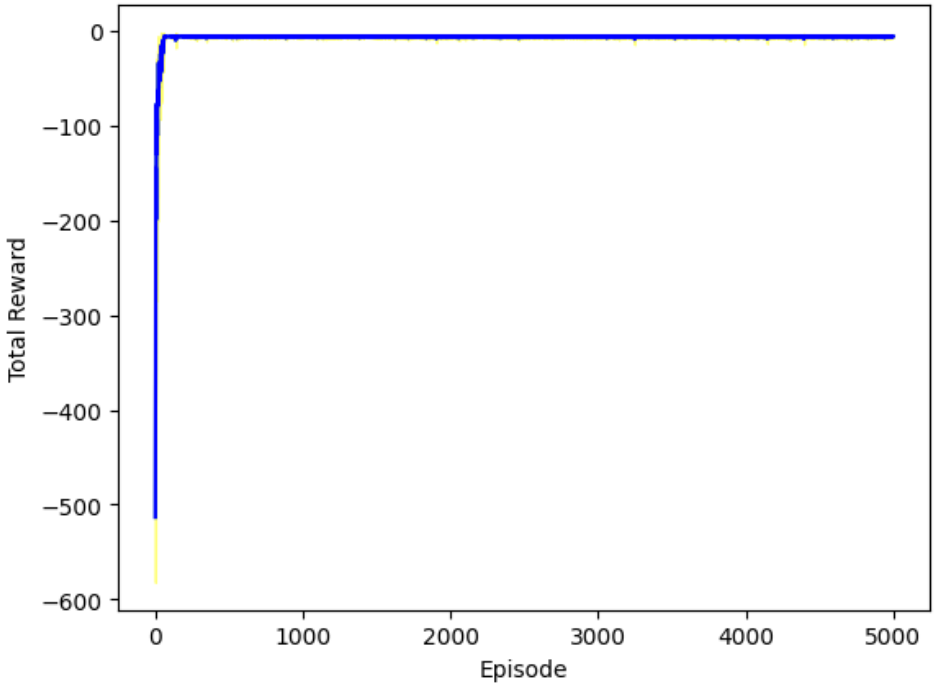
State Visit Count Plot



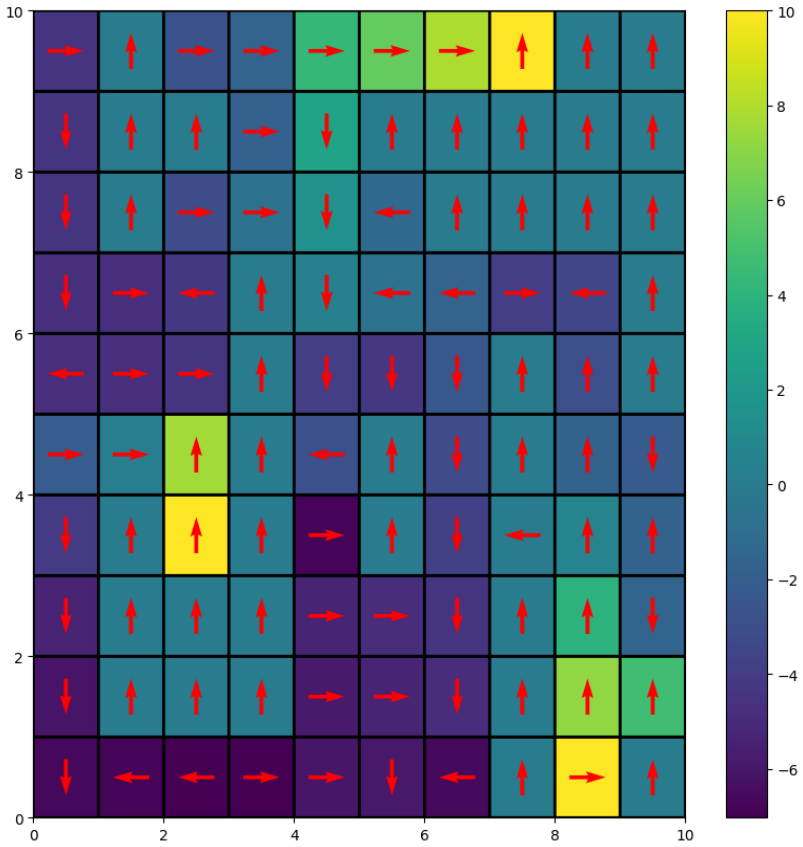
Plots for SARSA

Wind=False **P**=1 **Softmax** **Start State**=[0,4]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.214

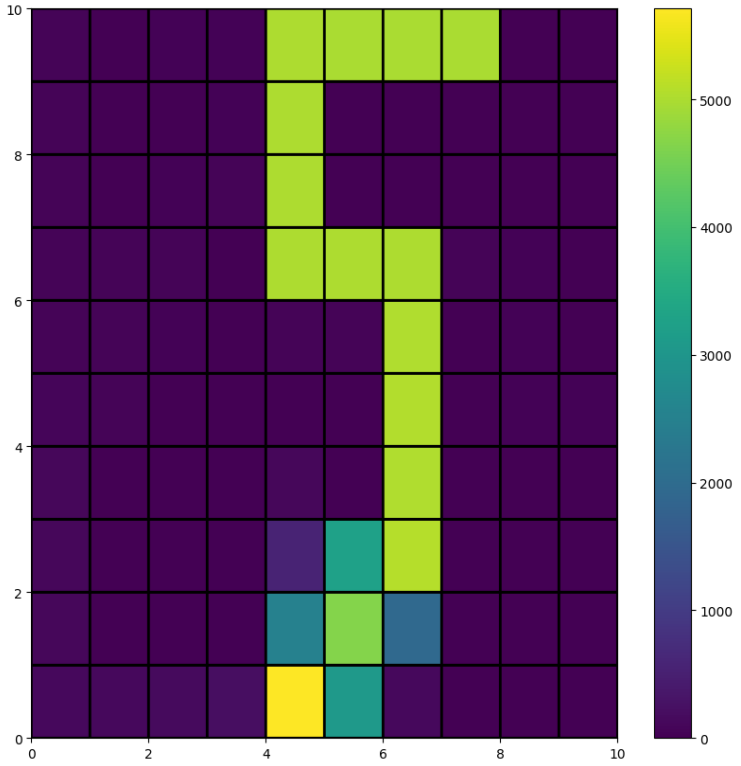




Episode 5000: Reward: -6.292929, Steps: 17.29, Qmax: 10.00, Qmin: -47.49

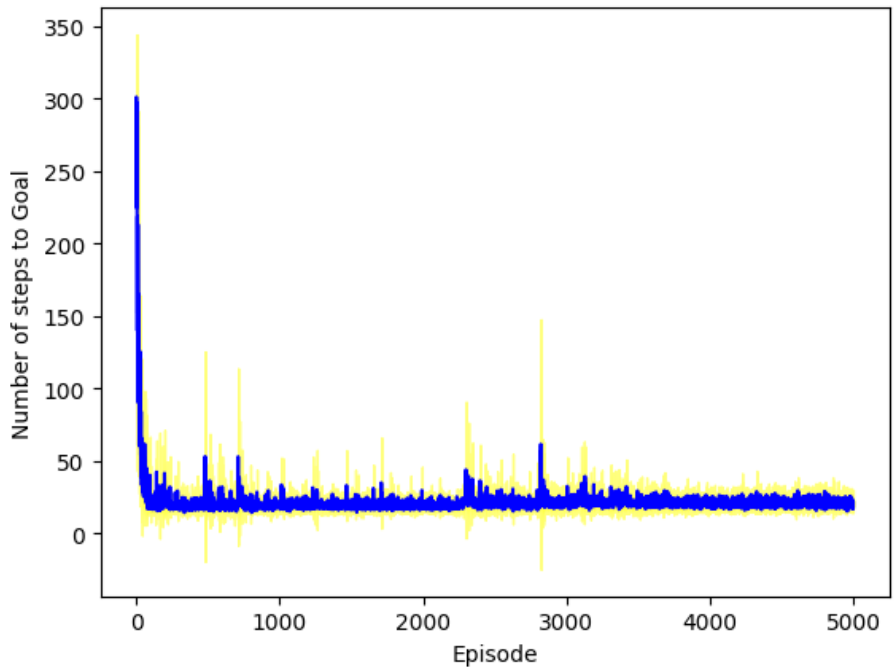


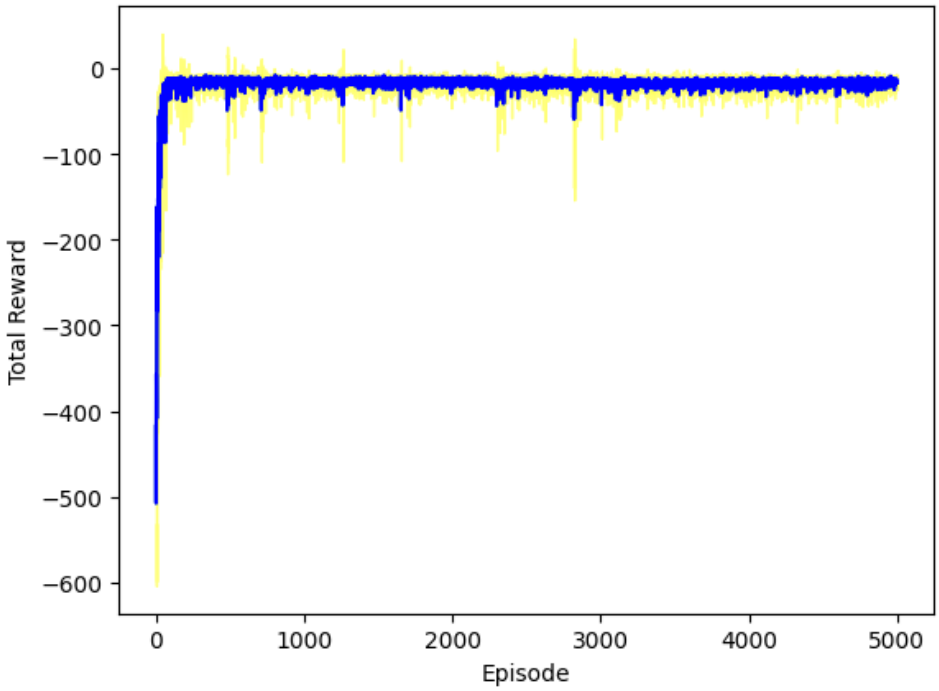
State Visit Count Plot



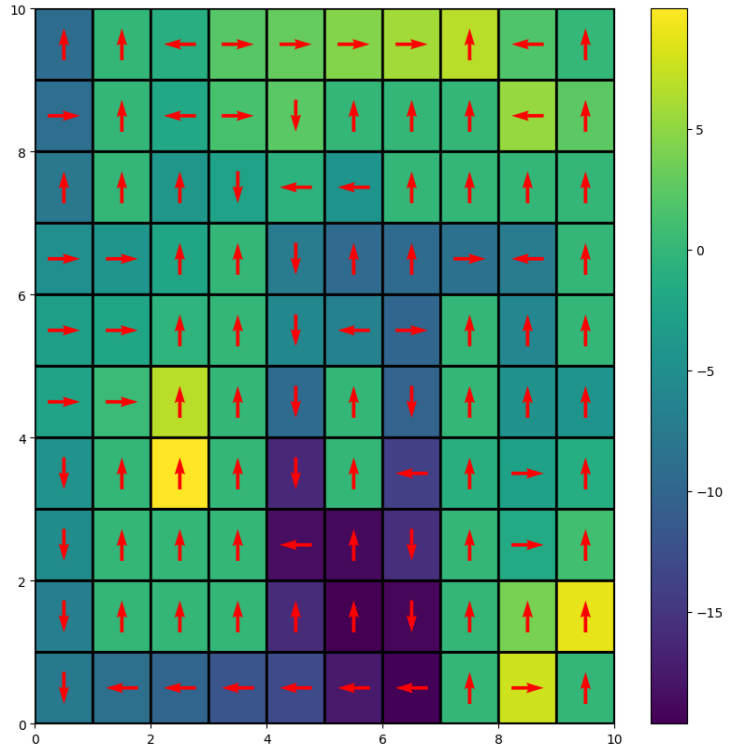
Plots for SARSA

Wind=False **P**=0.7 **Epsilon Greedy** **Start State**=[0,4]
Alpha= 0.361 **Gamma**=0.999 **Epsilon**= 0.0001

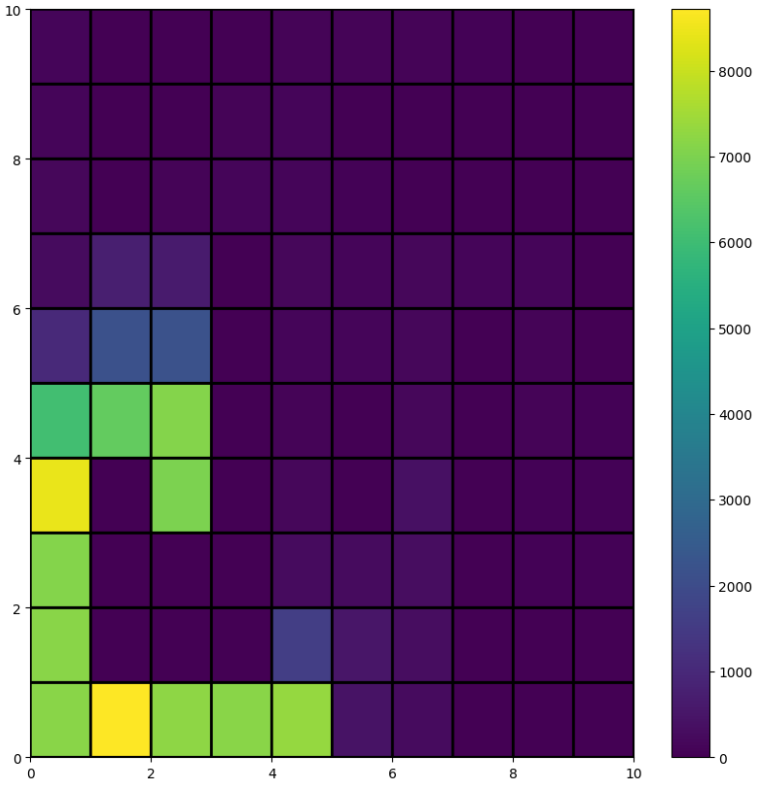




Episode 5000: Reward: -14.808081, Steps: 18.59, Qmax: 9.99, Qmin: -39.90

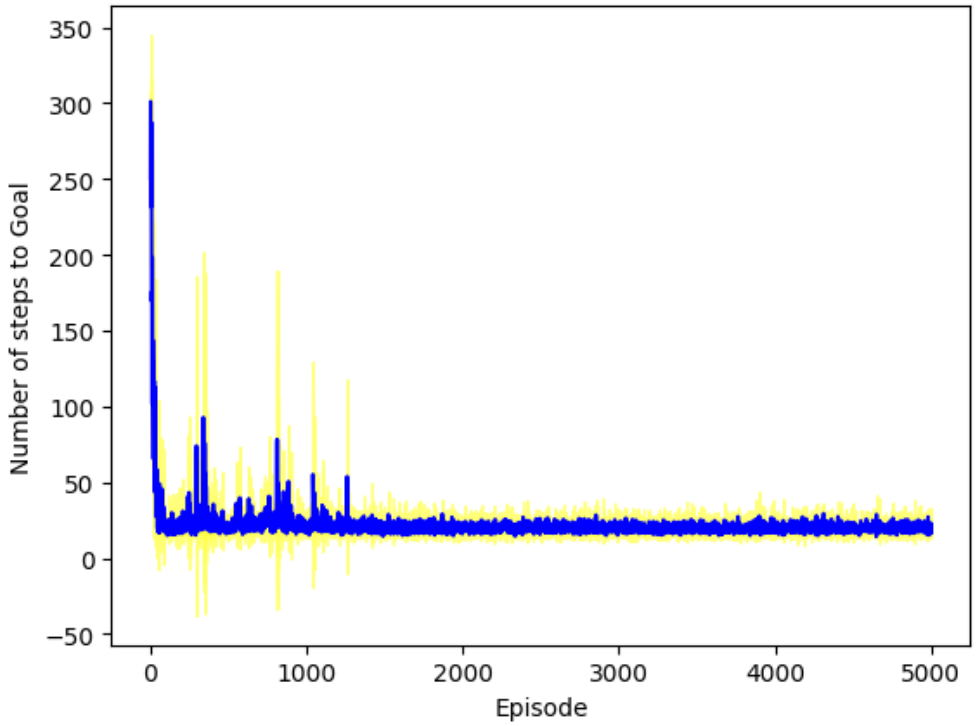


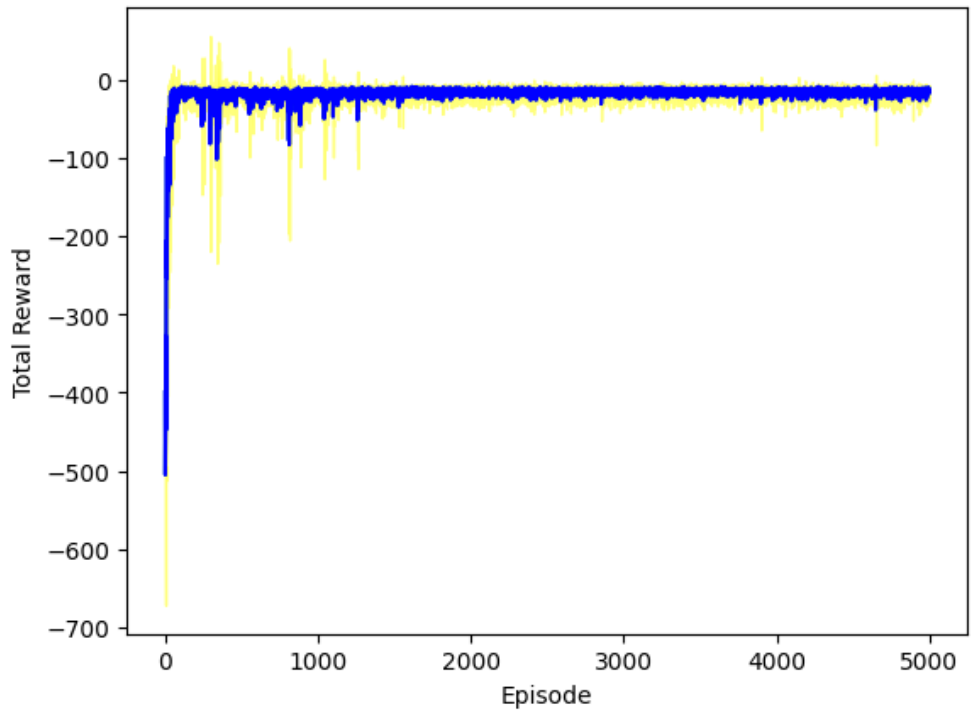
State Visit Count Plot



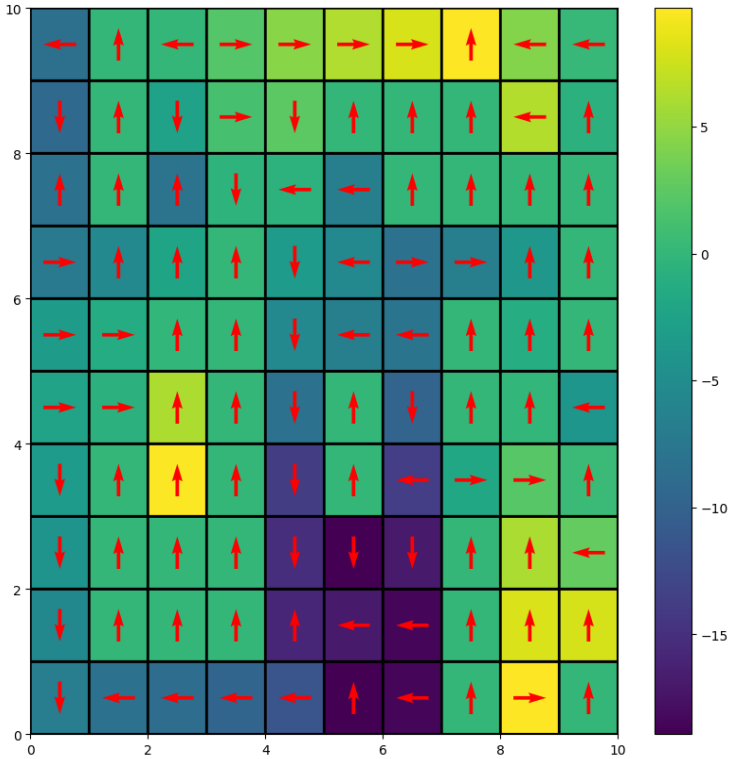
Plots for SARSA

Wind=False **P**=0.7 **Softmax** **Start State**=[0,4]
Alpha= 0.401 **Gamma**=0.980 **Tau**= 0.192

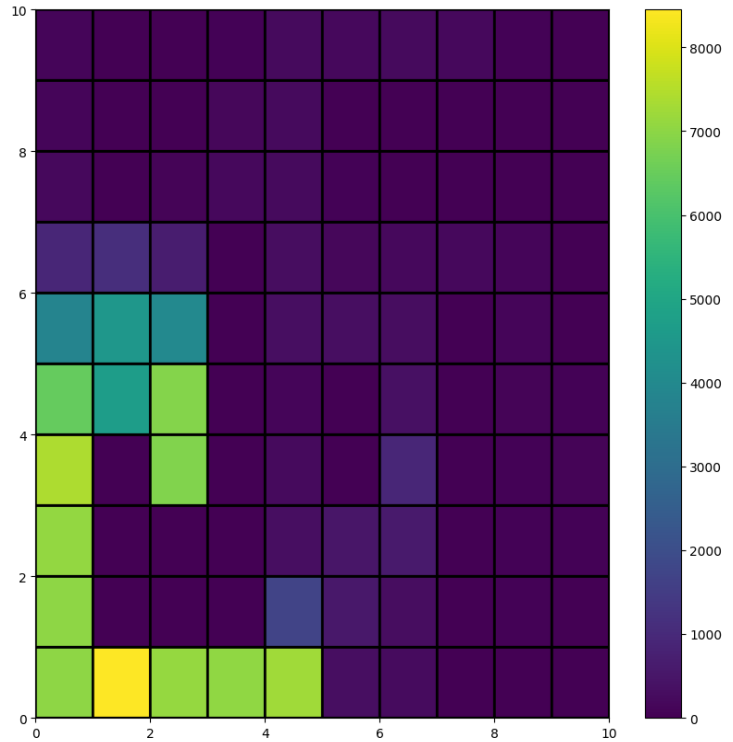




Episode 5000: Reward: -15.585859, Steps: 19.01, Qmax: 9.67, Qmin: -43.40

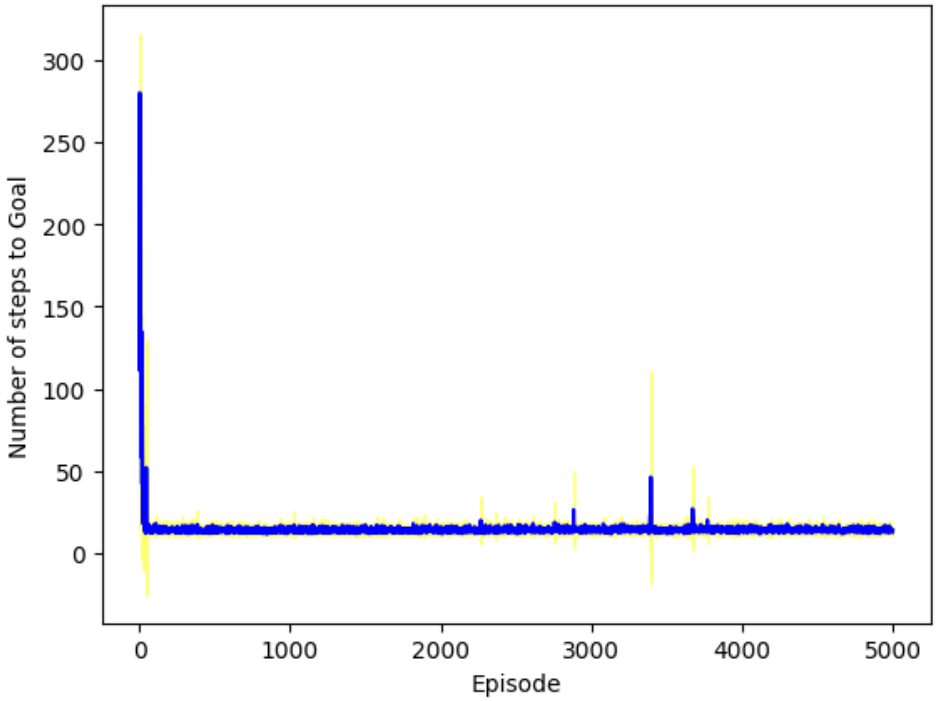


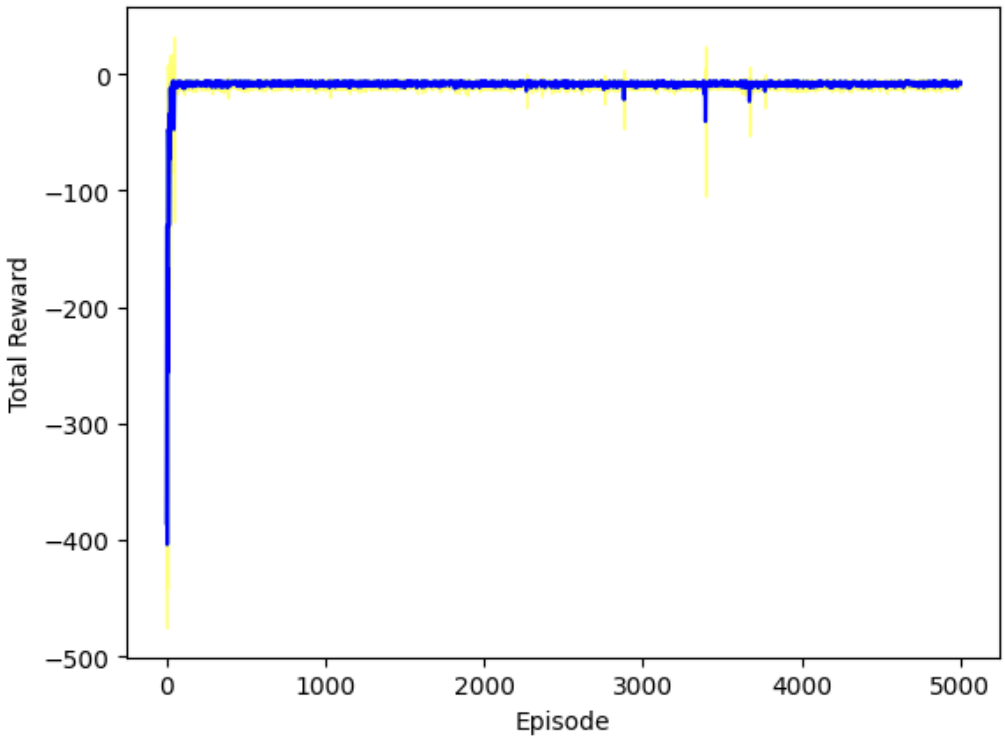
State Visit Count Plot



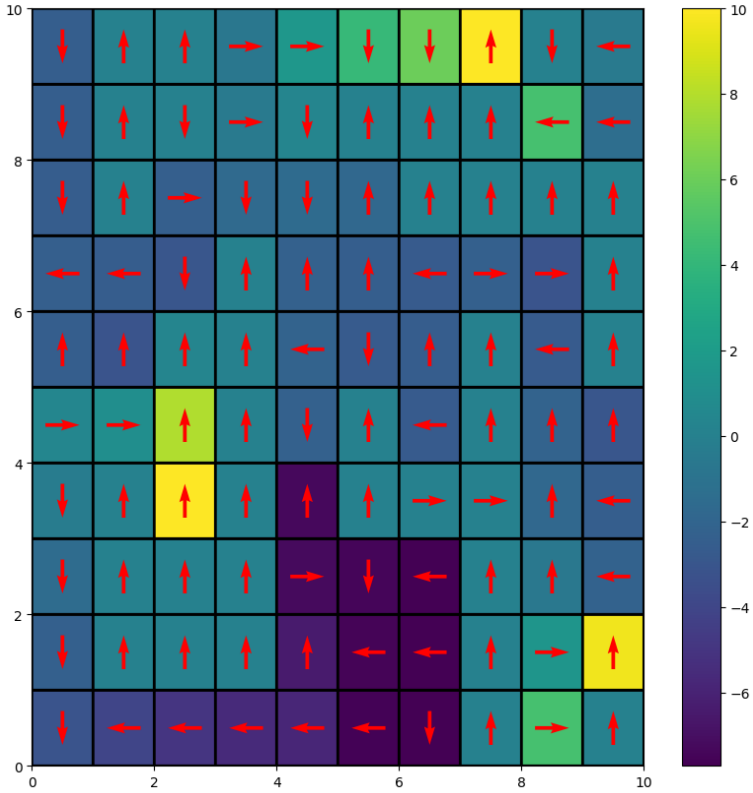
Plots for SARSA

Wind=True **P**=1 **Epsilon greedy** **Start State**=[0,4]
Alpha= 0.474 **Gamma**= 0.887 **Epsilon**= 0.0001

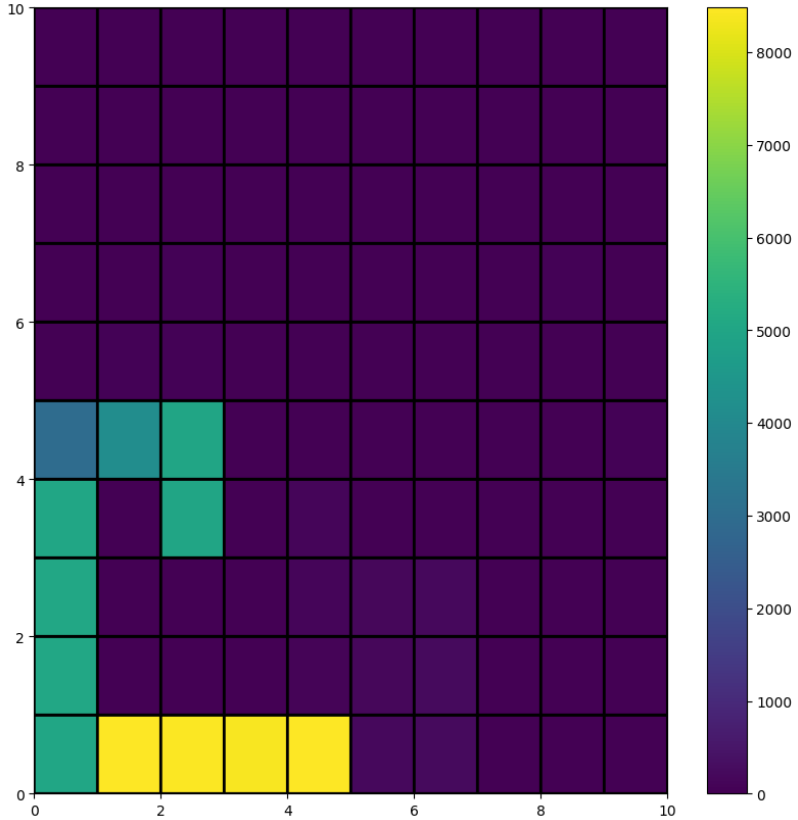




Episode 5000: Reward: -8.272727, Steps: 14.27, Qmax: 10.00, Qmin: -47.87

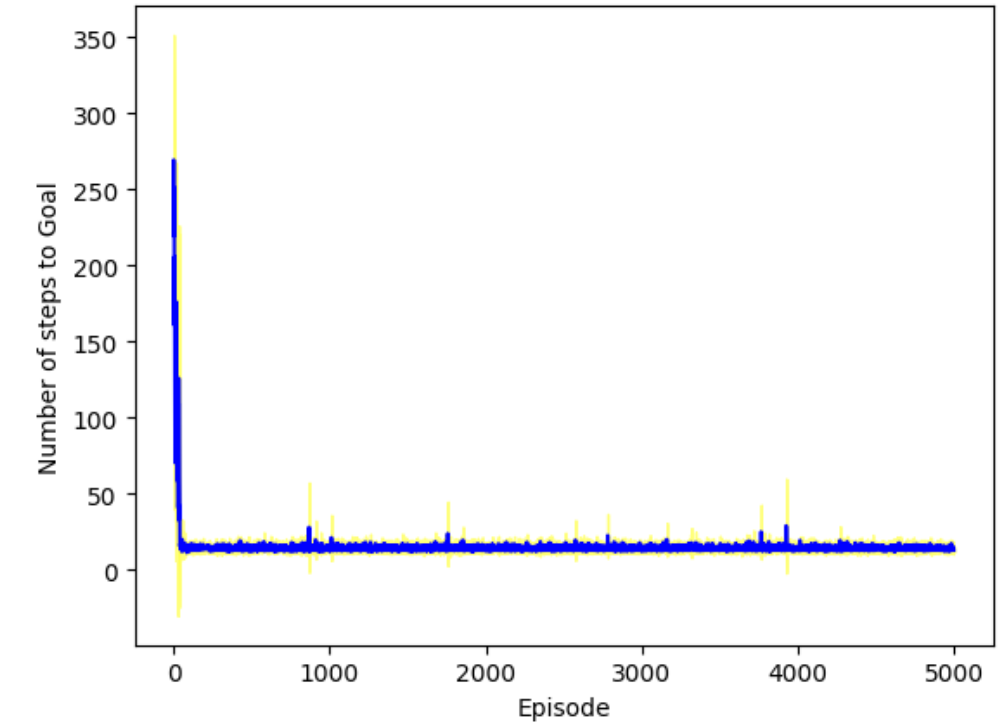


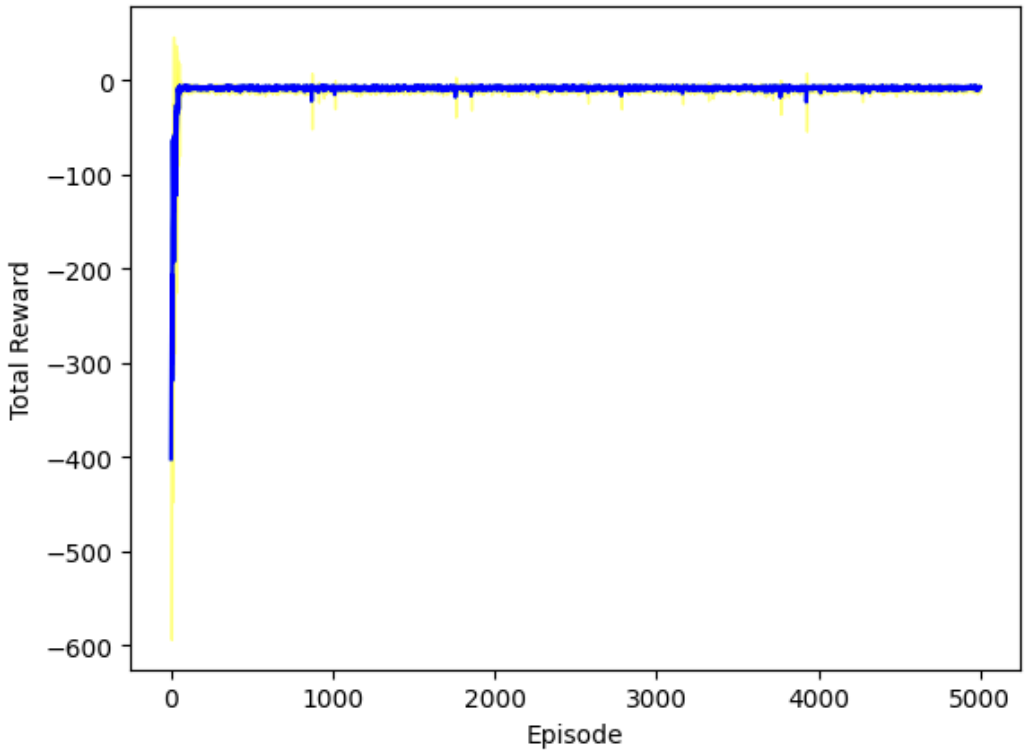
State Visit Count Plot



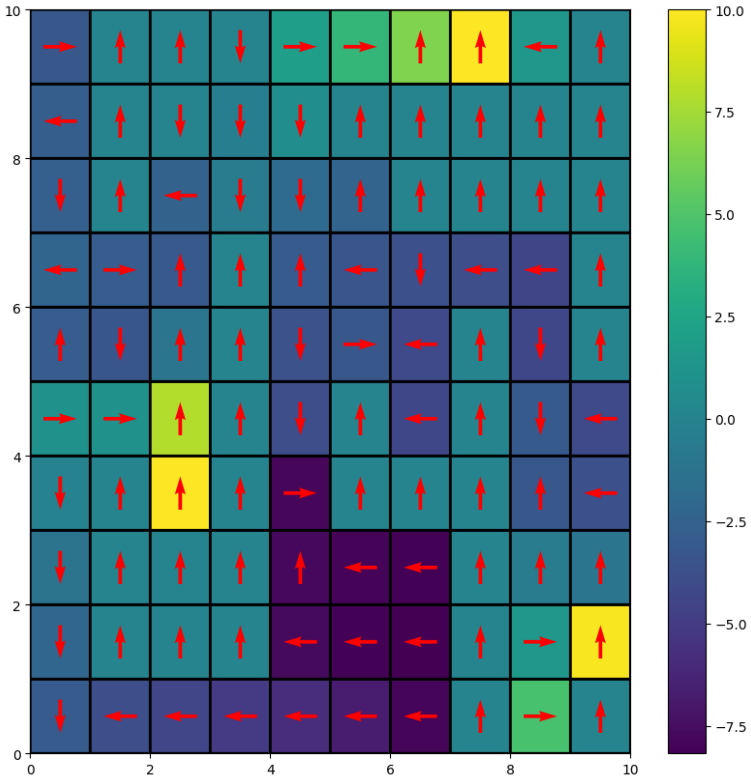
Plots for SARSA

Wind=True **P**=1 **Softmax** **Start State**=[0,4]
Alpha= 0.474 **Gamma**= 0.887 **Tau**= 0.214

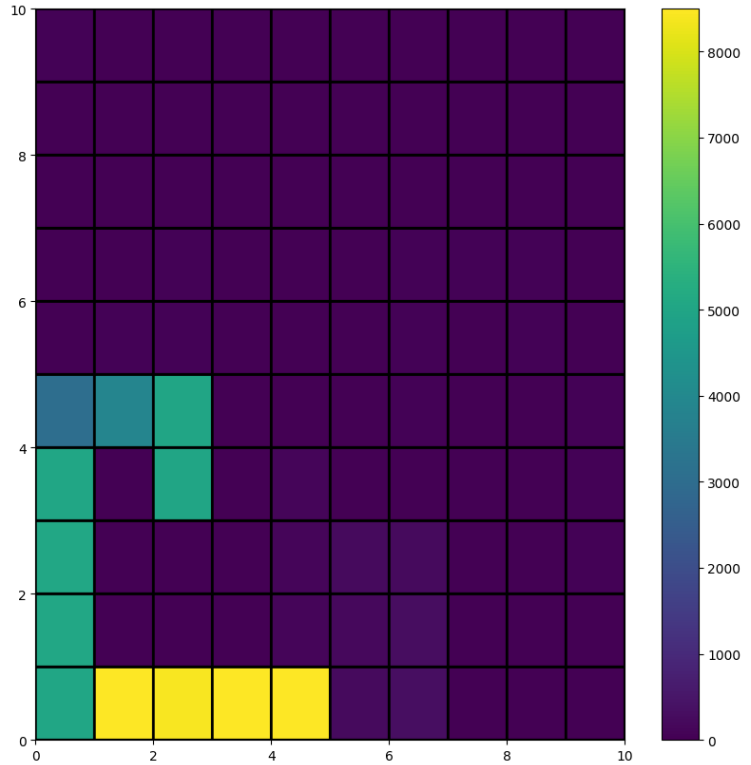




Episode 5000: Reward: -8.080808, Steps: 14.08, Qmax: 10.00, Qmin: -48.58

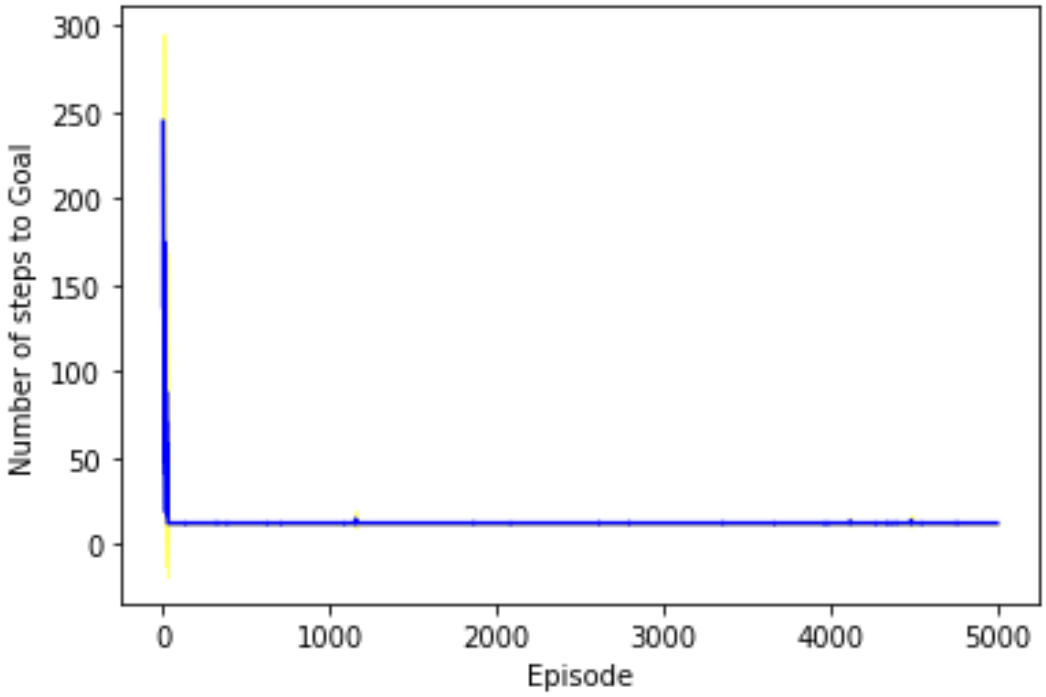


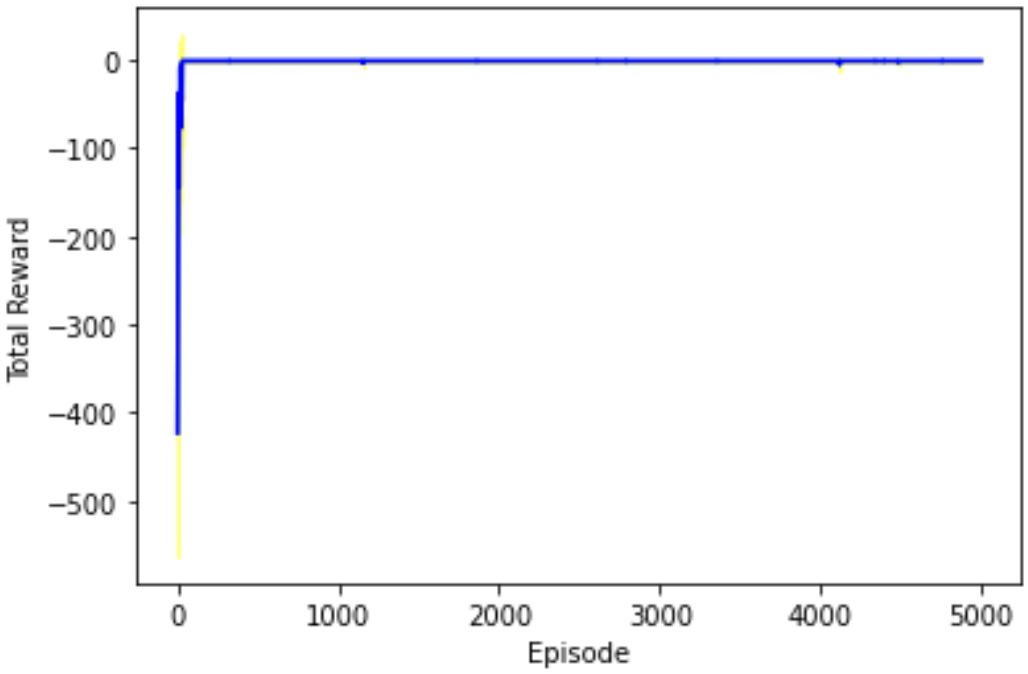
State Visit Count Plot

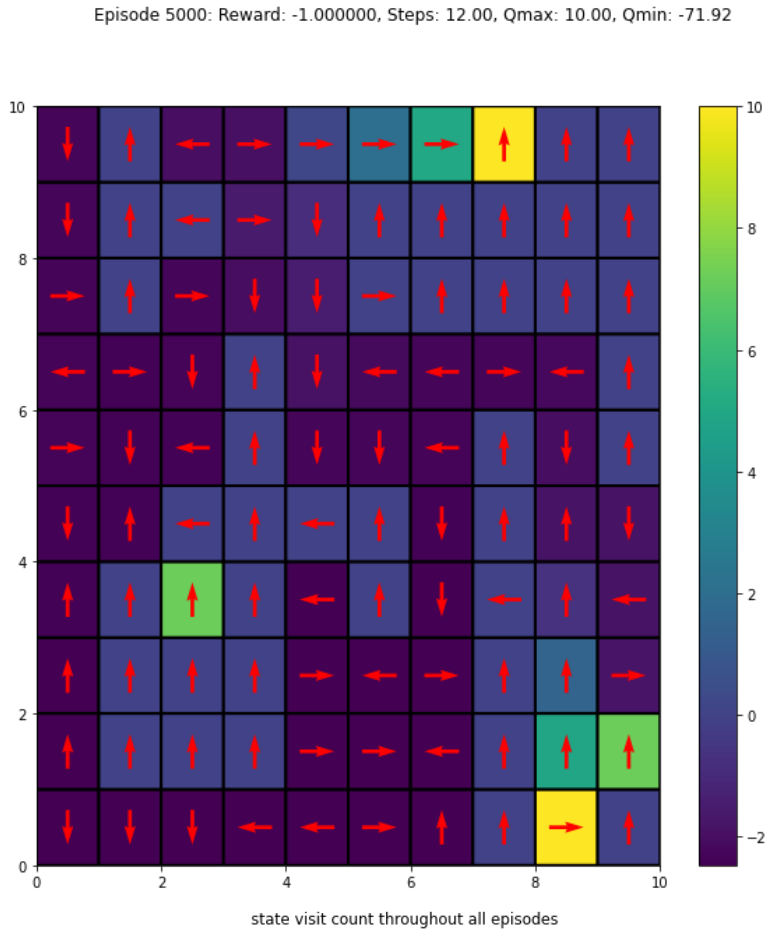


Plots for Q Learning

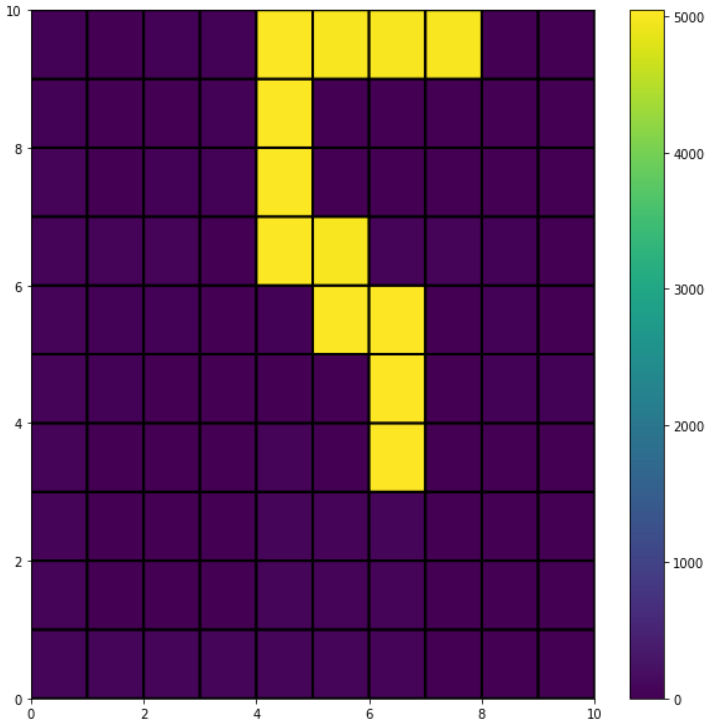
Wind=False **P**=1 **Epsilon Greedy** **Start State**=[3,6]
Alpha= 0.719 **Gamma**= 0.6 **Epsilon**= 0.0001





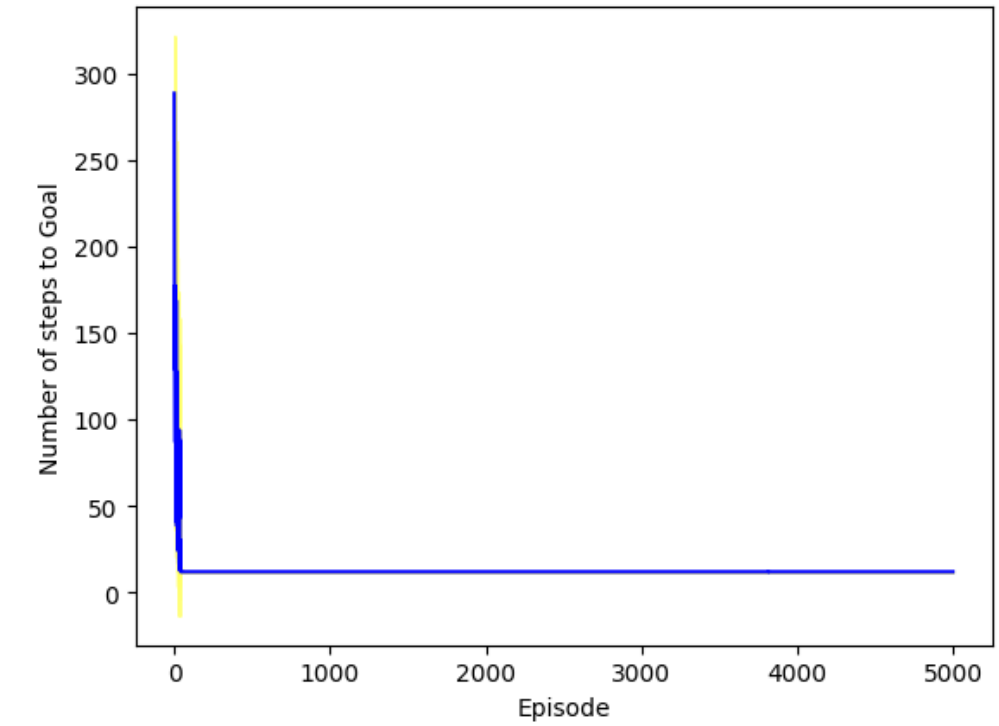


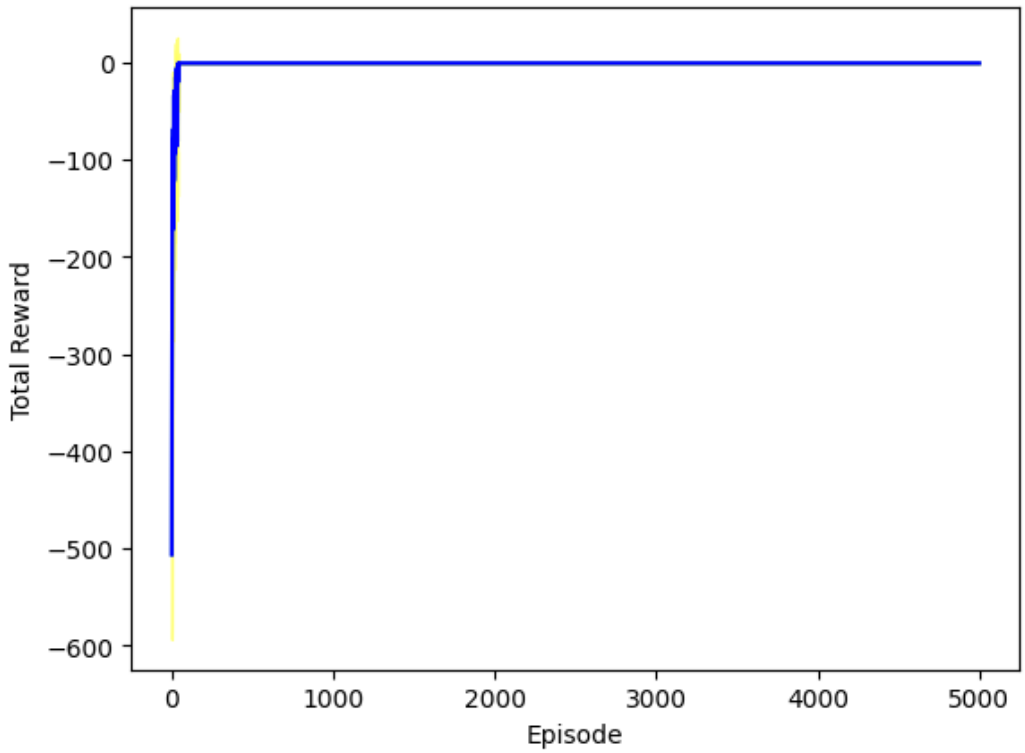
state visit count throughout all episodes



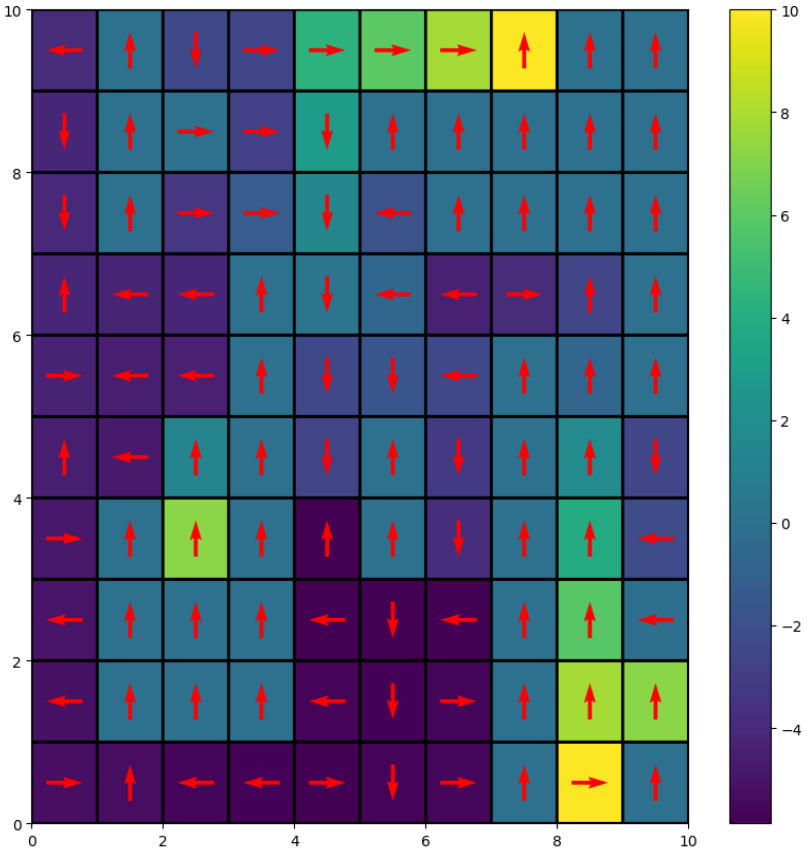
Plots for Q Learning

Wind=False **P**=1 **Softmax** **Start State**=[3,6]
Alpha= 0.474 **Gamma**= 0.887 **Tau**= 0.214

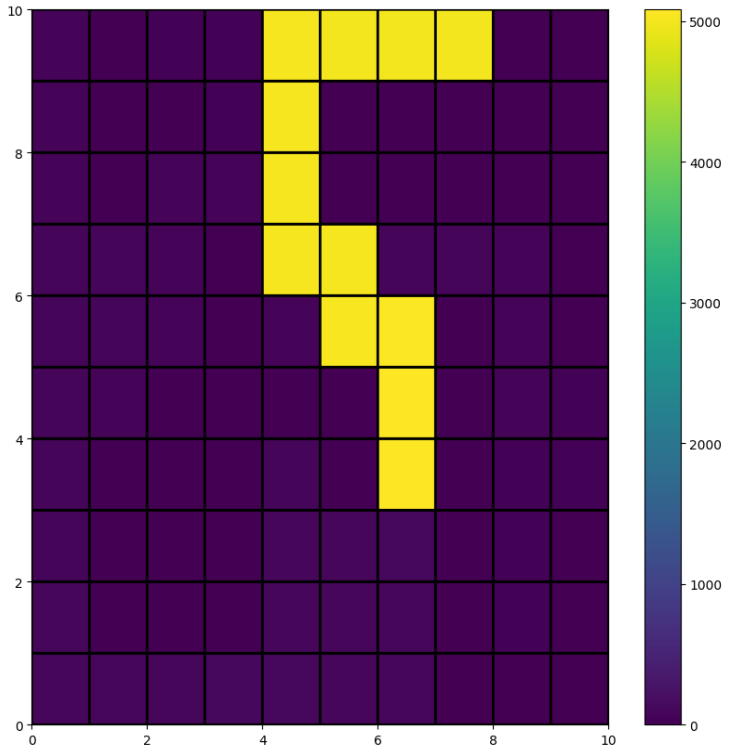




Episode 5000: Reward: -1.000000, Steps: 12.00, Qmax: 10.00, Qmin: -47.49

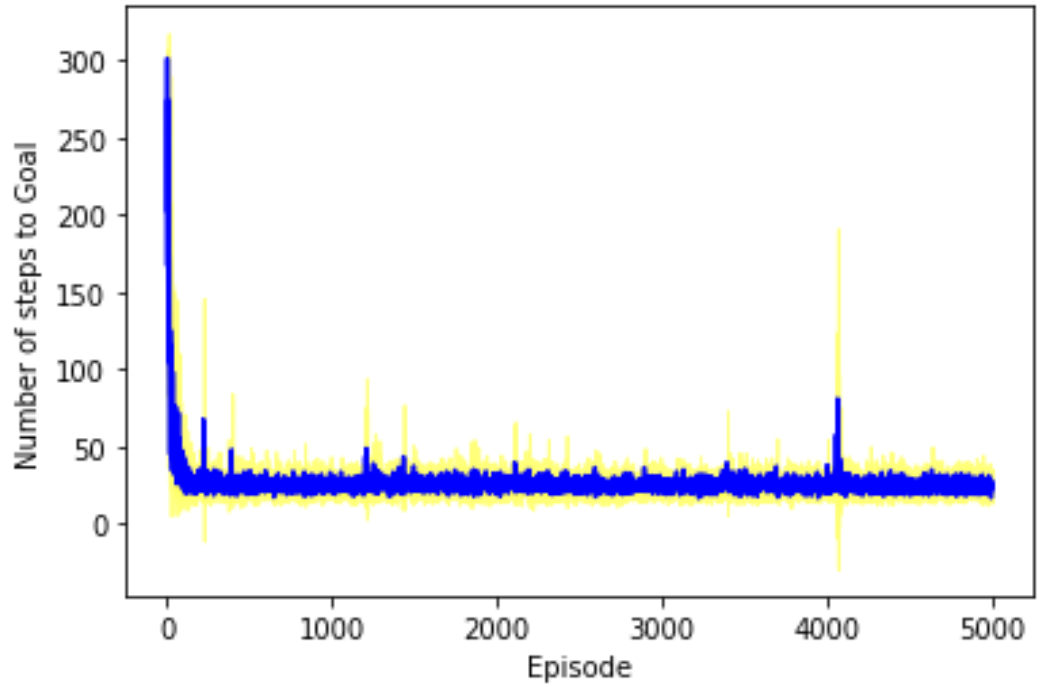


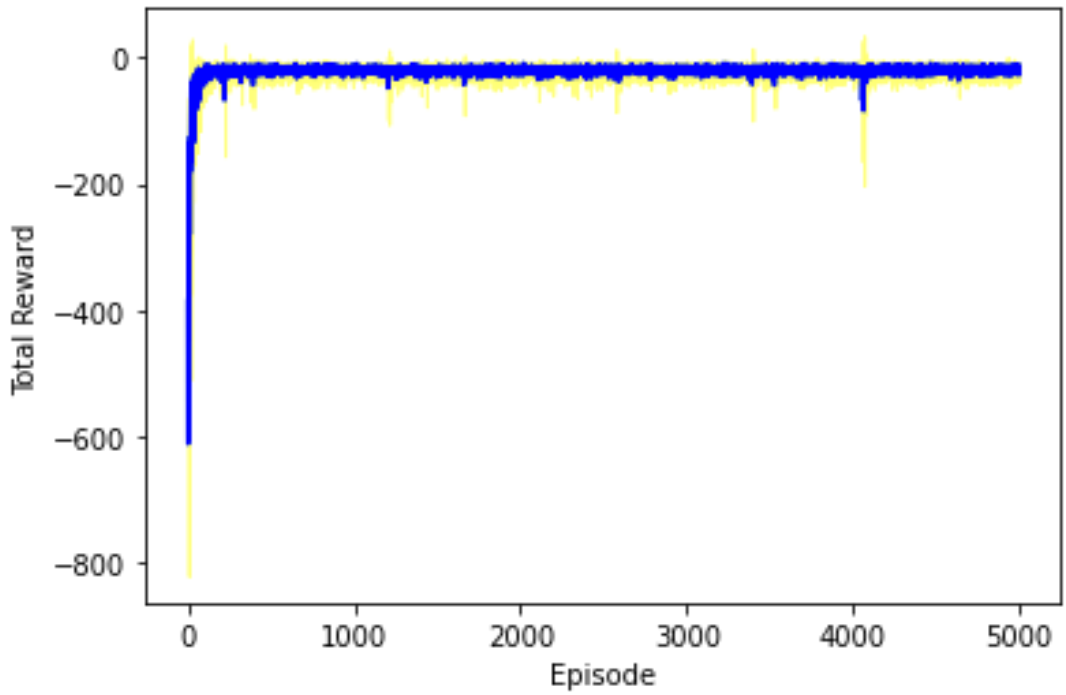
State Visit Count Plot



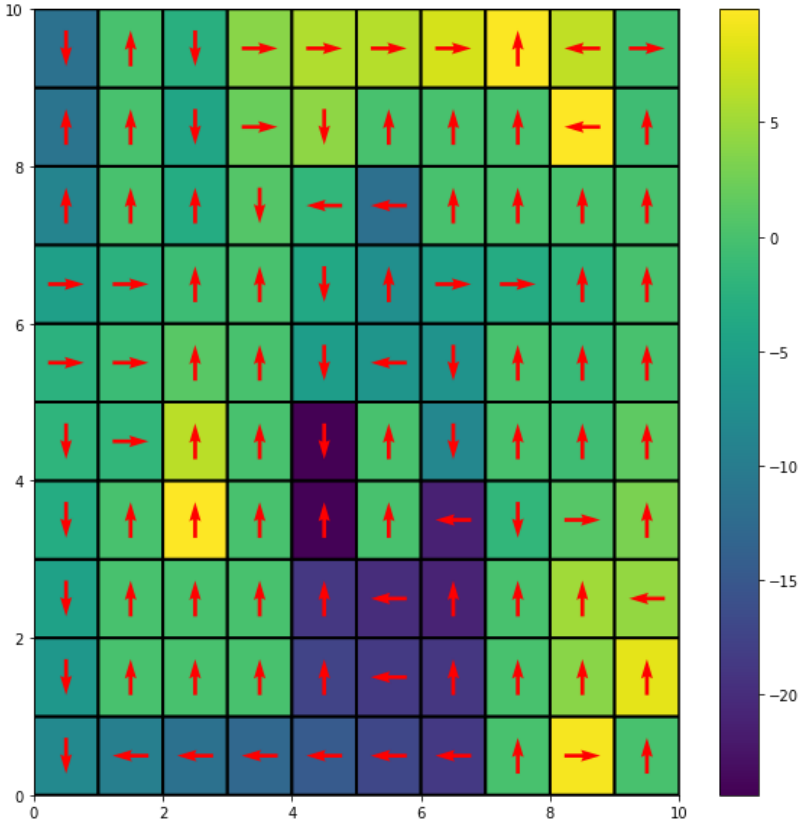
Plots for Q Learning

Wind=False **P**=0.7 **Epsilon Greedy** **Start State**=[3,6]
Alpha= 0.4498 **Gamma**=0.999 **Epsilon**= 0.0001

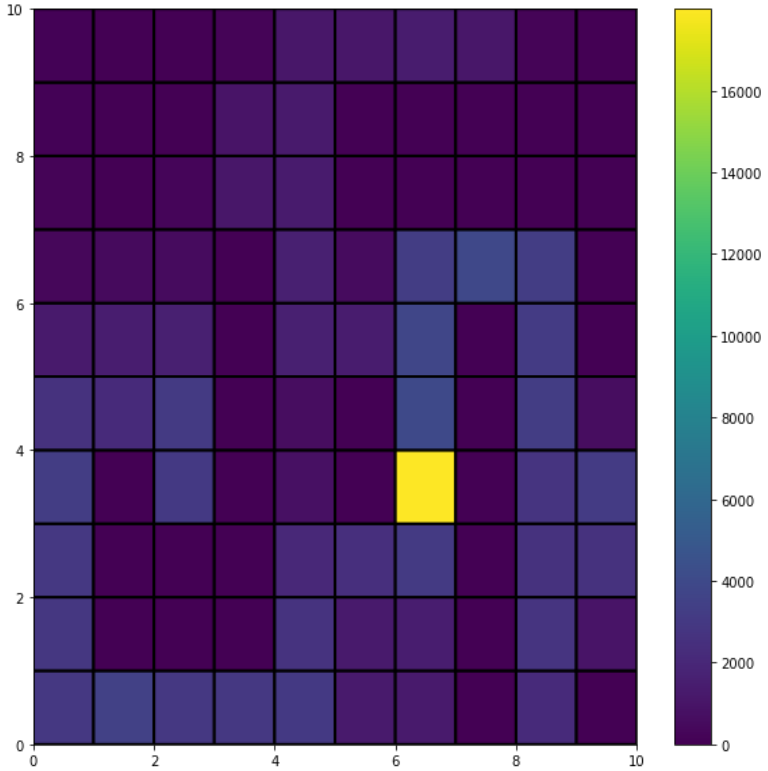




Episode 5000: Reward: -19.626263, Steps: 26.03, Qmax: 9.95, Qmin: -47.77

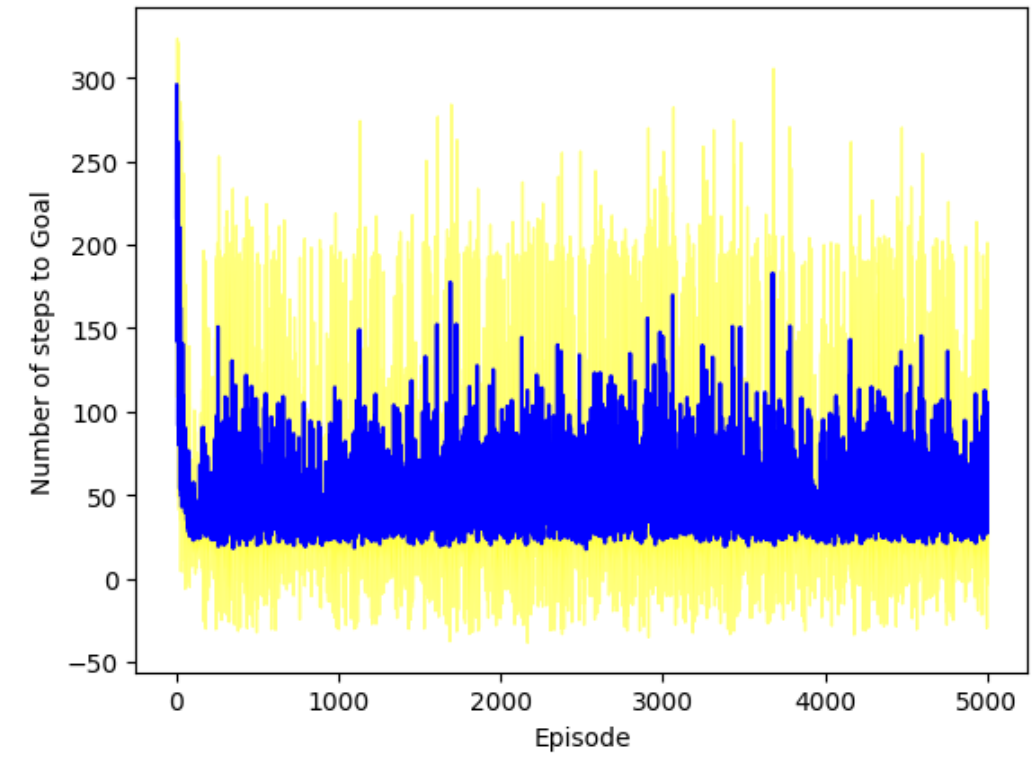


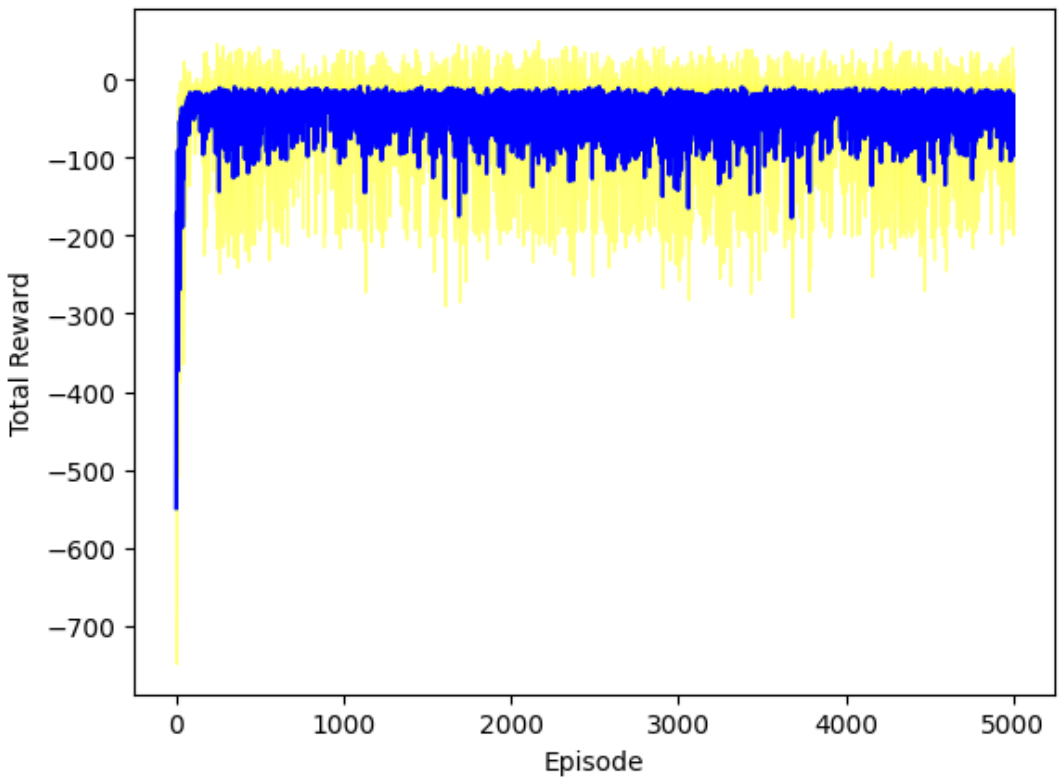
state visit count throughout all episodes



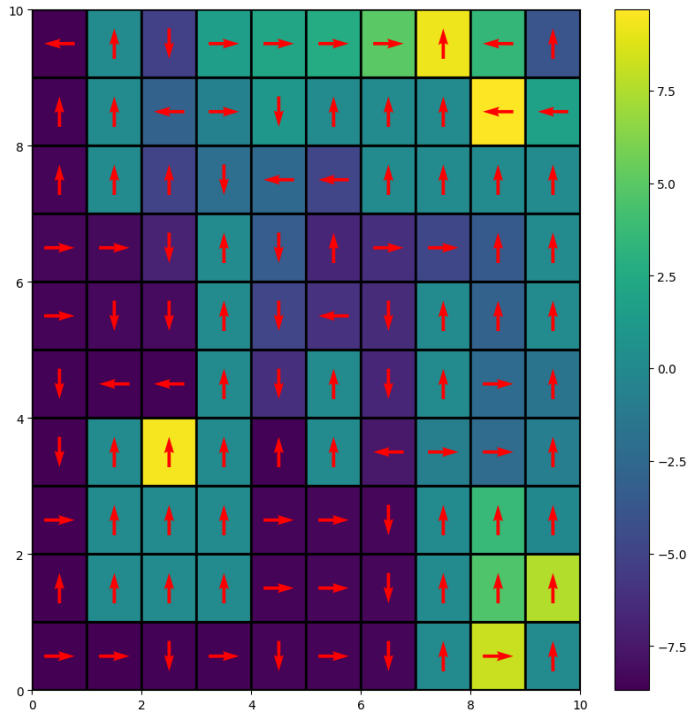
Plots for Q Learning

Wind=False **P**=0.7 **Softmax** **Start State**=[3,6]
Alpha= 0.474 **Gamma**= 0.887 **Tau**= 0.214

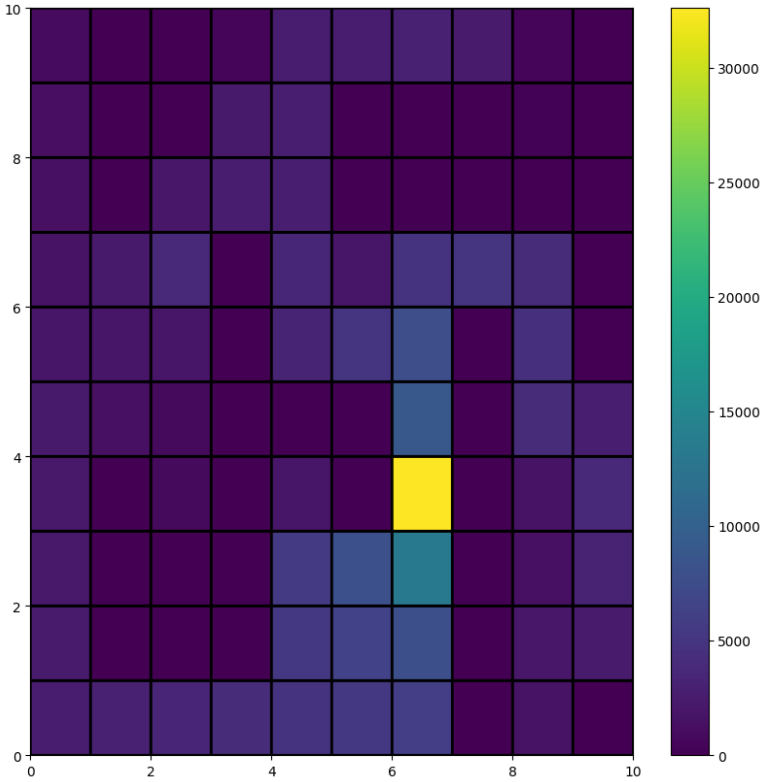




Episode 5000: Reward: -42.191919, Steps: 49.68, Qmax: 9.69, Qmin: -51.32

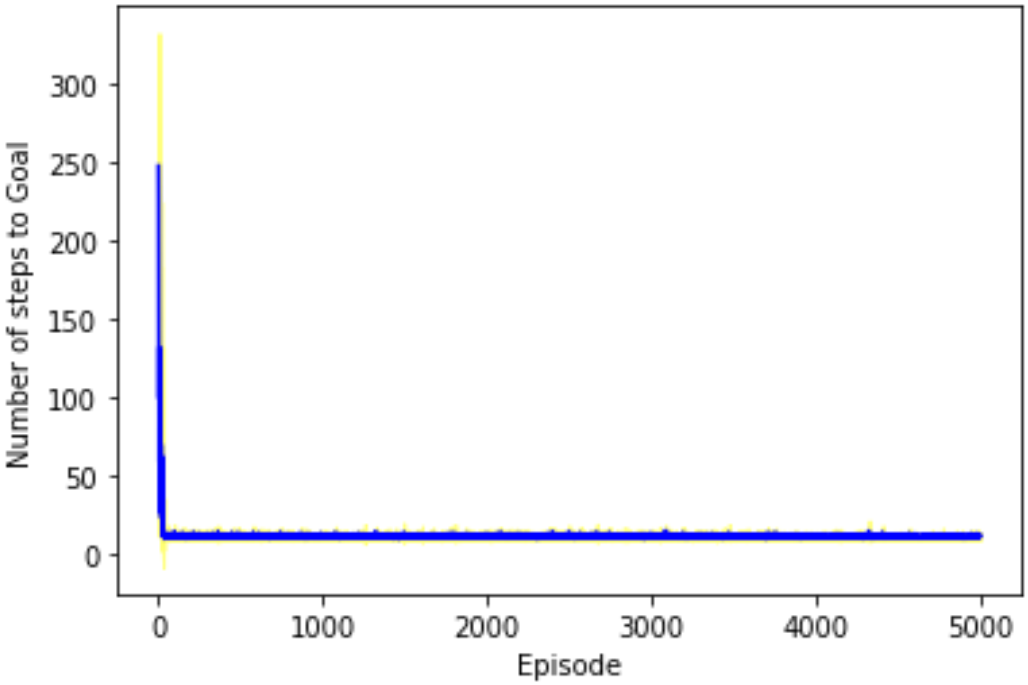


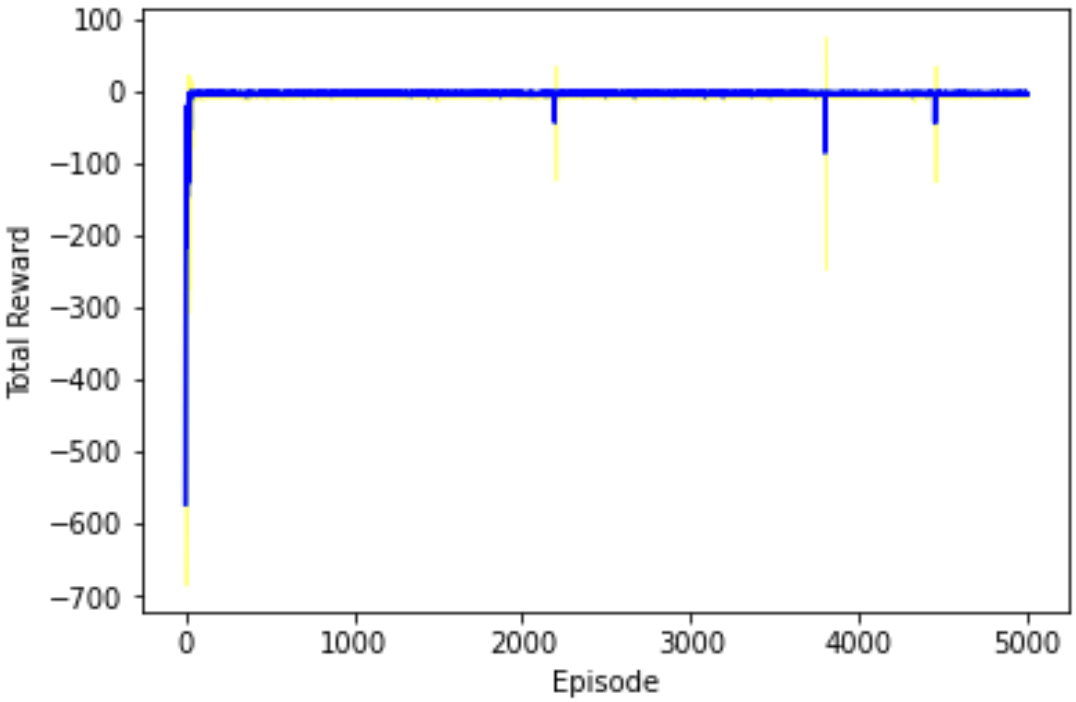
State Visit Count Plot



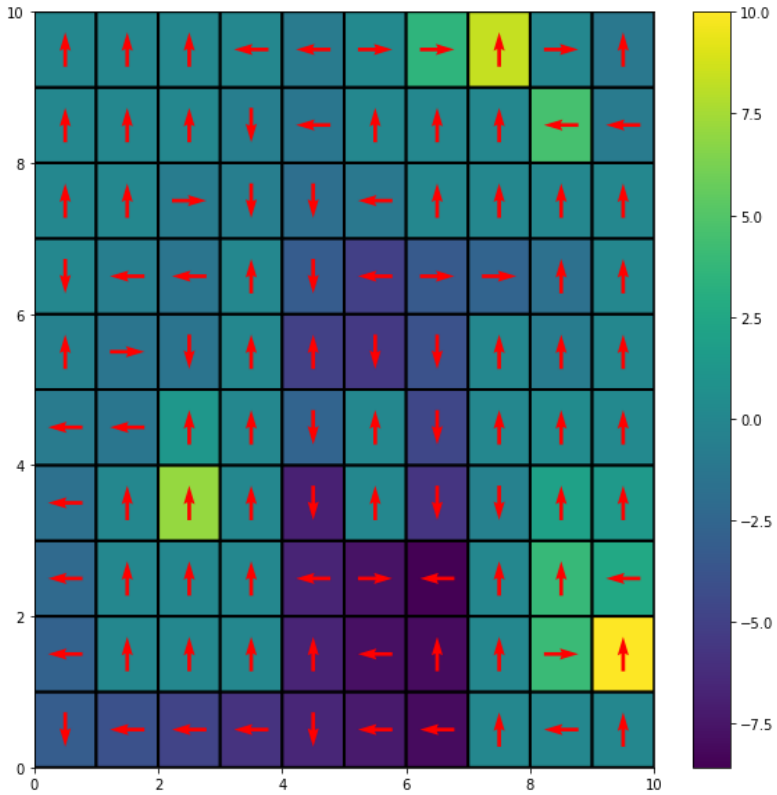
Plots for Q Learning

Wind=True **P**=1 **Epsilon Greedy** **Start State**=[3,6]
Alpha= 0.455 **Gamma**=0.999 **Epsilon**= 0.0001

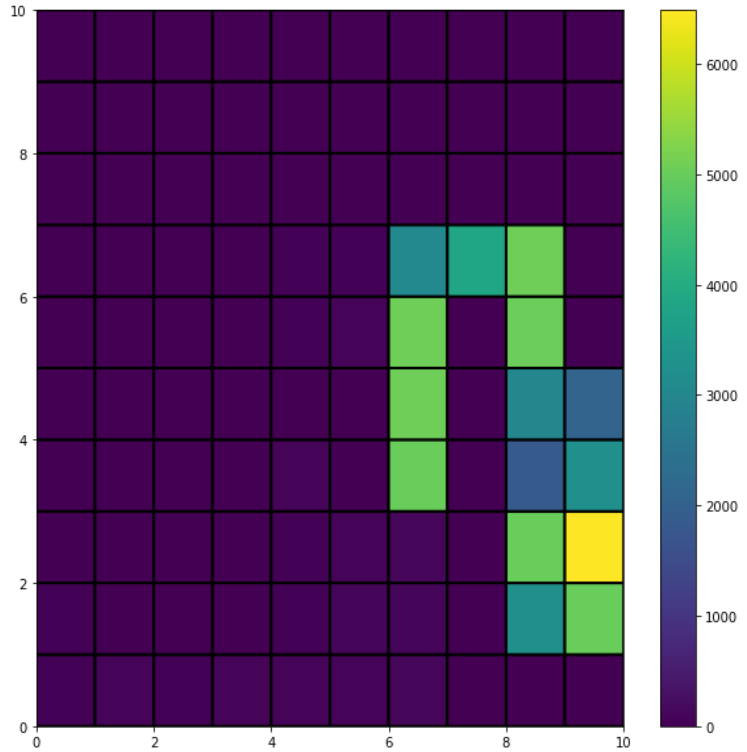




Episode 5000: Reward: -6.000000, Steps: 12.00, Qmax: 10.00, Qmin: -45.78

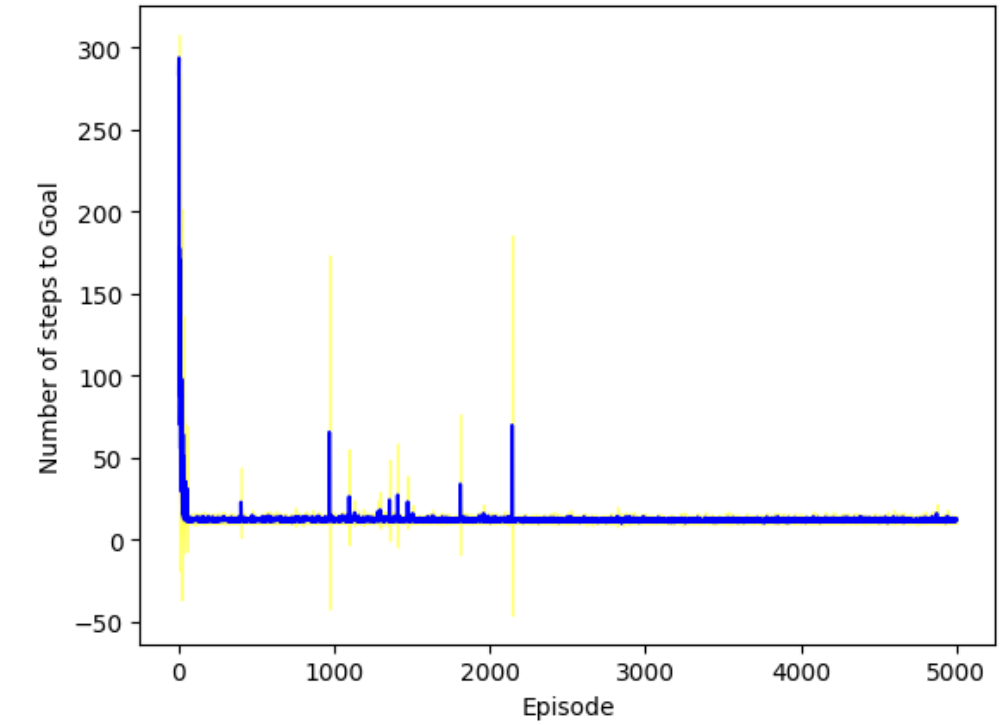


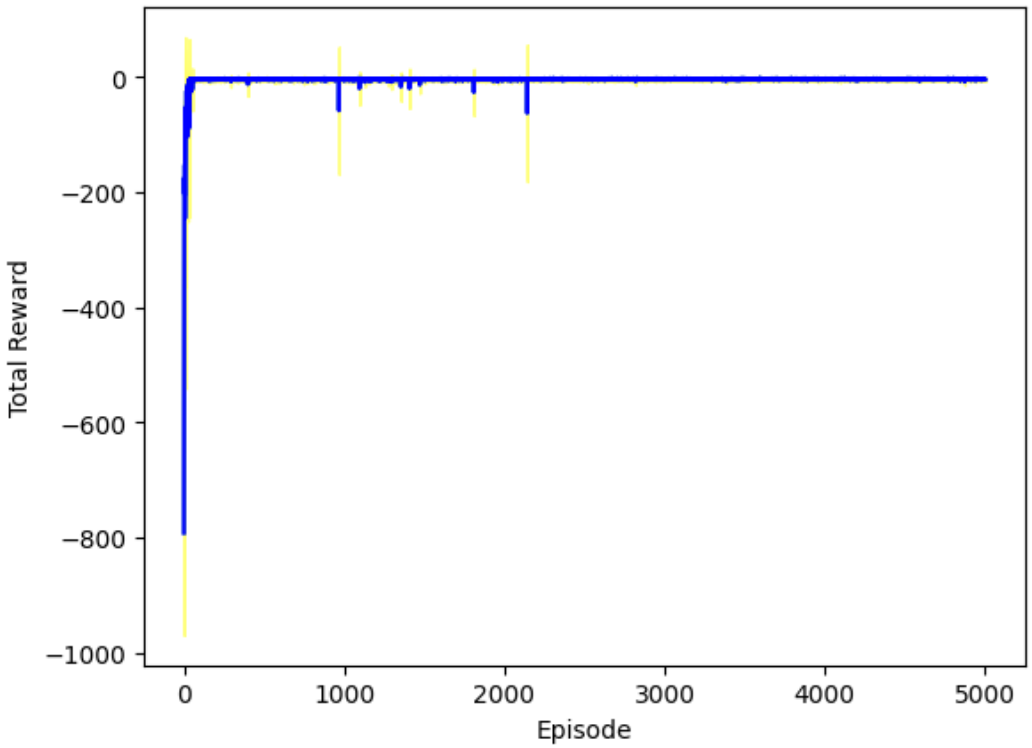
state visit count throughout all episodes



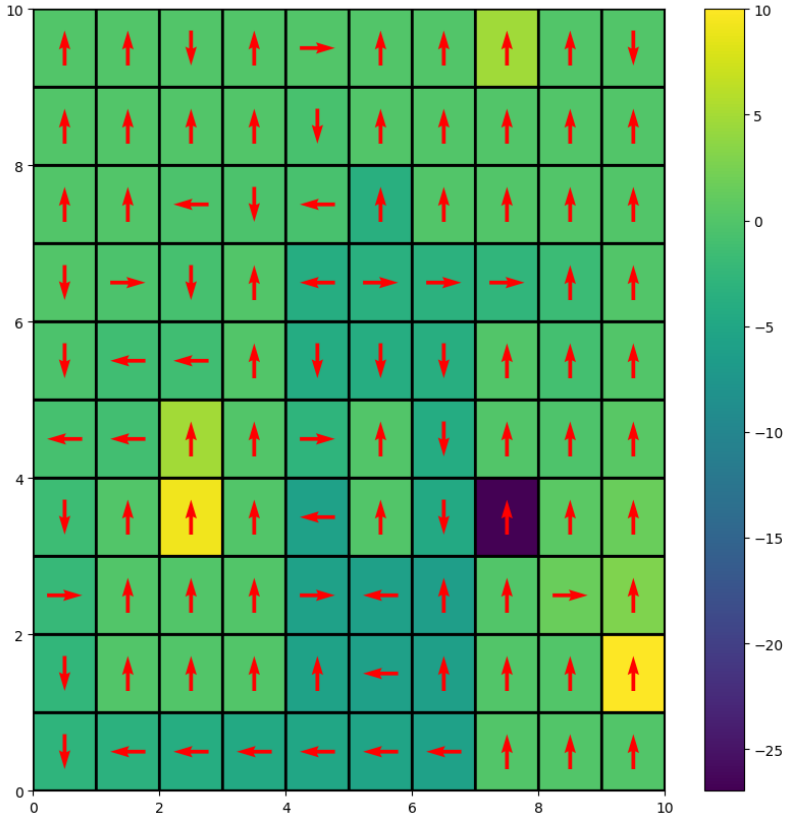
Plots for Q Learning

Wind=True **P**=1 **Softmax** **Start State**=[3,6]
Alpha= 0.474 **Gamma**= 0.887 **Tau**= 0.214

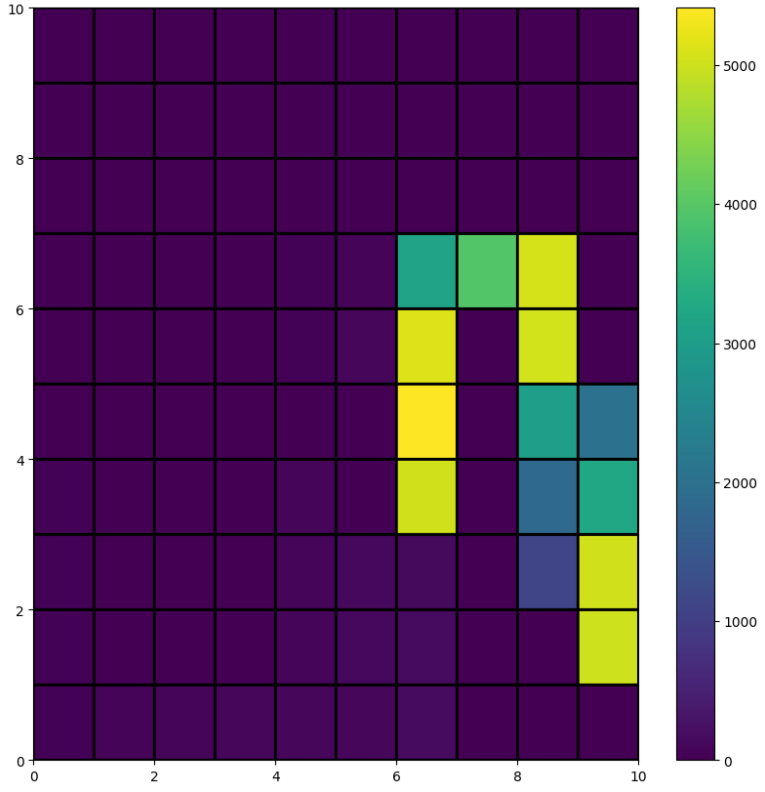




Episode 5000: Reward: -5.747475, Steps: 11.75, Qmax: 10.00, Qmin: -101.87

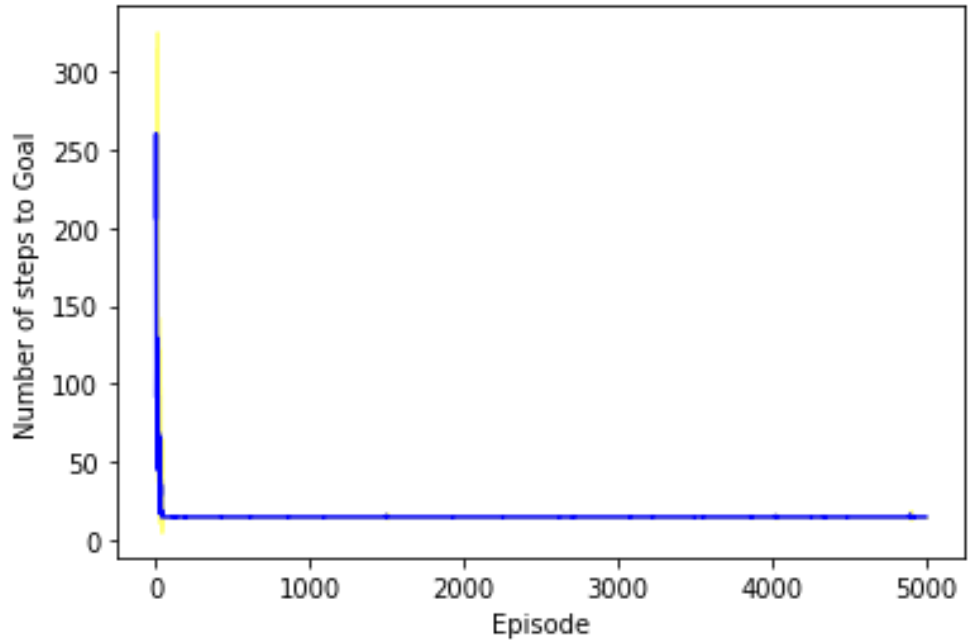


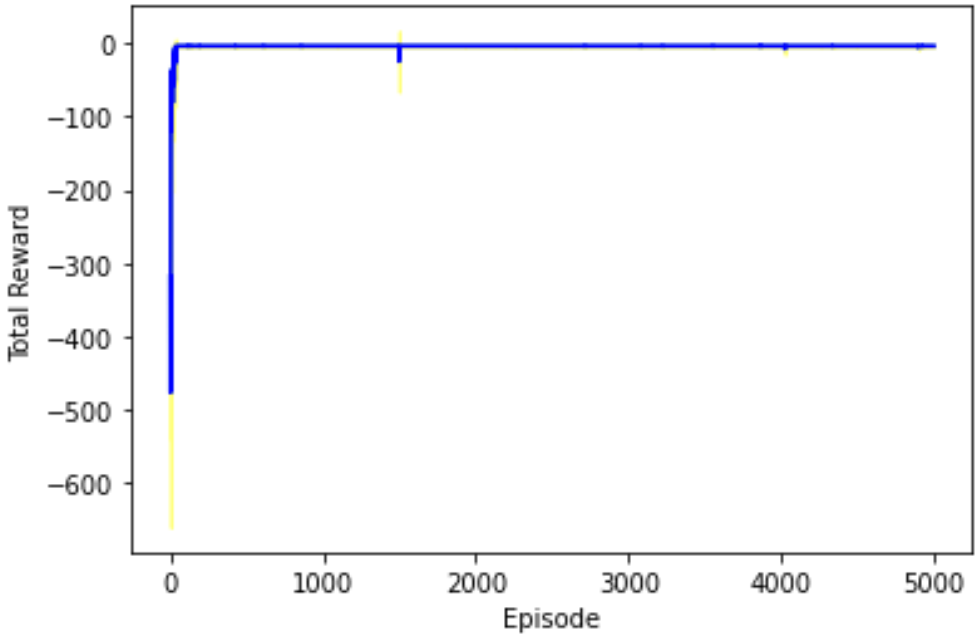
State Visit Count Plot



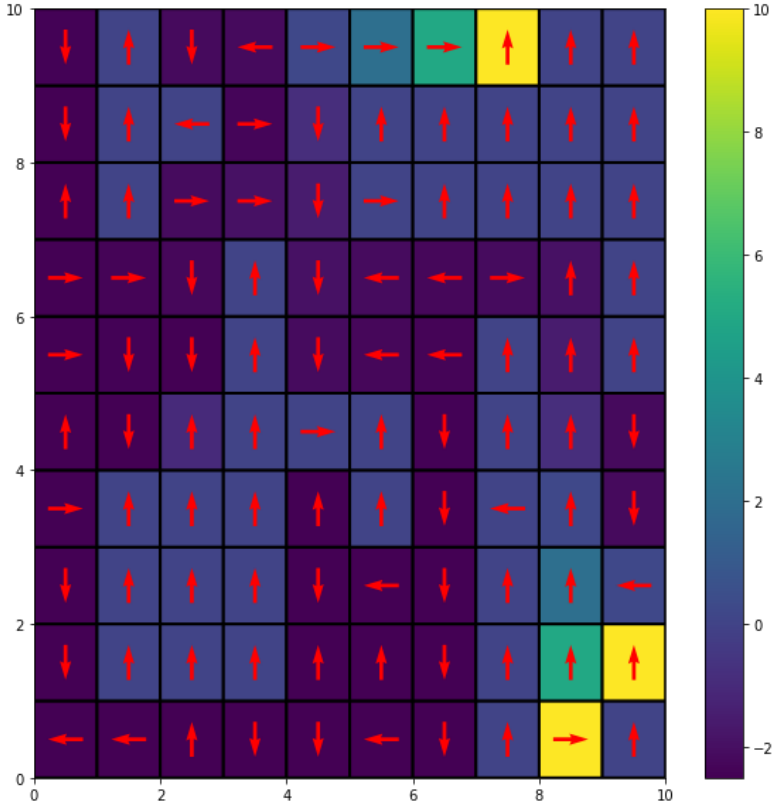
Plots for Q Learning

Wind=False **P**=1 **Epsilon Greedy** **Start State**=[0,4]
Alpha= 0.999 **Gamma**=0.6 **Epsilon**= 0.0001

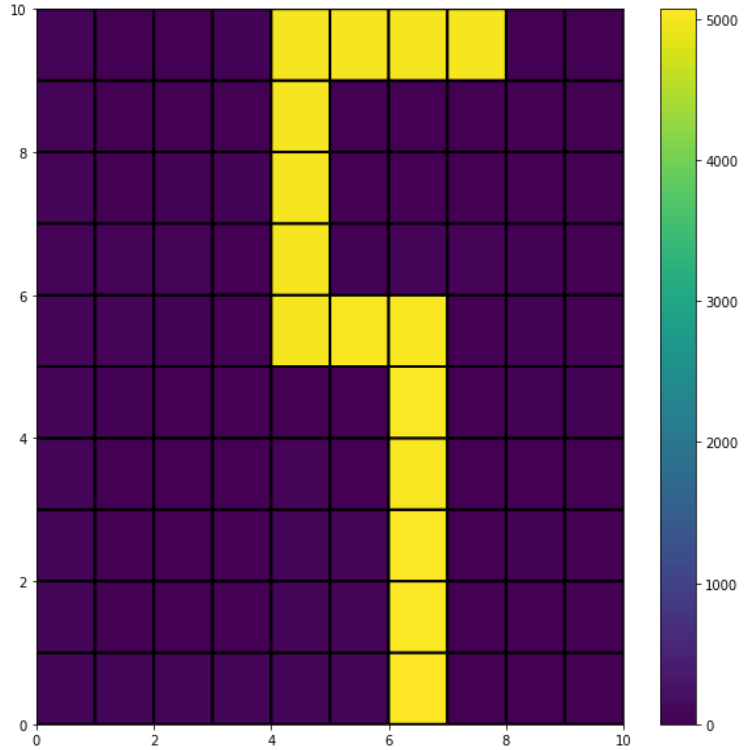




Episode 5000: Reward: -4.000000, Steps: 15.00, Qmax: 10.00, Qmin: -99.90

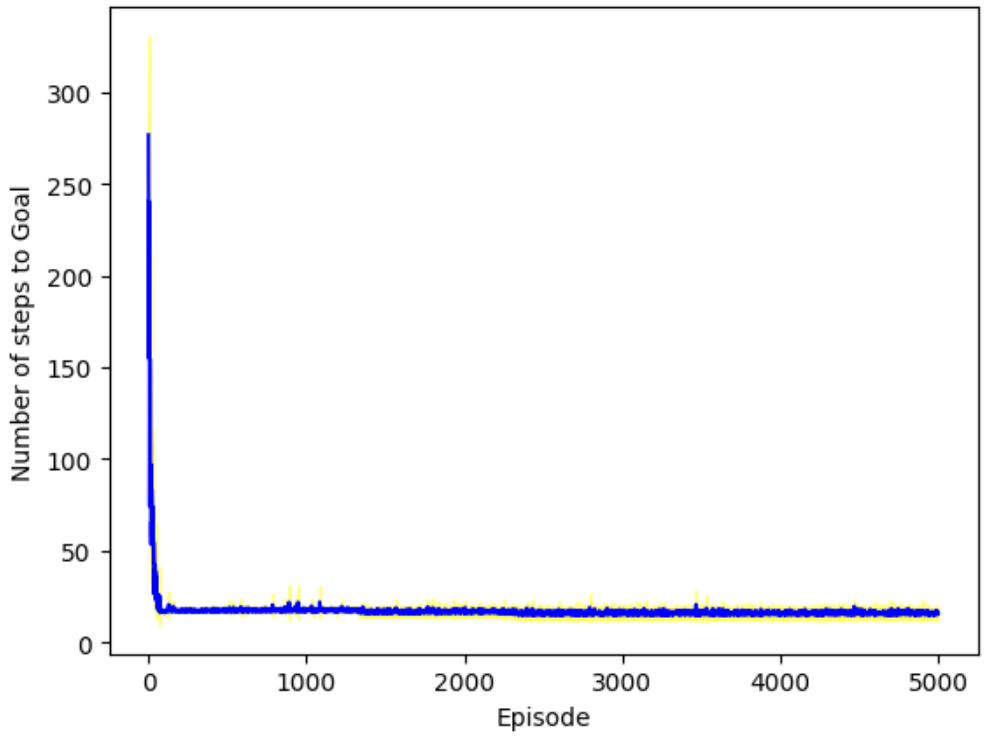


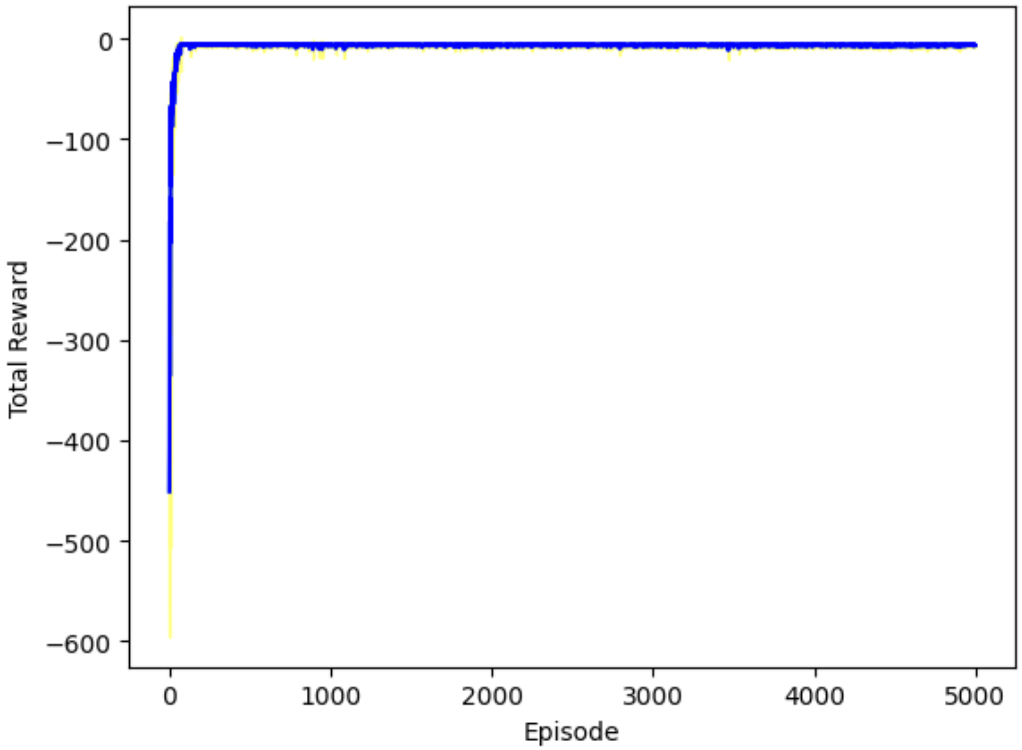
state visit count throughout all episodes



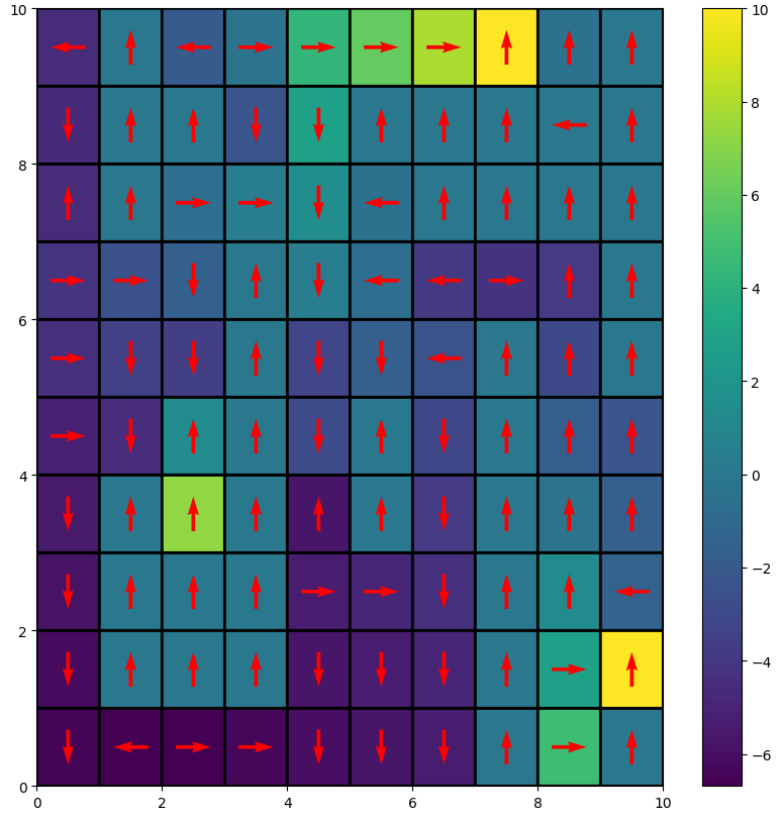
Plots for Q Learning

Wind=False **P**=1 **Softmax** **Start State**=[0,4]
Alpha= 0.474 **Gamma**= 0.887 **Tau**= 0.214

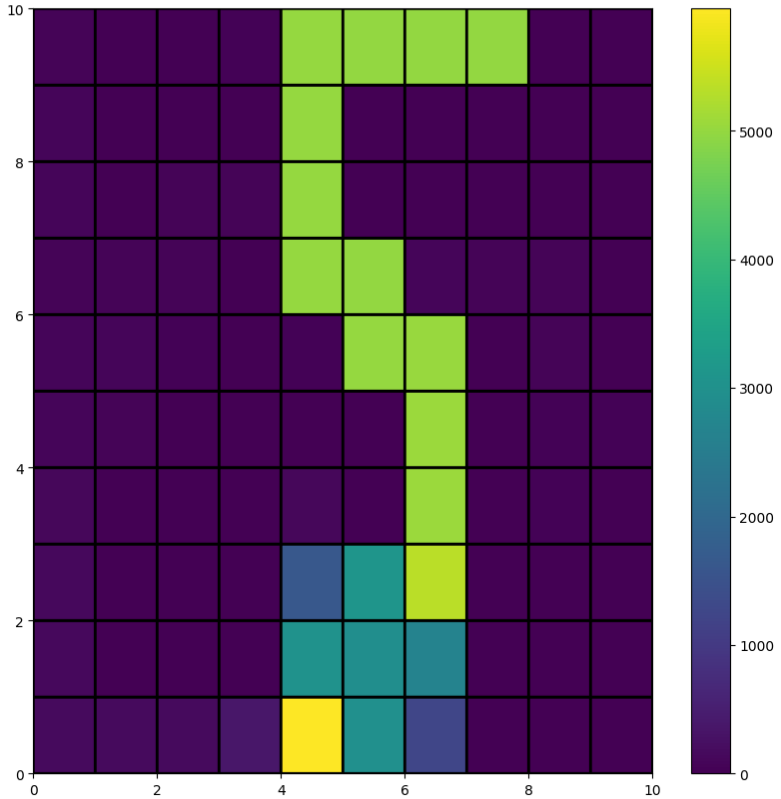




Episode 5000: Reward: -6.848485, Steps: 17.85, Qmax: 10.00, Qmin: -47.49

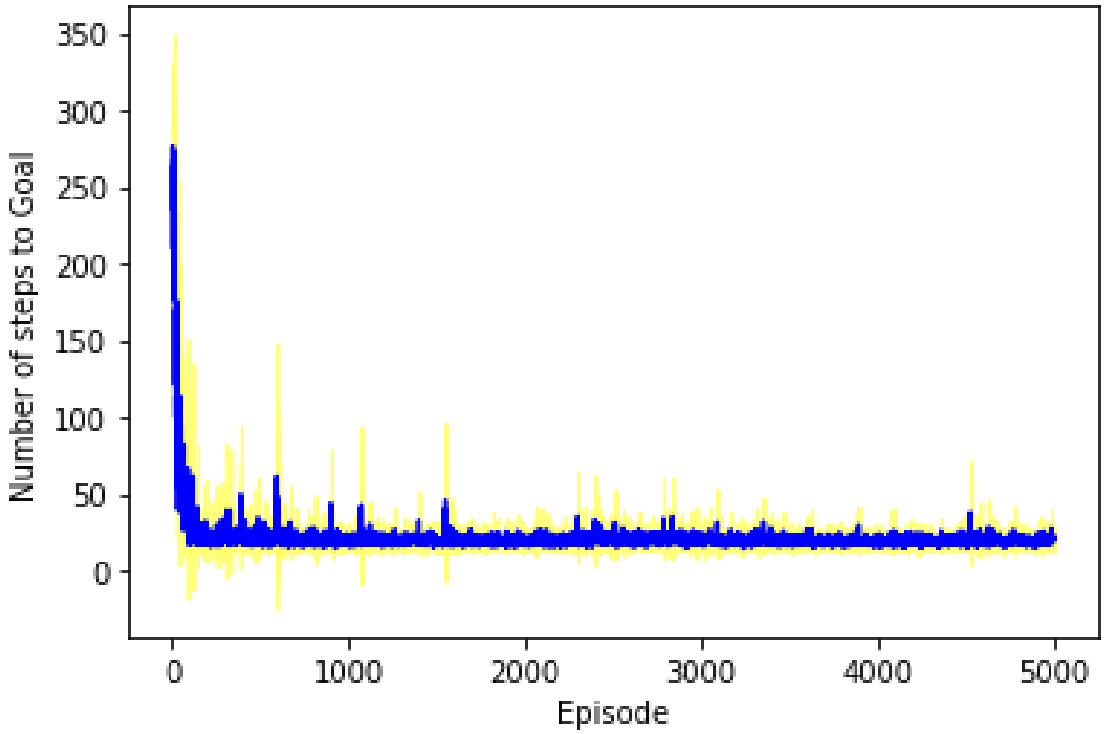


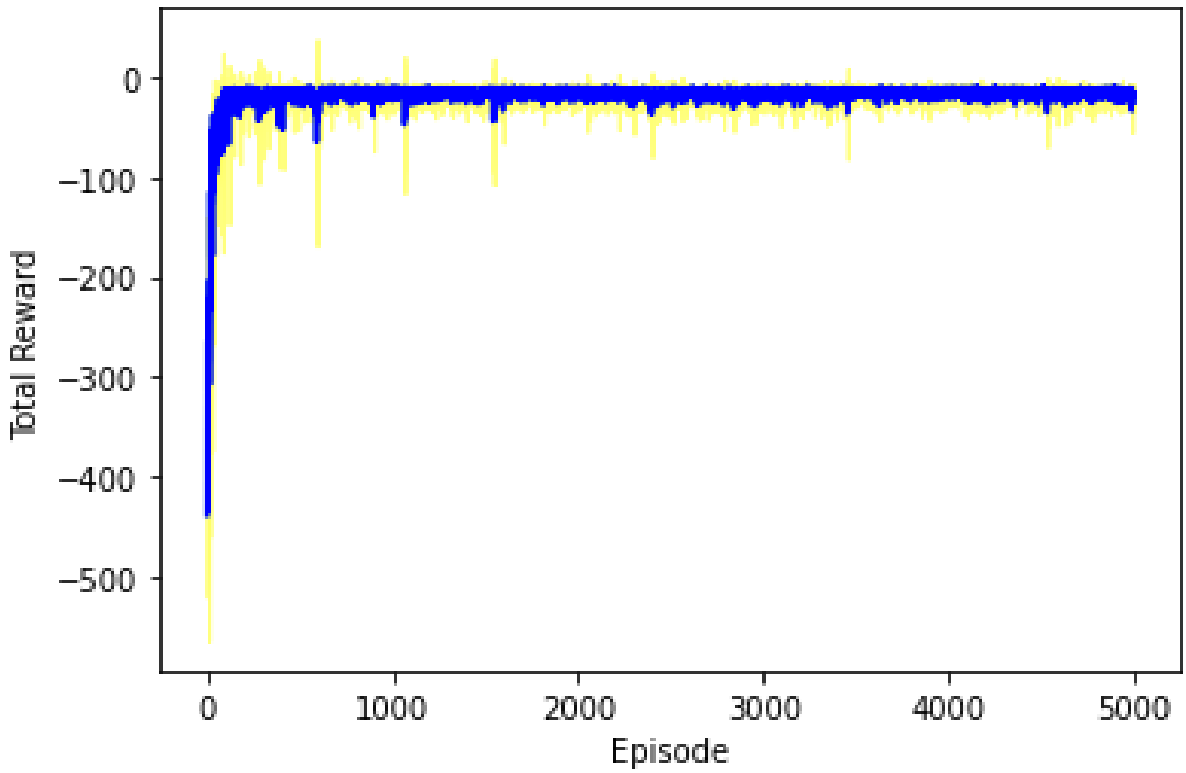
State Visit Count Plot



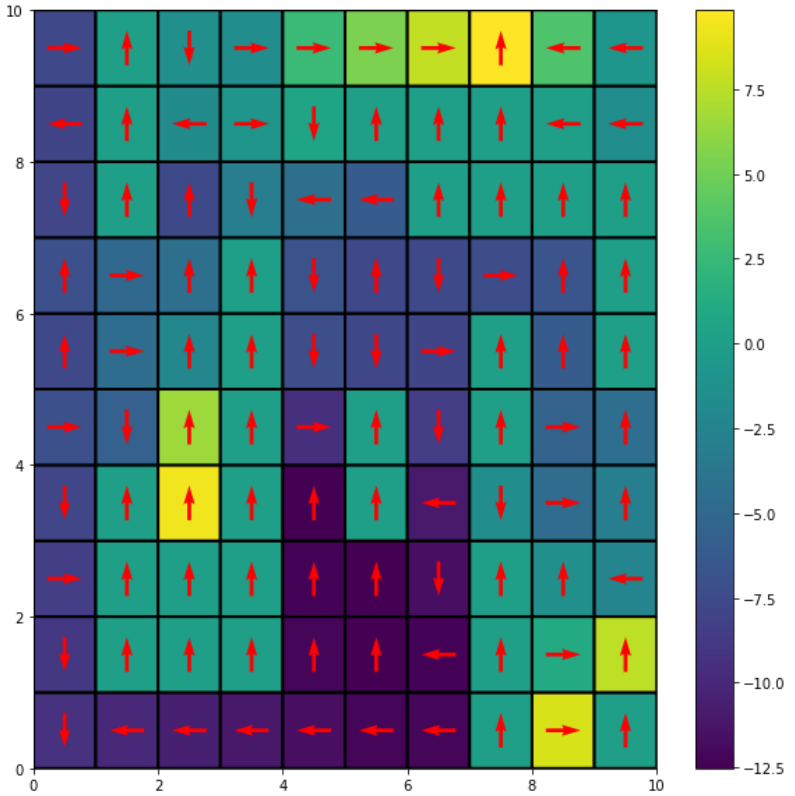
Plots for Q Learning

Wind=False **P**=0.7 **Epsilon Greedy** **Start State**=[0,4]
Alpha= 0.2756 **Gamma**=0.940 **Epsilon**= 0.005979

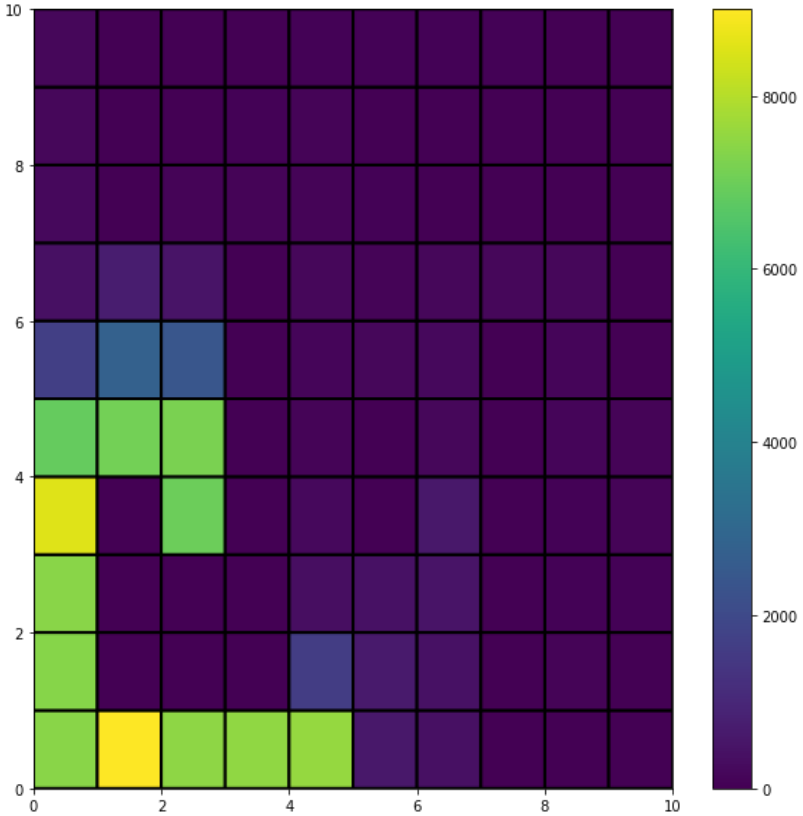




Episode 5000: Reward: -21.393939, Steps: 24.46, Qmax: 9.82, Qmin: -47.60

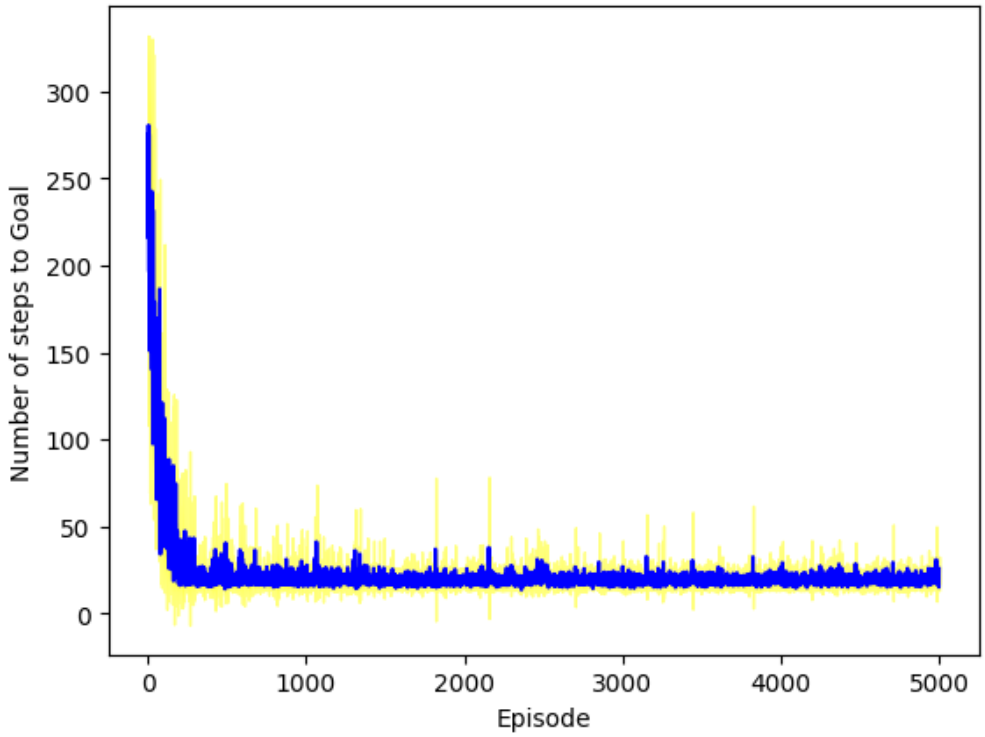


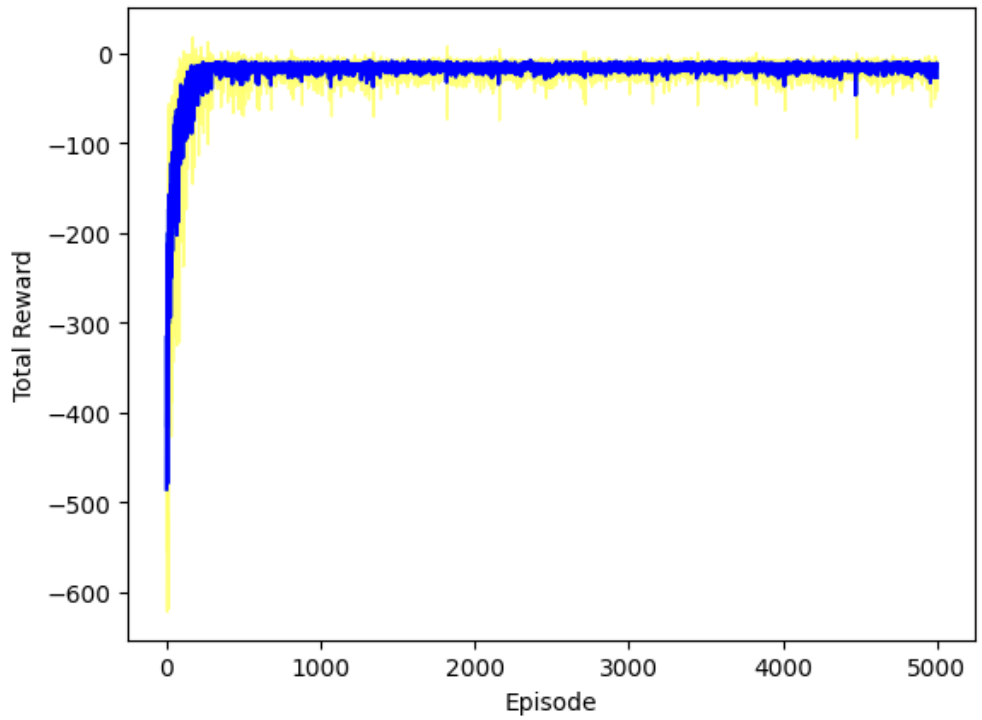
state visit count throughout all episodes



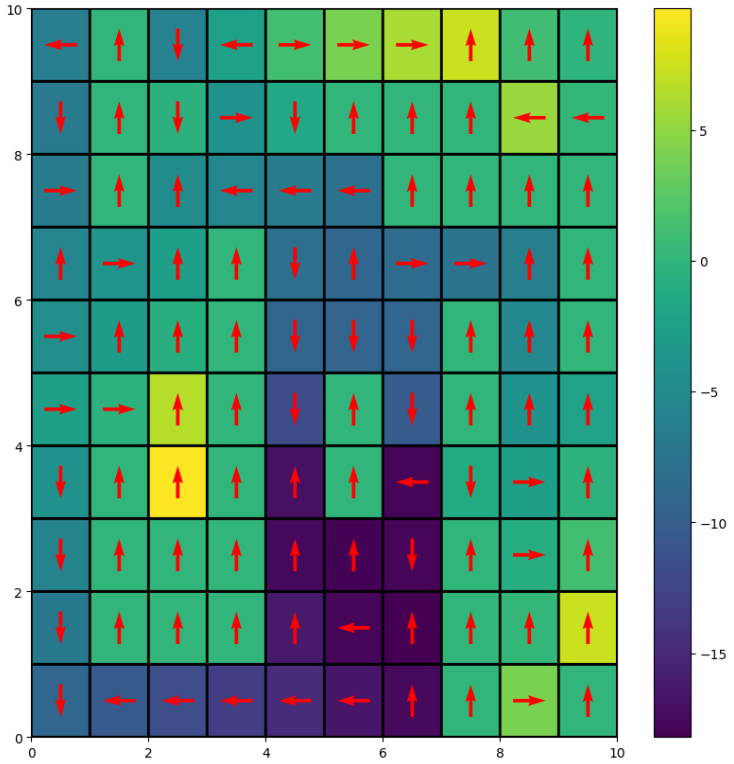
Plots for Q Learning

Wind=False **P**=0.7 **Softmax** **Start State**=[0,4]
Alpha= 0.1 **Gamma**= 0.999 **Tau**= 0.1

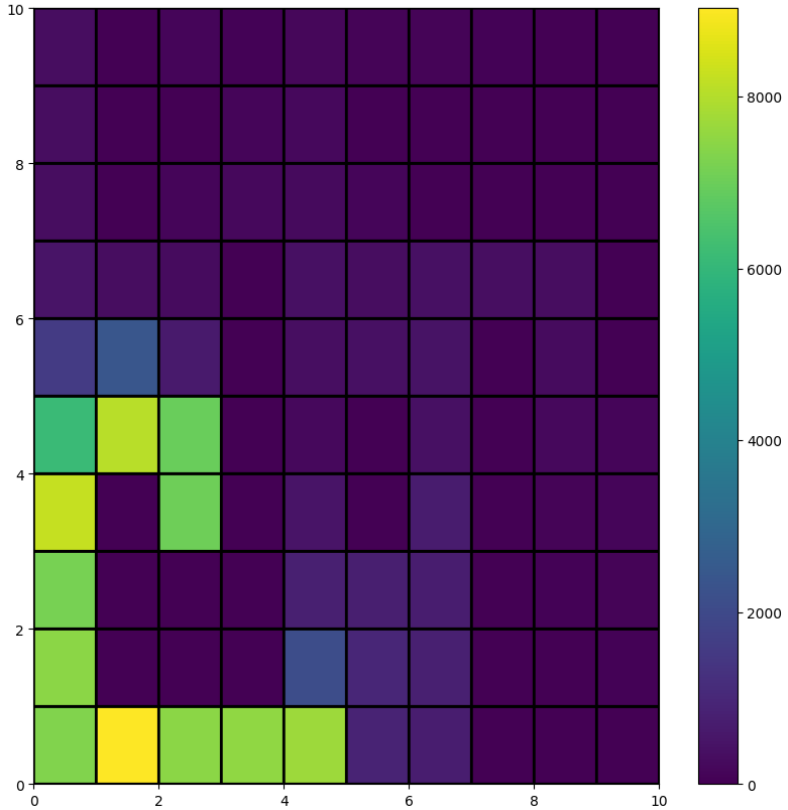




Episode 5000: Reward: -14.939394, Steps: 18.97, Qmax: 9.66, Qmin: -23.52

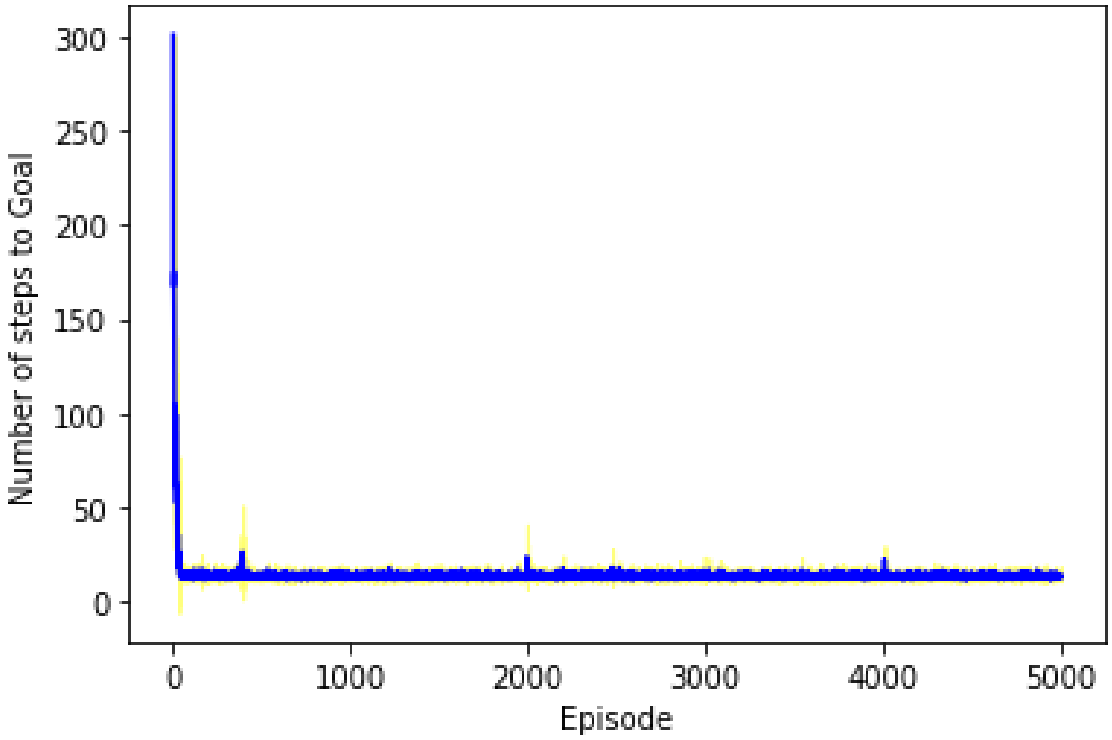


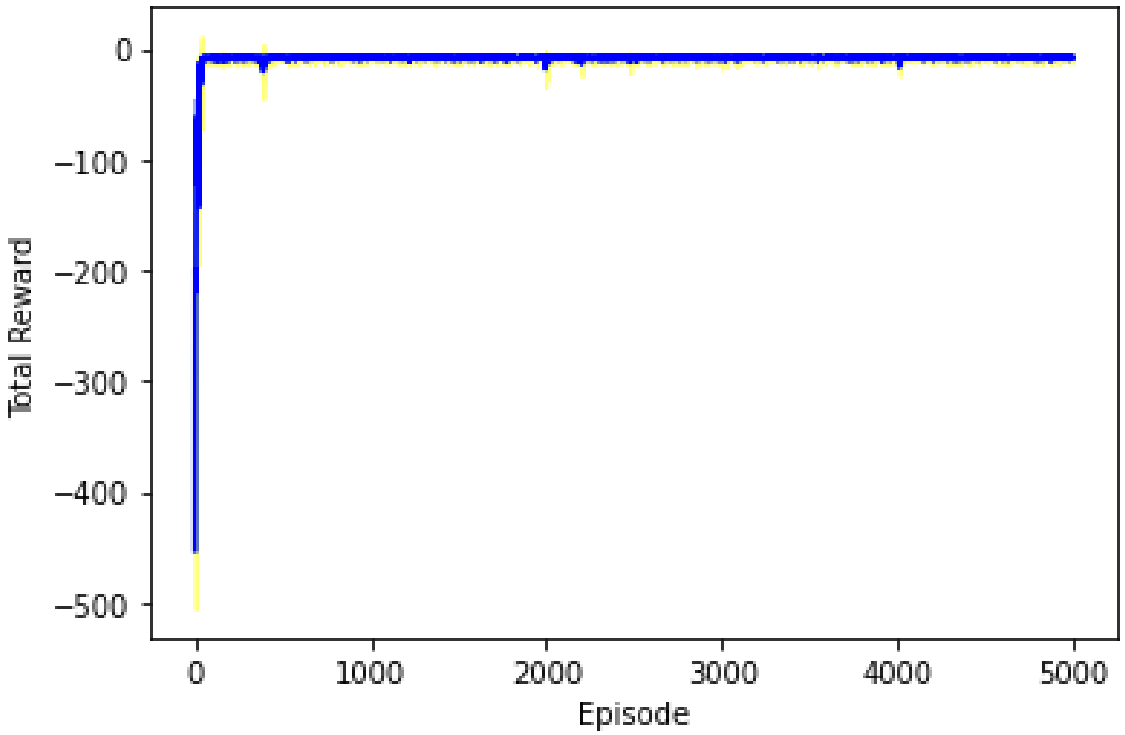
State Visit Count Plot



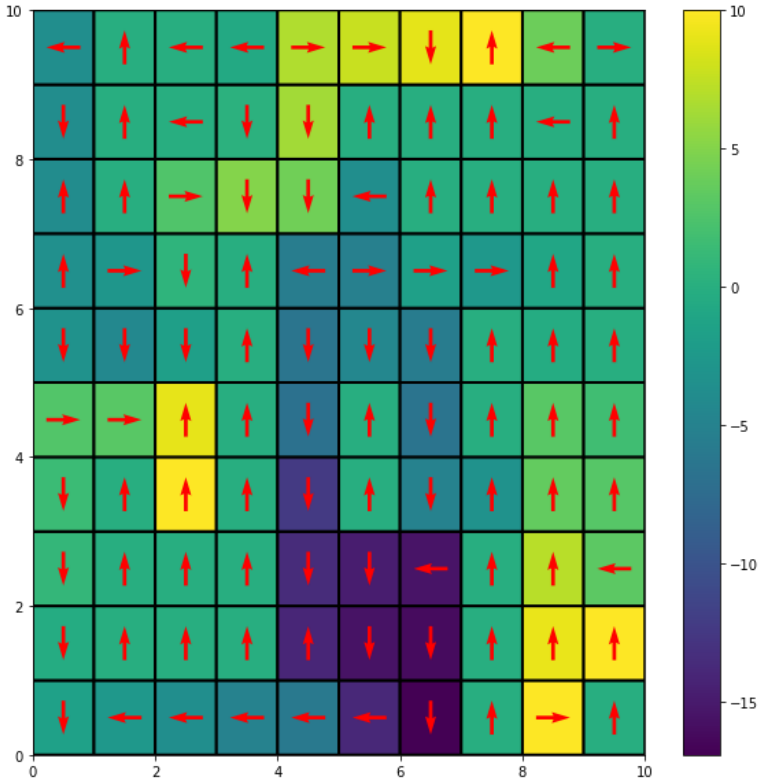
Plots for Q Learning

Wind=True **P**=1 **Epsilon greedy** **Start State**=[0,4]
Alpha= 0.68046 **Gamma**= 0.999 **Epsilon**= 0.0001

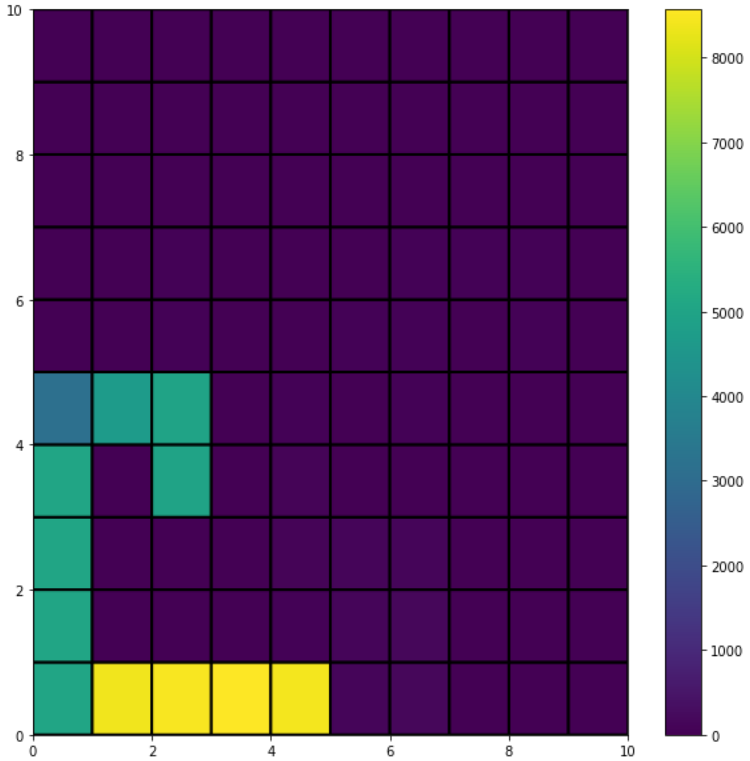




Episode 5000: Reward: -7.828283, Steps: 13.83, Qmax: 10.00, Qmin: -70.09

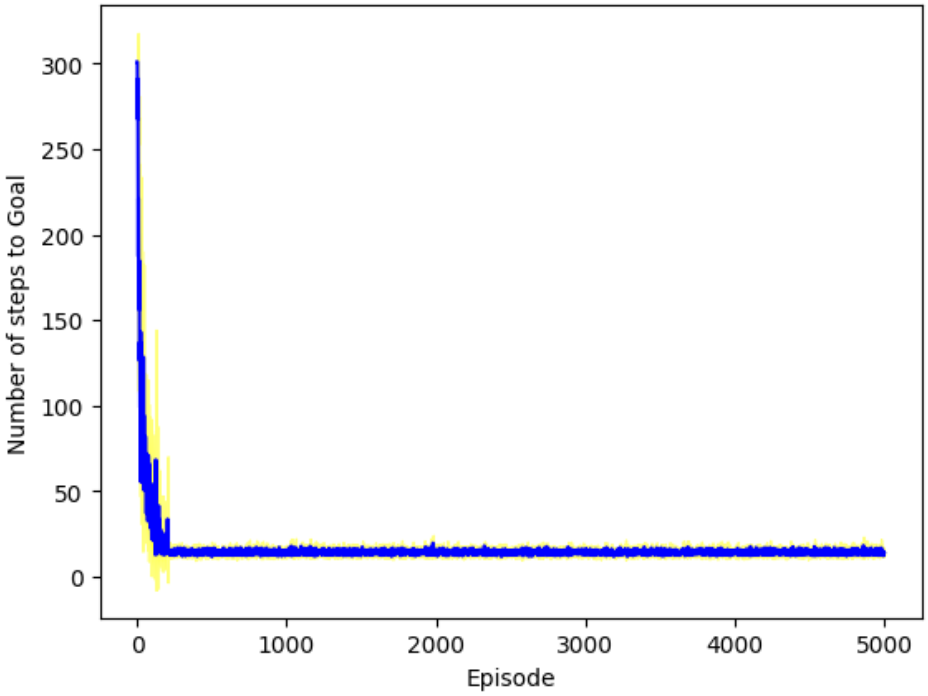


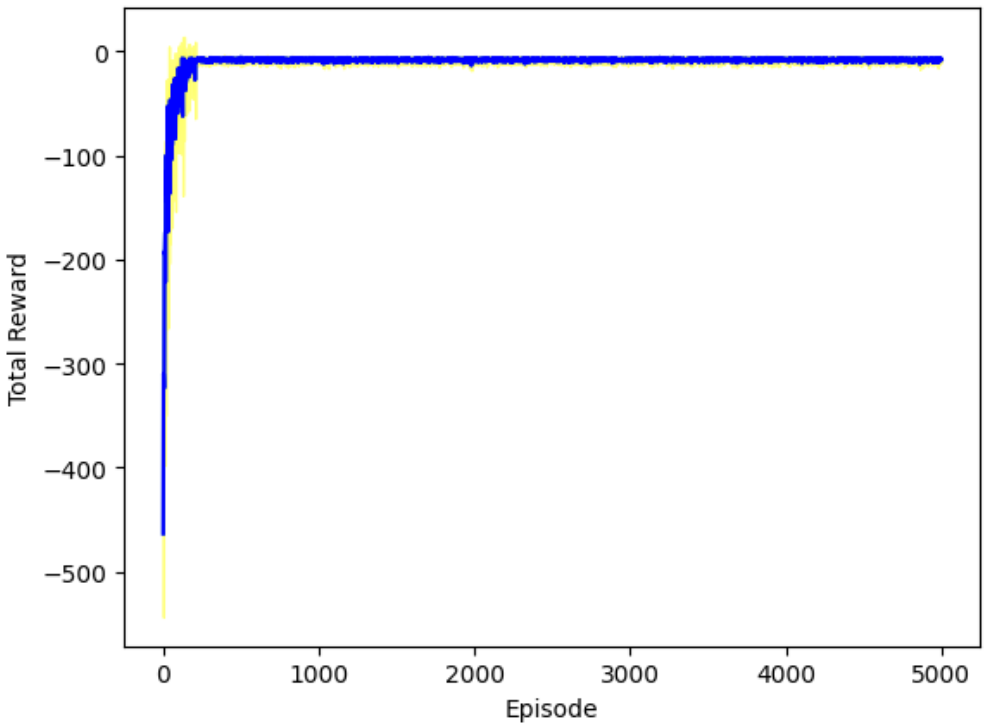
state visit count throughout all episodes



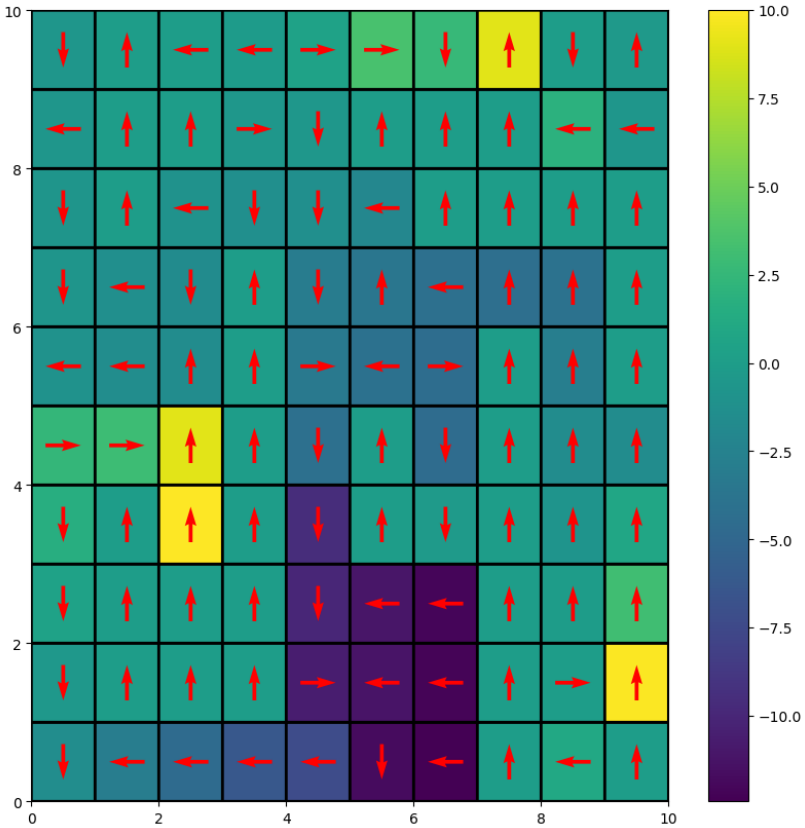
Plots for Q Learning

Wind=True **P**=1 **Softmax** **Start State**=[0,4]
Alpha= 0.1 **Gamma**= 0.999 **Tau**= 0.1

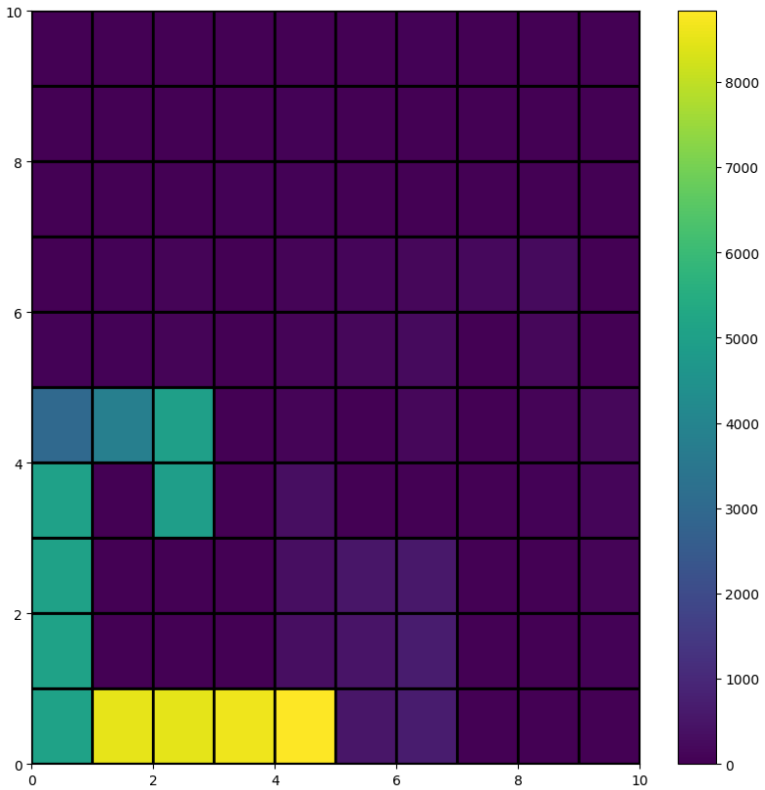




Episode 5000: Reward: -7.868687, Steps: 13.87, Qmax: 10.00, Qmin: -18.52



State Visit Count Plot



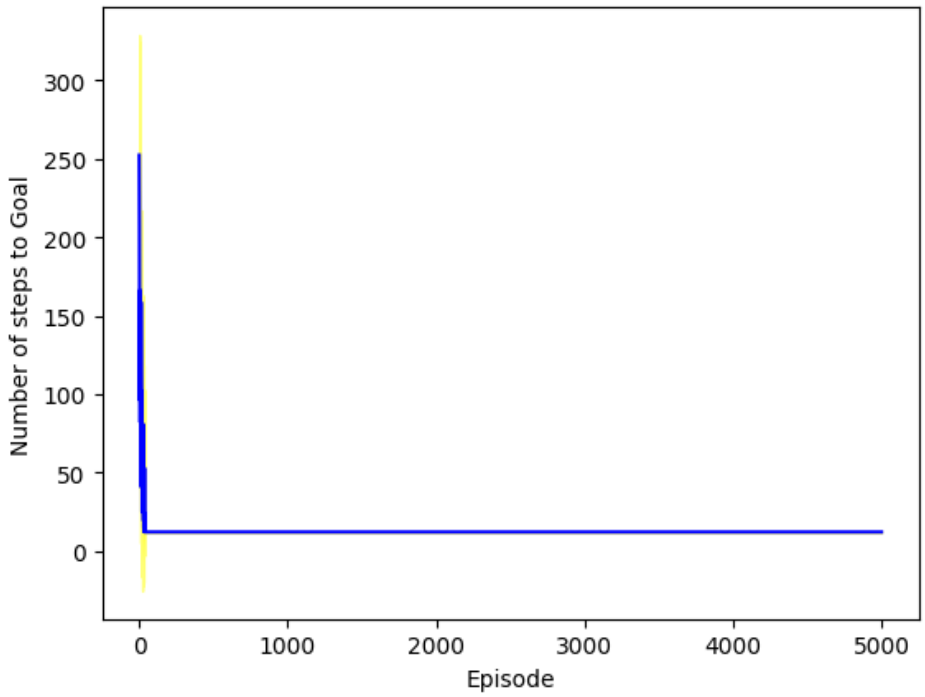
Softmax Iteration-2

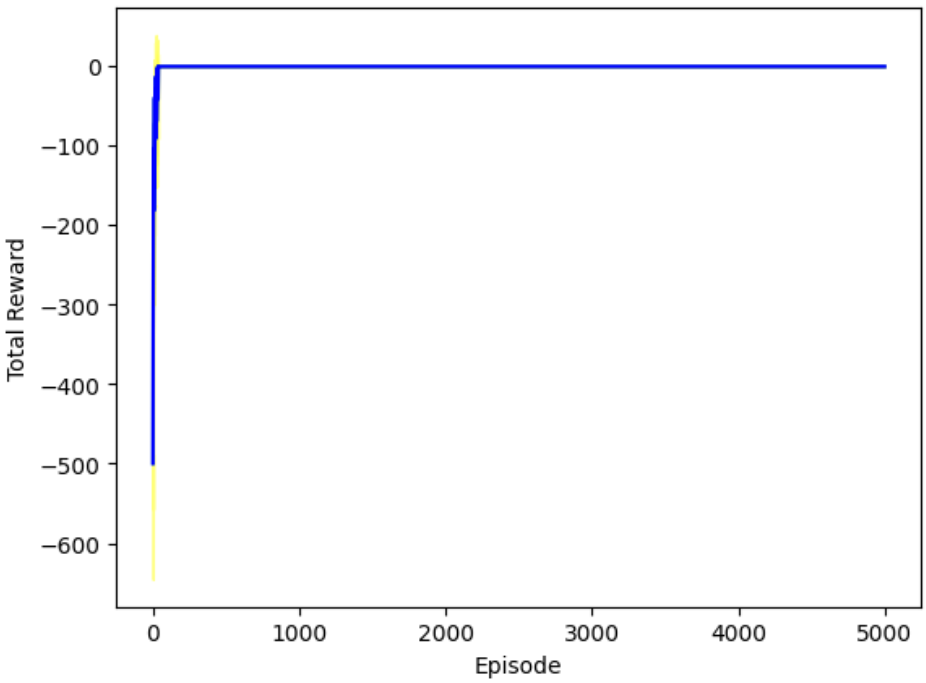
From the above optimization for different experiments, it is clear that softmax seems to have very similar results in terms of the hyperparameters. This is because the tau was iterated over a very large range of values (0.1,1000). From the 29 experiments, it made clear that tau must have a smaller value, implying that we don't need a significantly large degree of exploration or randomness in our strategy. Therefore, we need to run a second iteration of optimization of the Tau values to perform a confined space search, ie, search the range more relevant to us to maximize the average rewards. As tau increases, it is observed that the rewards are decreasing by a lot, just 5001 episodes of training, so larger tau requires a larger number of episodes.

So the second iteration will be to search for optimal tau within the range of (0.1,5), using bayesian optimization, using 10 iterations. All the ranges of other parameters i.e., gamma and alpha, are kept the same as they seem relevant and possibly optimal from the first experiments.

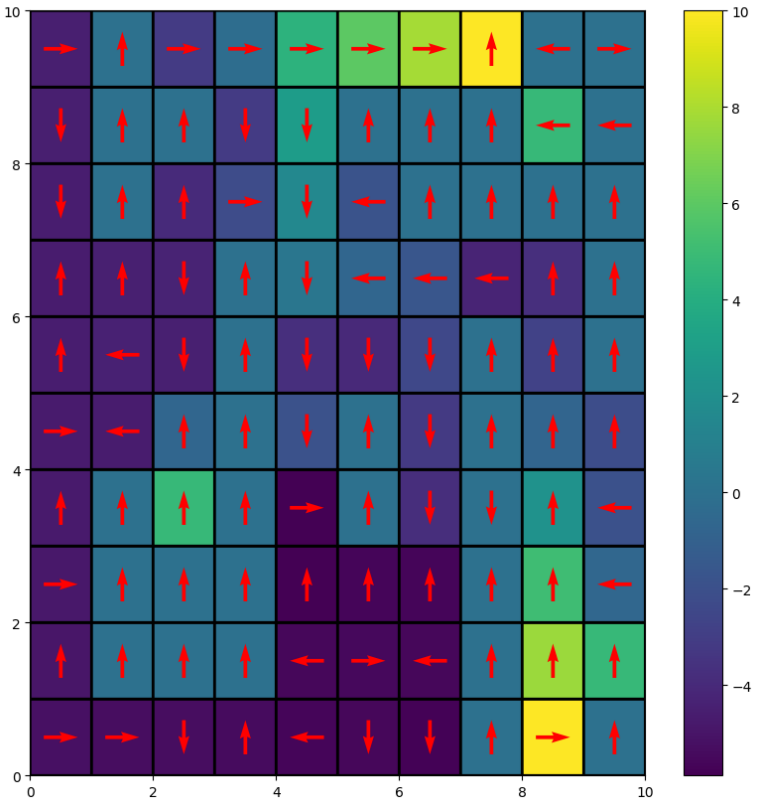
Plots for SARSA

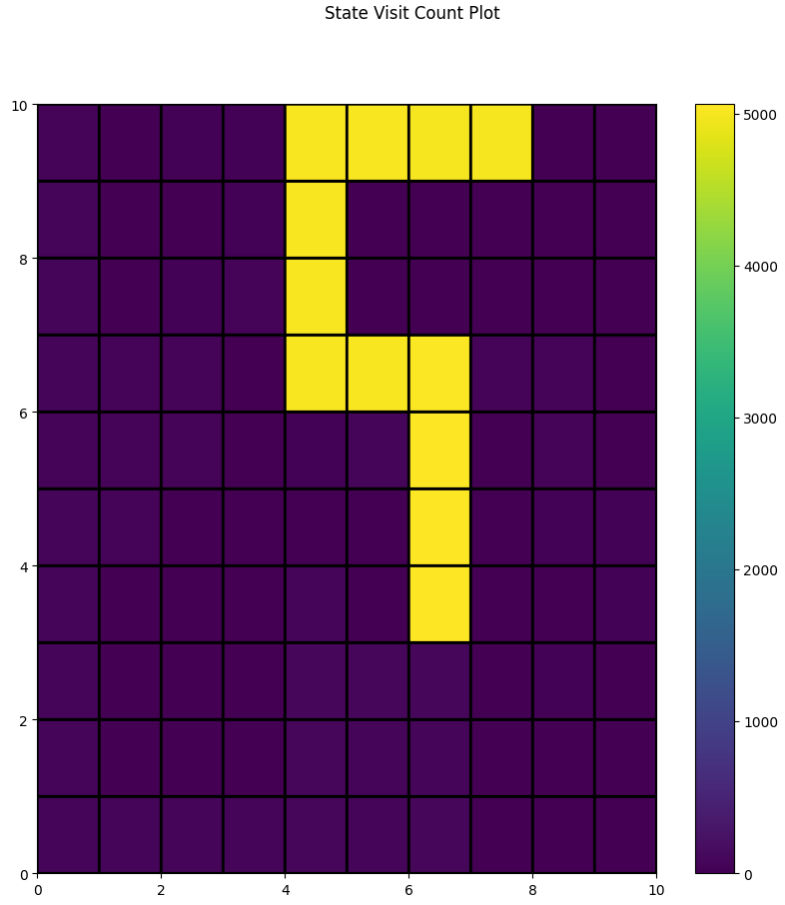
Wind=False **P**=1 **Softmax-2** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.100





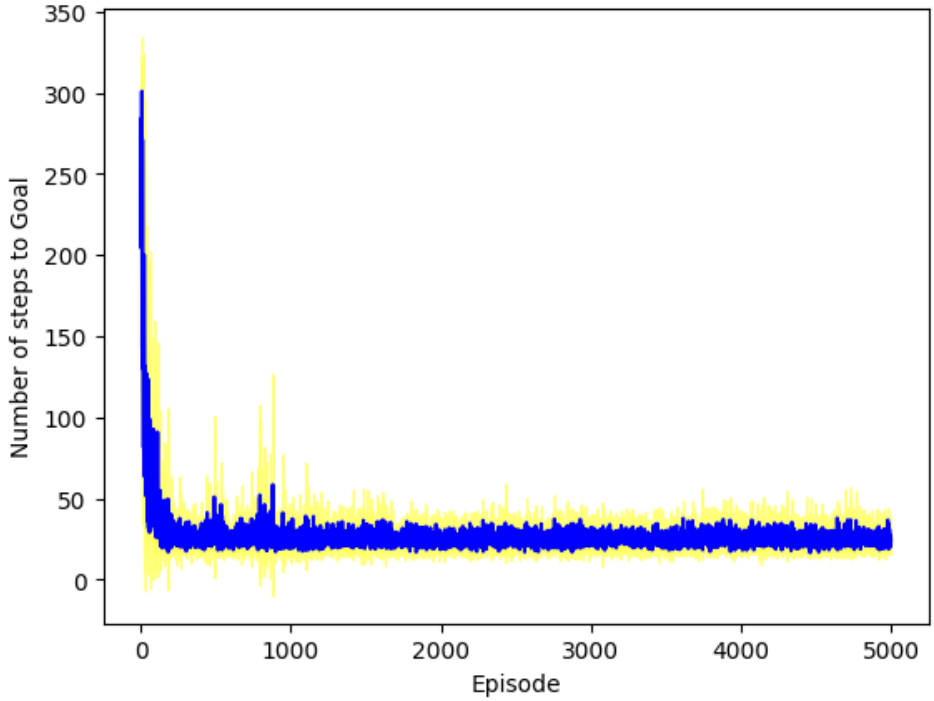
Episode 5000: Reward: -1.000000, Steps: 12.00, Qmax: 10.00, Qmin: -47.49

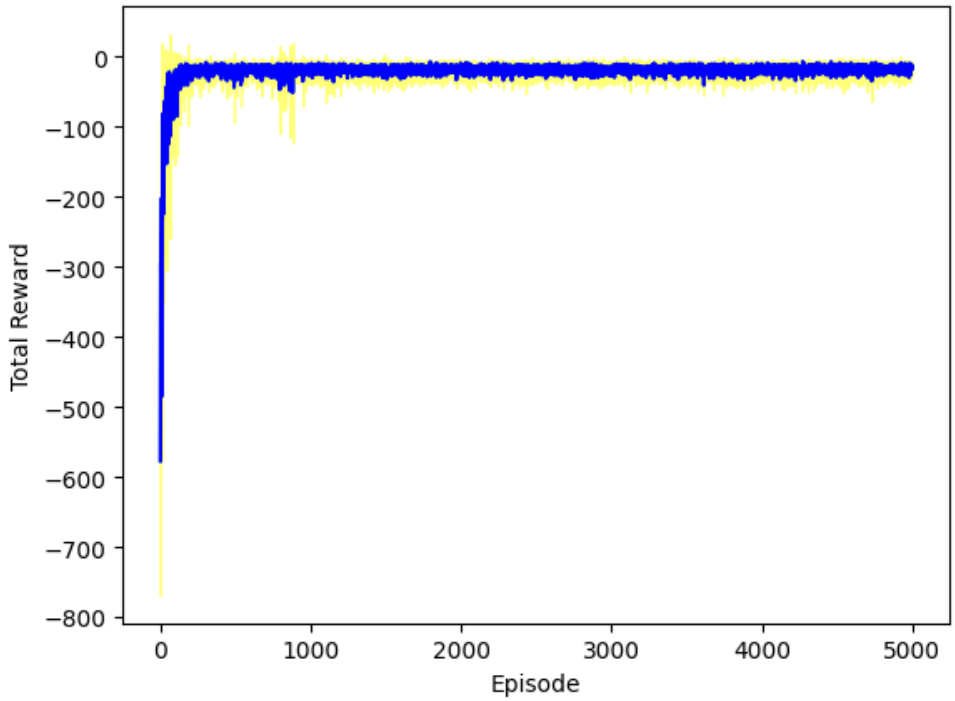



State Visit Count Plot
10
8
6
4
2
0
0
2
4
6
8
10
5000
4000
3000
2000
1000
0

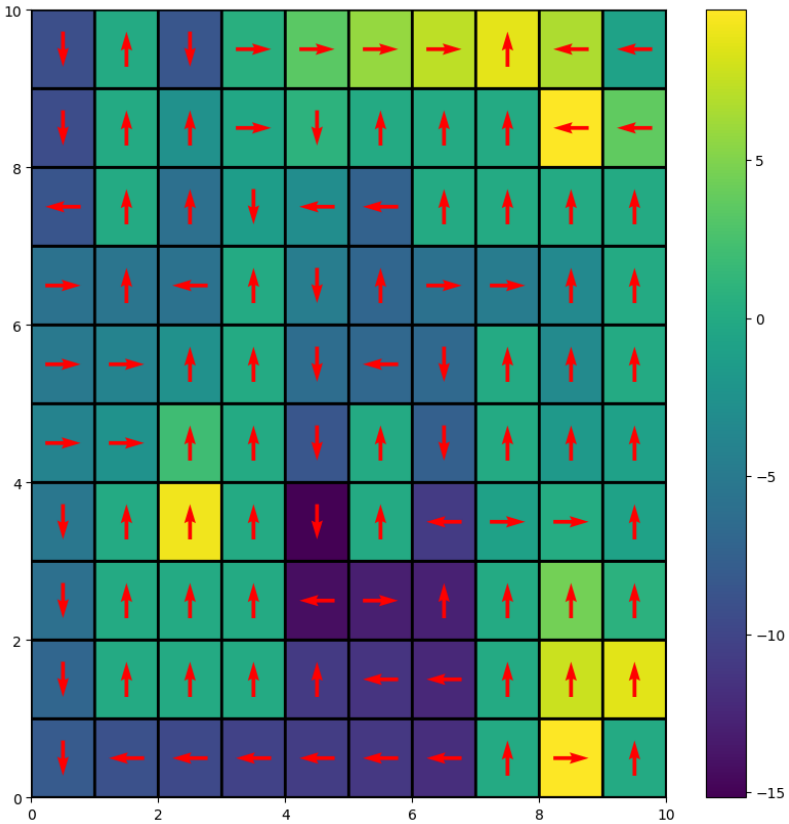
Plots for SARSA

Wind=False **P**=0.7 **Softmax-2** **Start State**=[3,6]
Alpha= 0.283 **Gamma**=0.950 **Tau**= 0.234

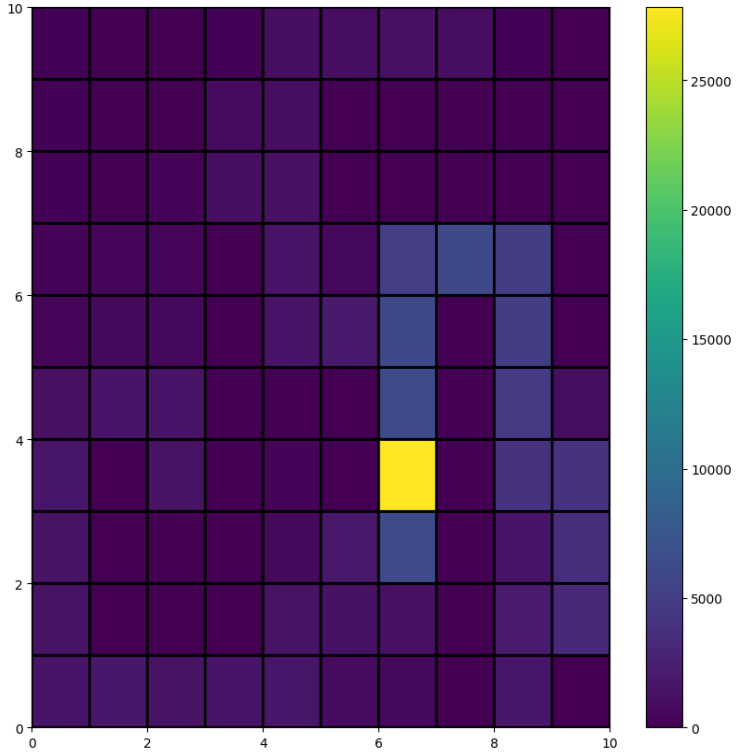




Episode 5000: Reward: -18.000000, Steps: 24.15, Qmax: 9.75, Qmin: -30.60

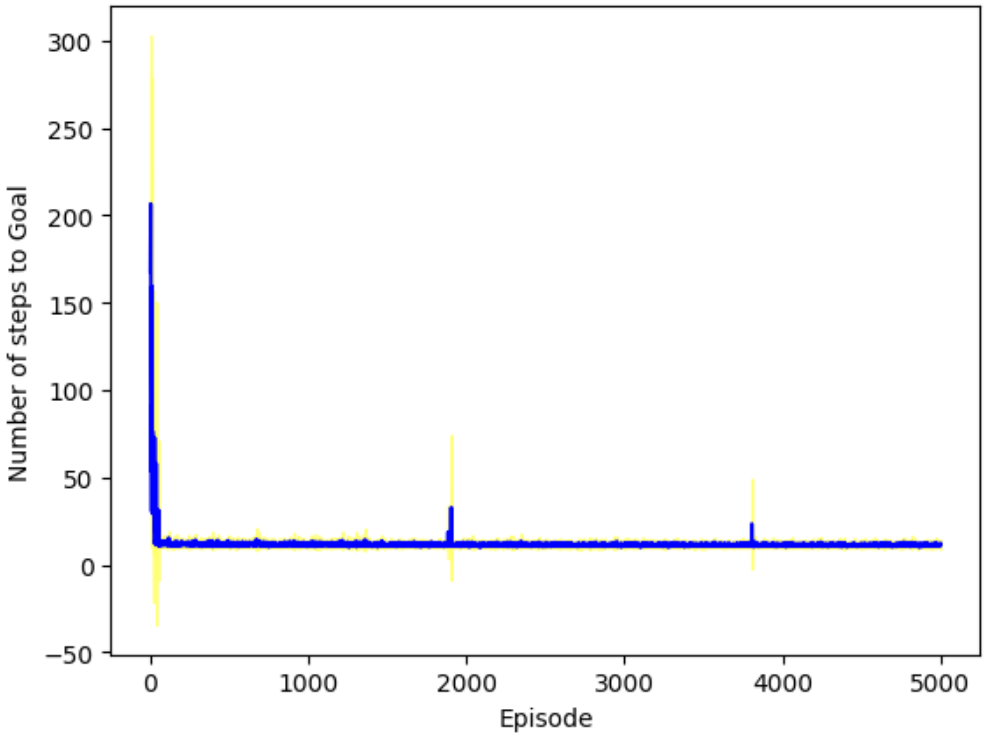


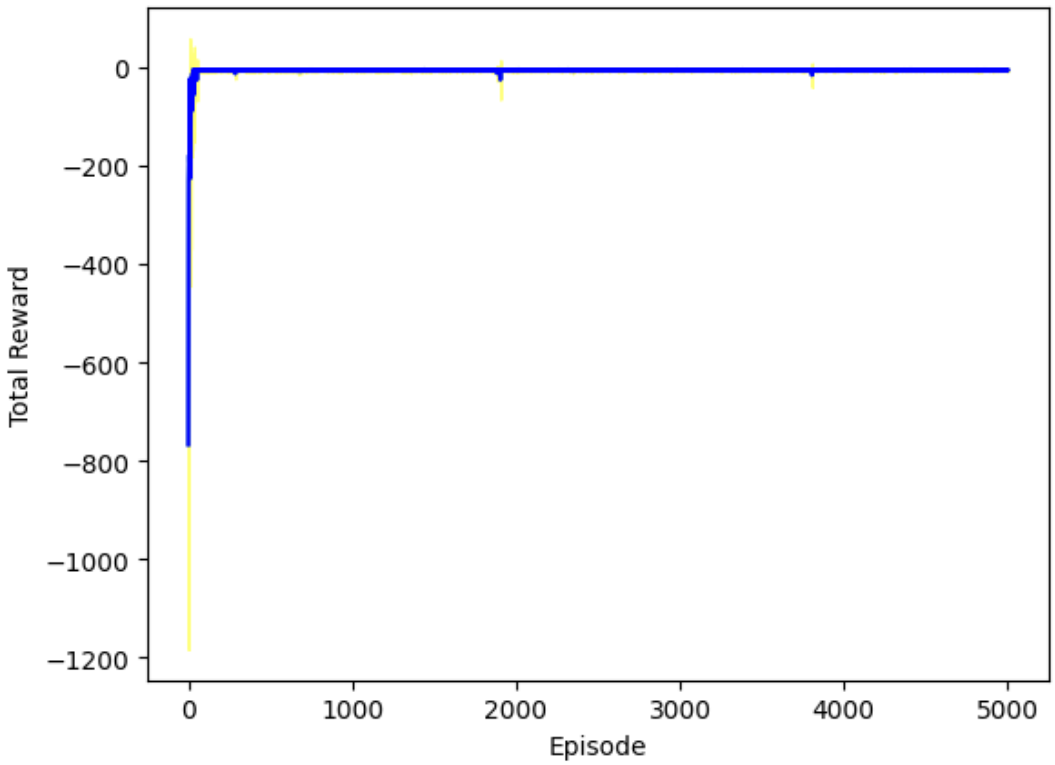
State Visit Count Plot



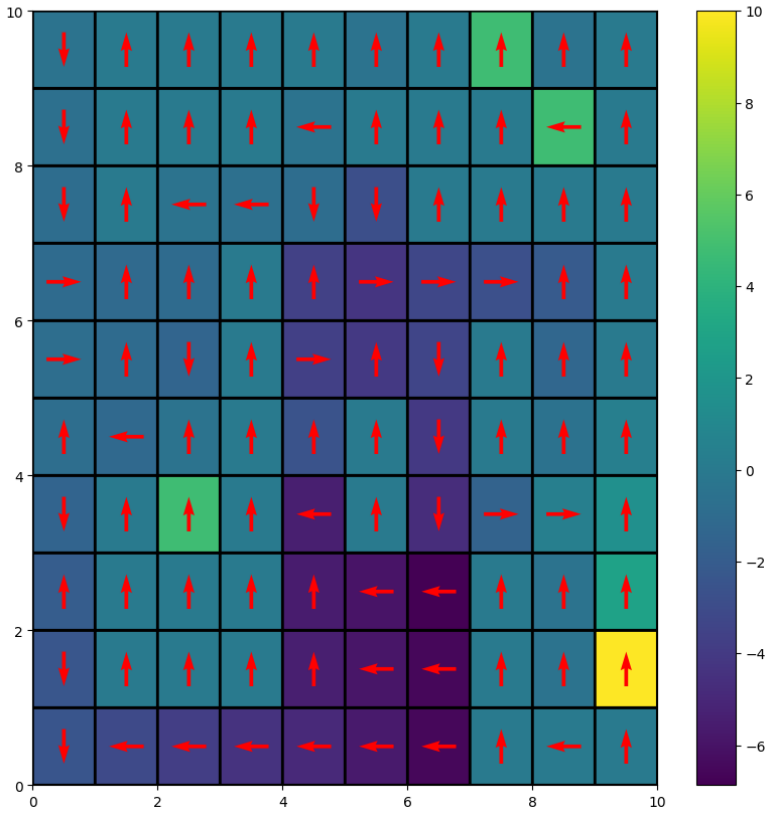
Plots for SARSA

Wind=True **P**=1 **Softmax-2** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.100

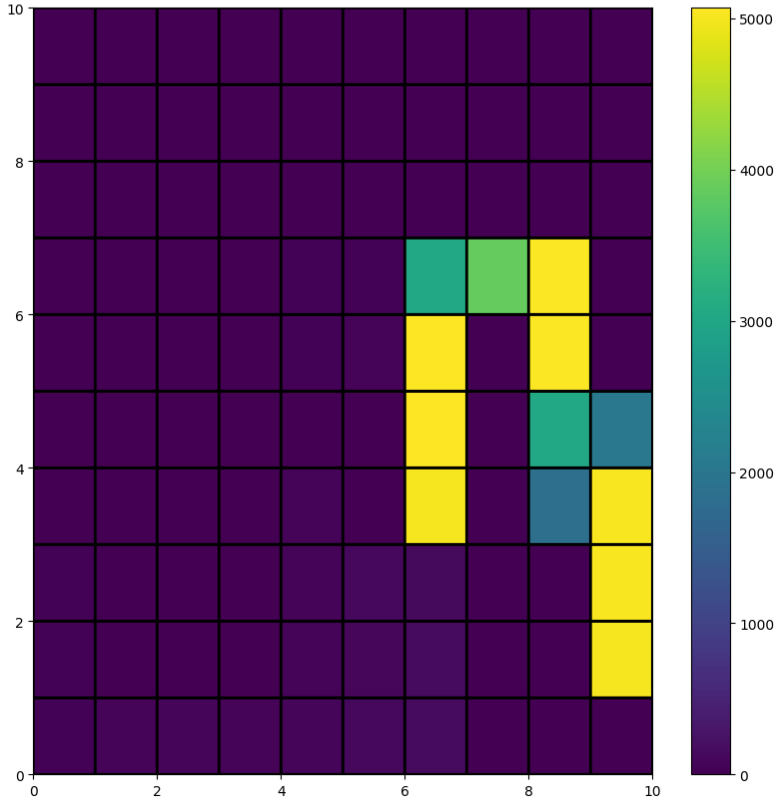




Episode 5000: Reward: -4.797980, Steps: 10.80, Qmax: 10.00, Qmin: -48.29

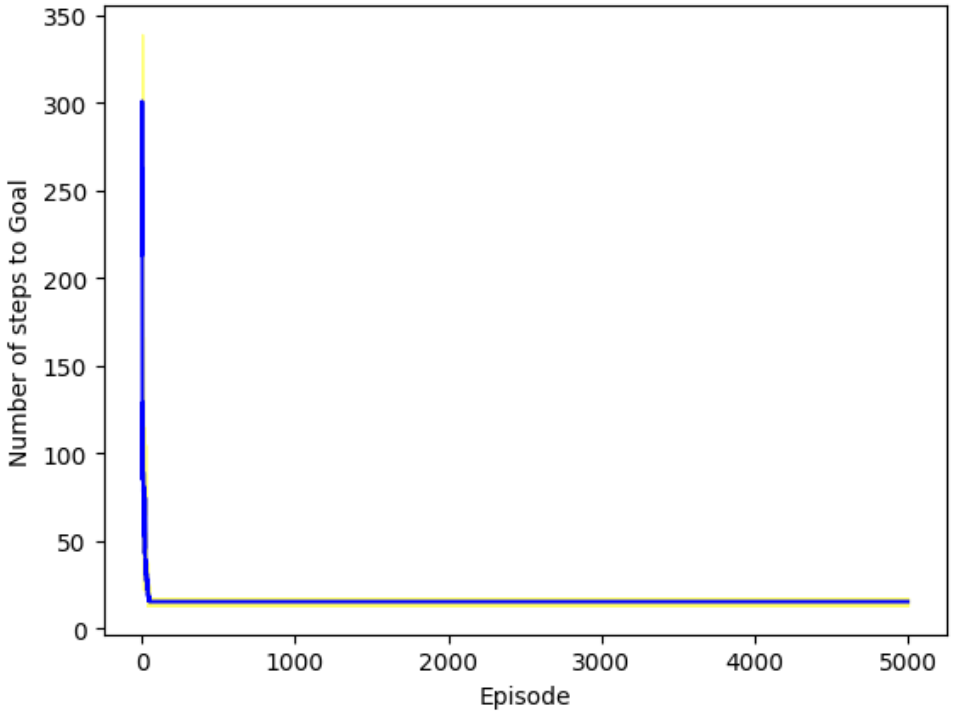


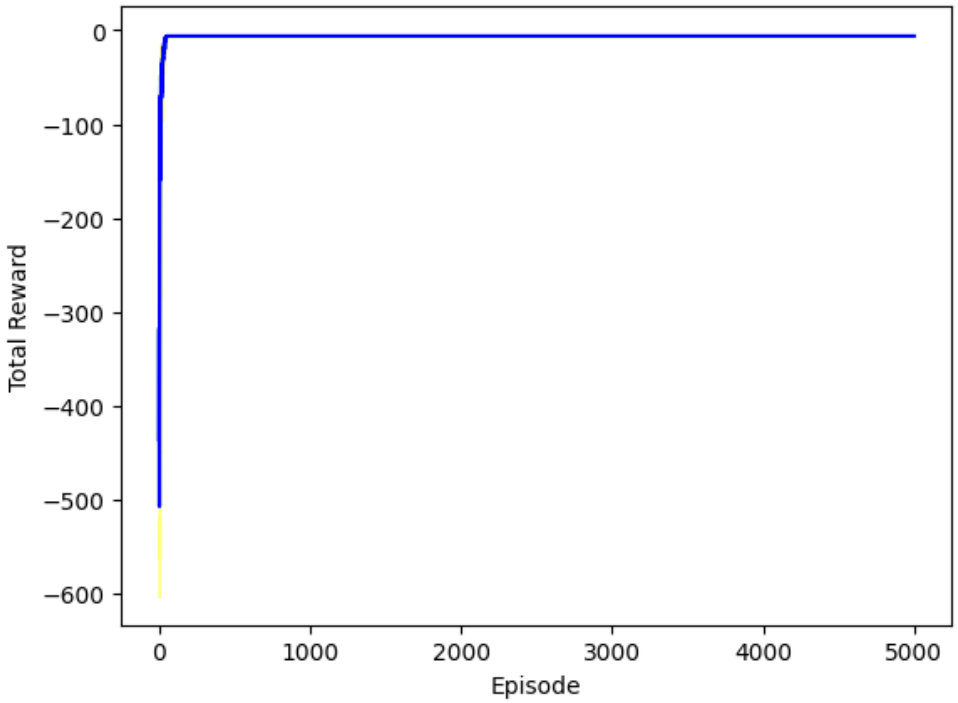
State Visit Count Plot



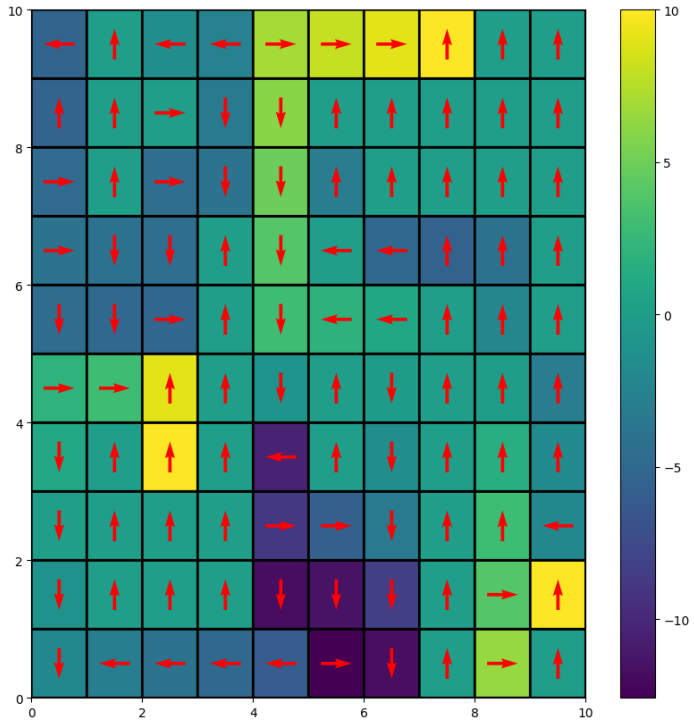
Plots for SARSA

Wind=False **P**=1 **Softmax-2** **Start State**=[0,4]
Alpha= 0.647 **Gamma**=0.999 **Tau**= 0.285

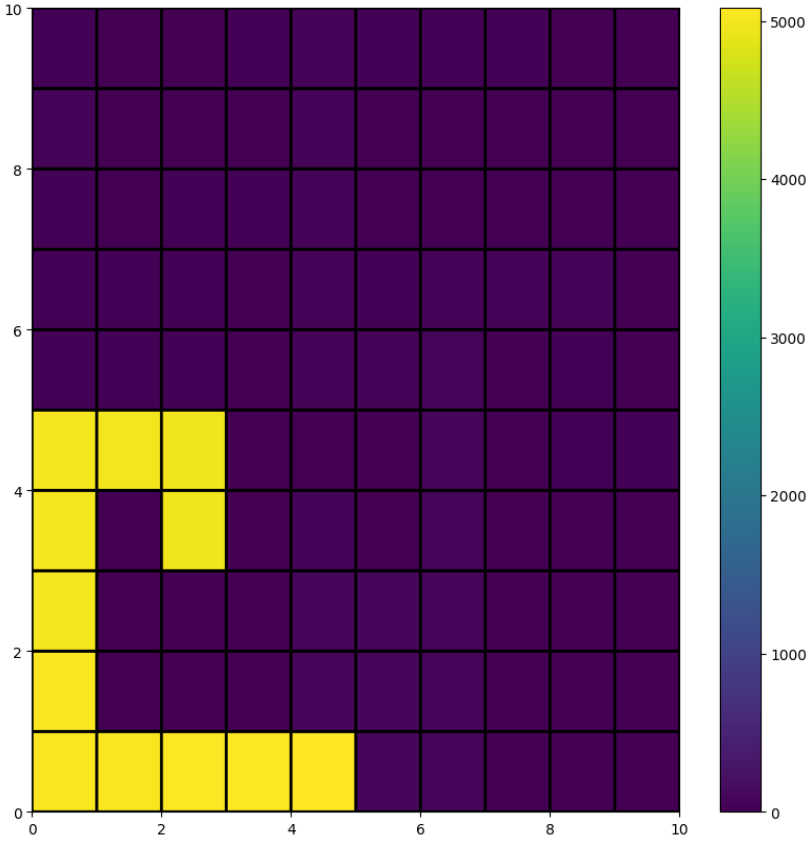




Episode 5000: Reward: -6.000000, Steps: 12.00, Qmax: 10.00, Qmin: -64.71

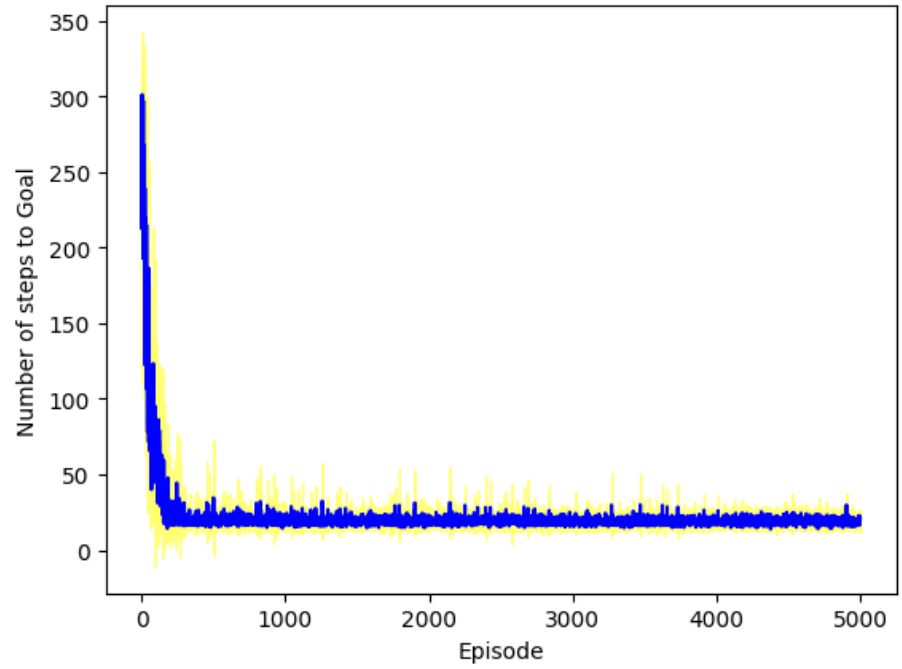


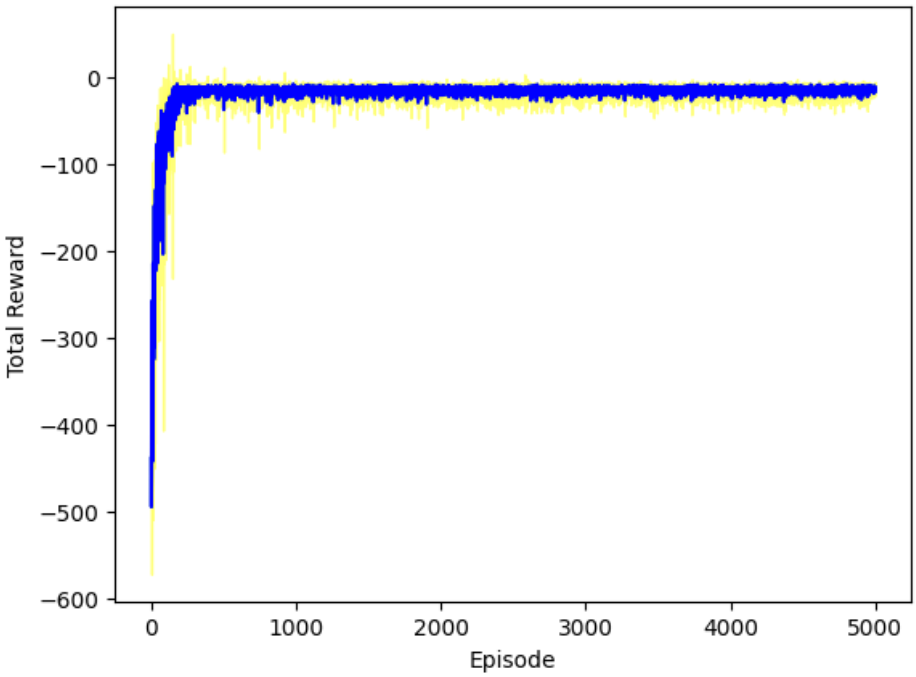
State Visit Count Plot



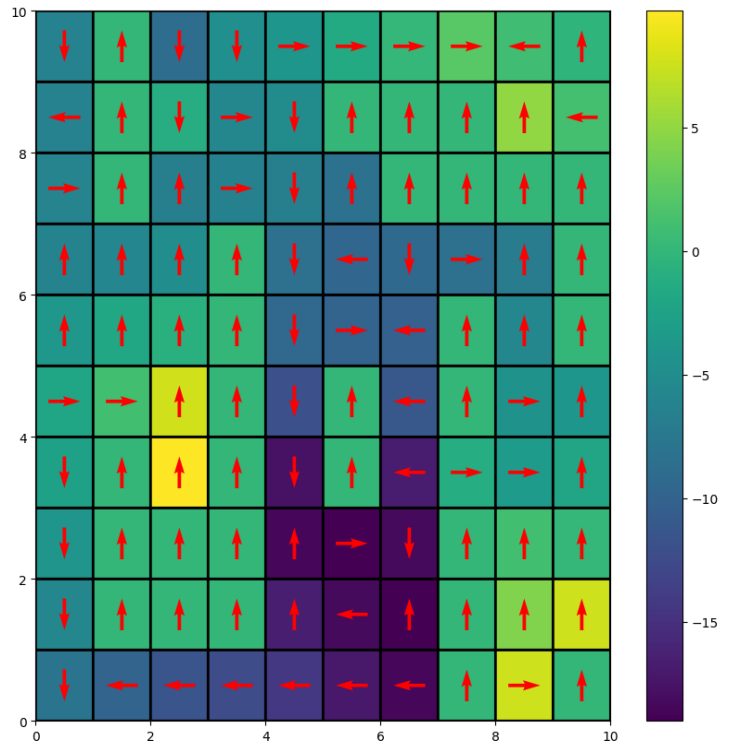
Plots for SARSA

Wind=False **P**=0.7 **Softmax-2** **Start State**=[0,4]
Alpha= 0.115 **Gamma**=0.999 **Tau**= 0.150

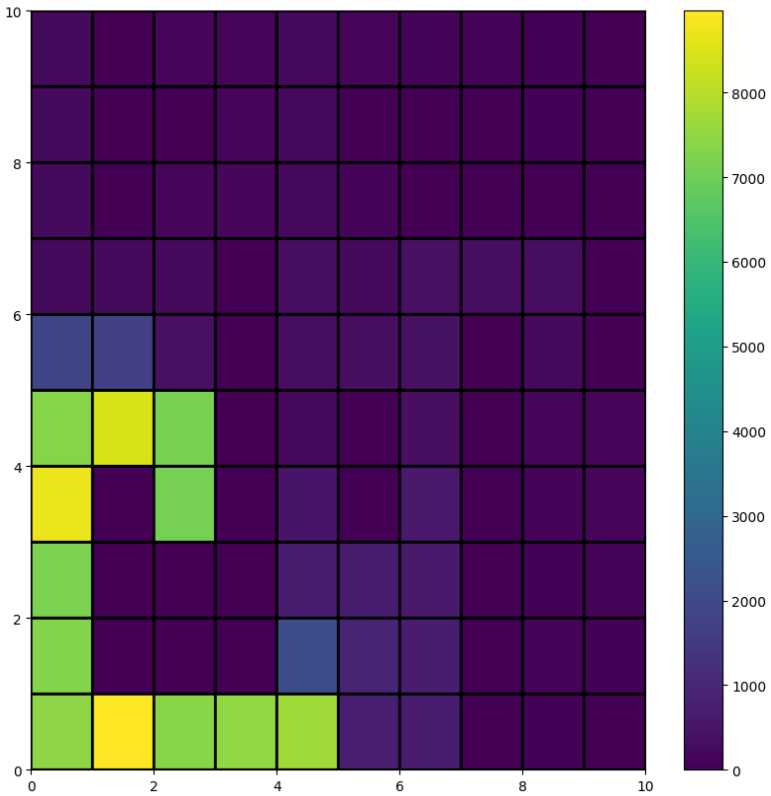




Episode 5000: Reward: -14.252525, Steps: 18.64, Qmax: 9.72, Qmin: -22.30

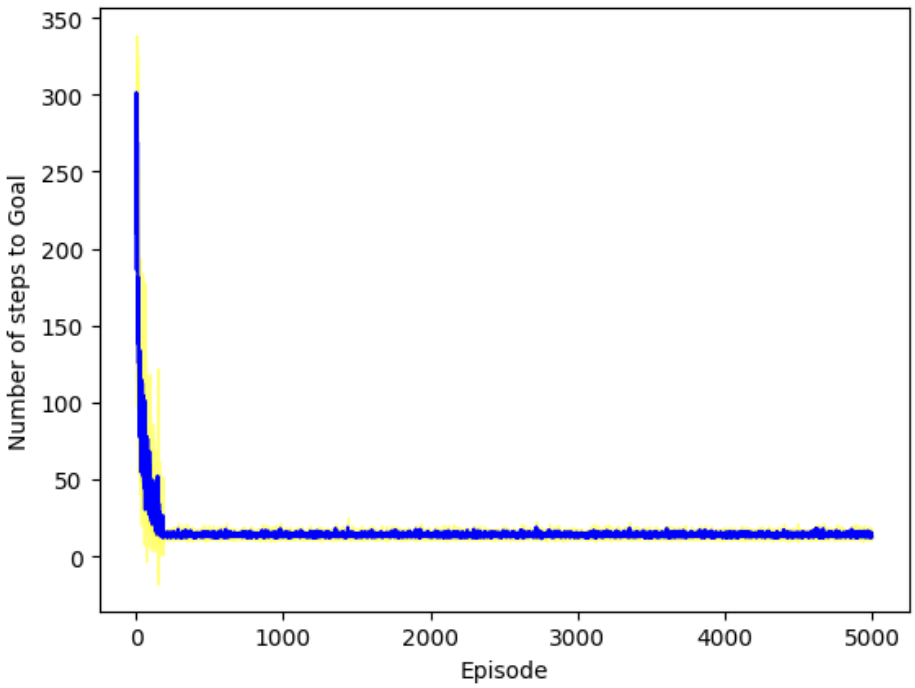


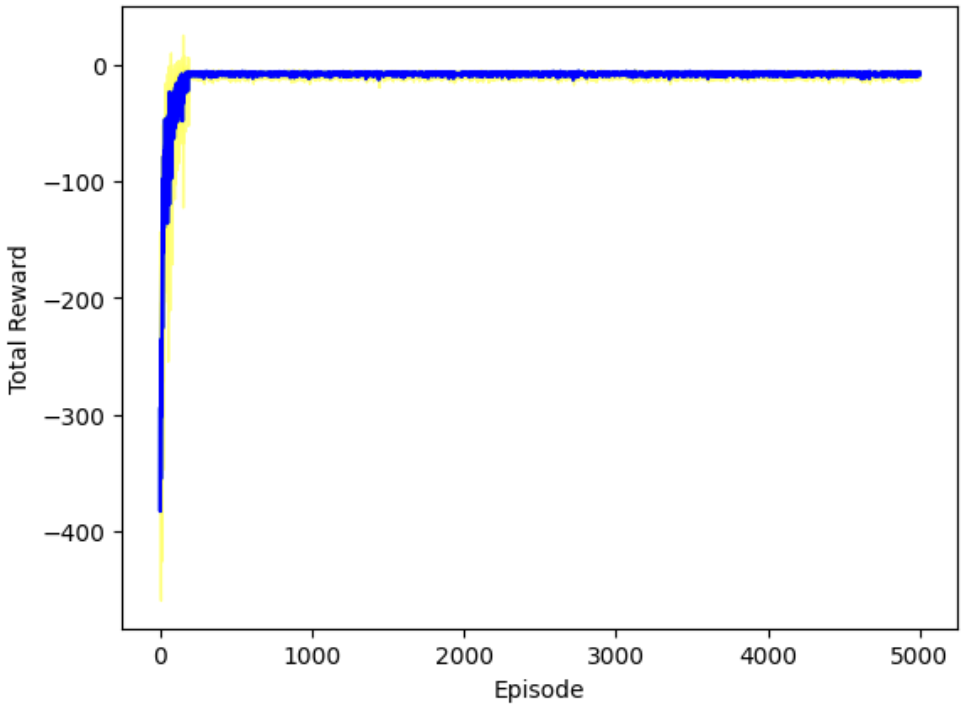
State Visit Count Plot



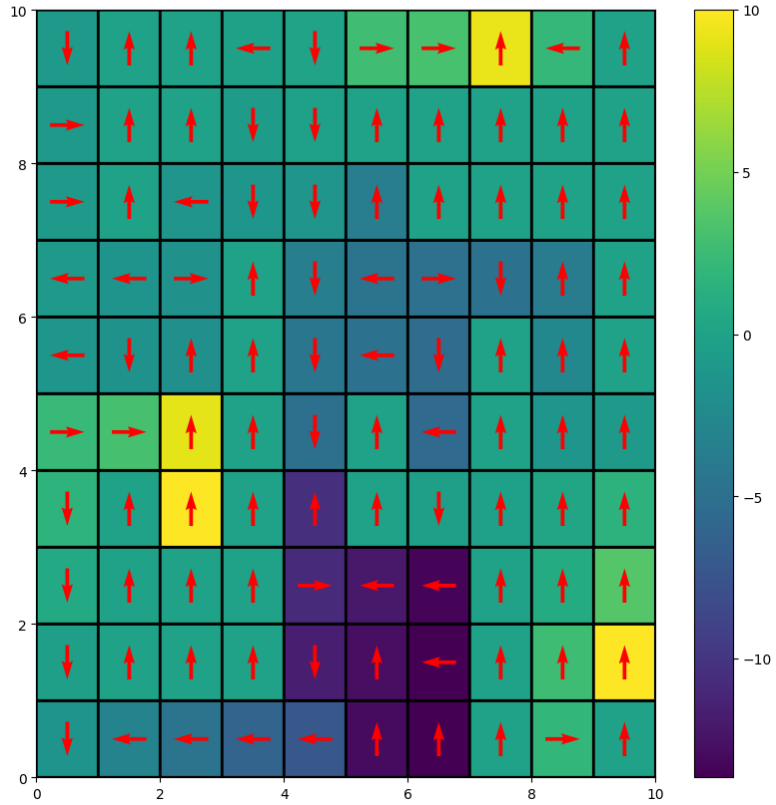
Plots for SARSA

Wind=True **P**=1 **Softmax-2** **Start State**=[0,4]
Alpha= 0.1 **Gamma**=0.999 **Tau**= 0.1

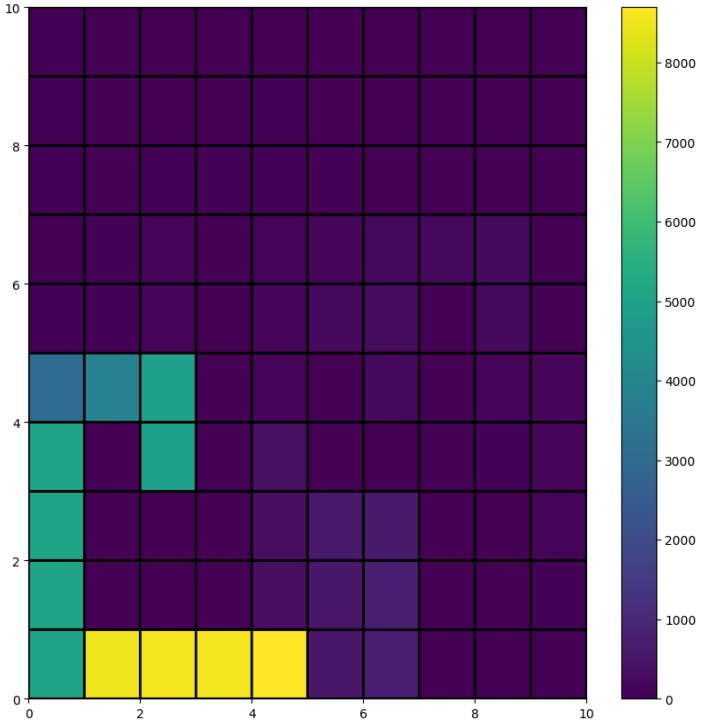




Episode 5000: Reward: -8.000000, Steps: 14.00, Qmax: 10.00, Qmin: -16.31

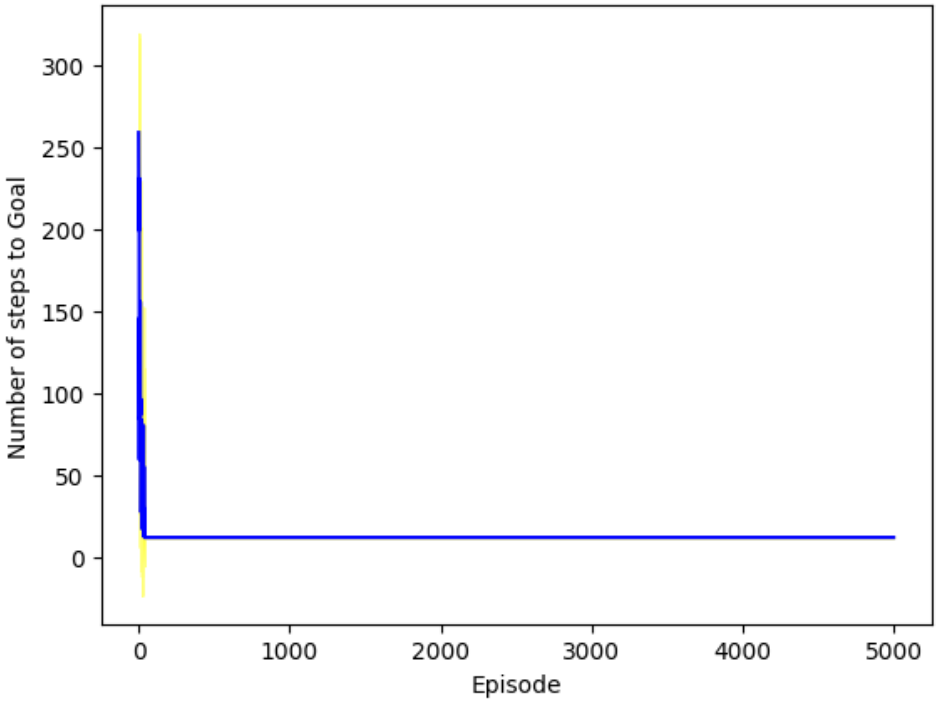


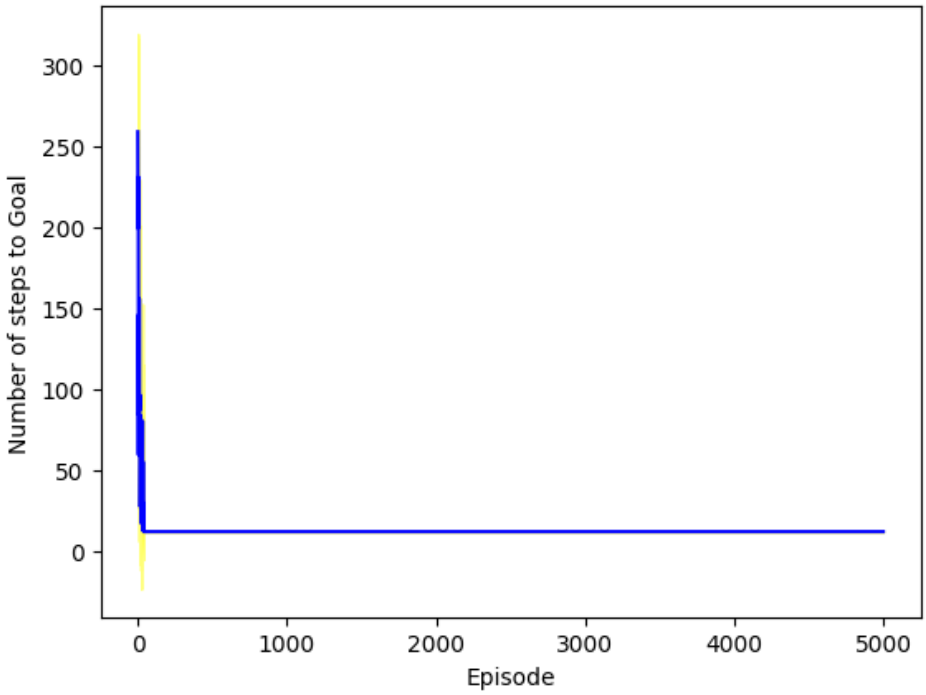
State Visit Count Plot



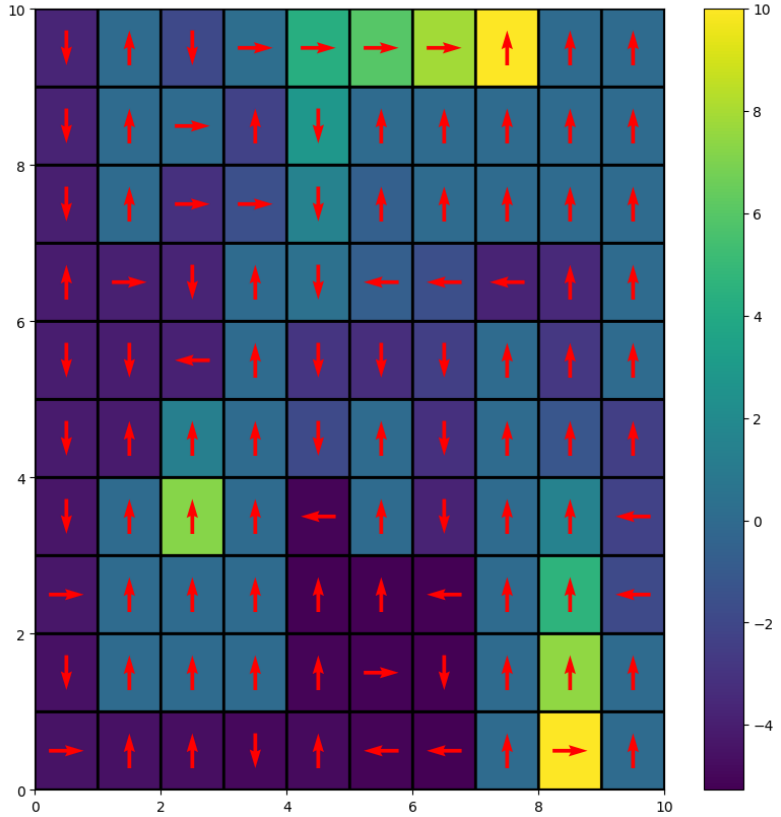
Plots for Q Learning

Wind=False **P**=1 **Softmax** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.1000

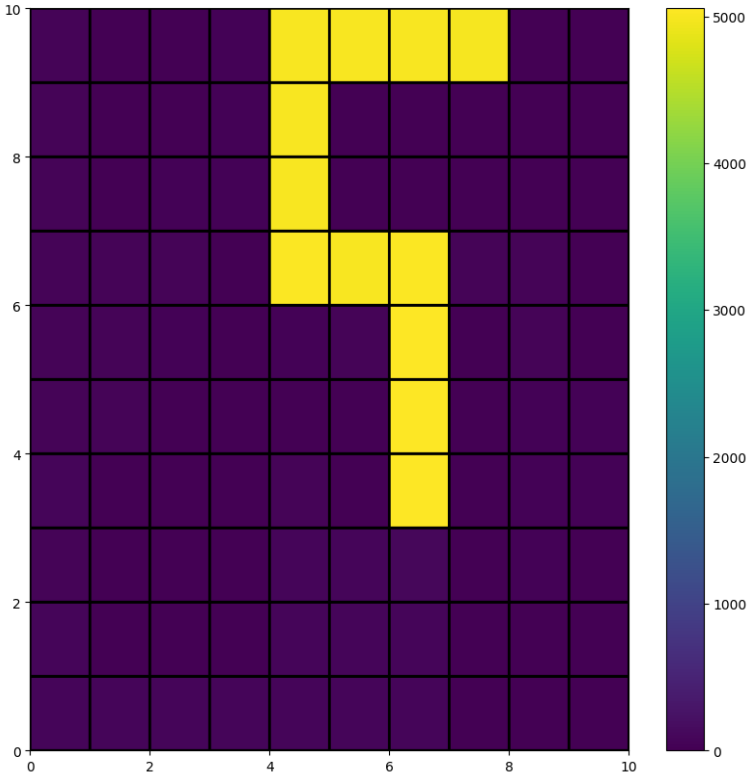




Episode 5000: Reward: -1.000000, Steps: 12.00, Qmax: 10.00, Qmin: -47.49

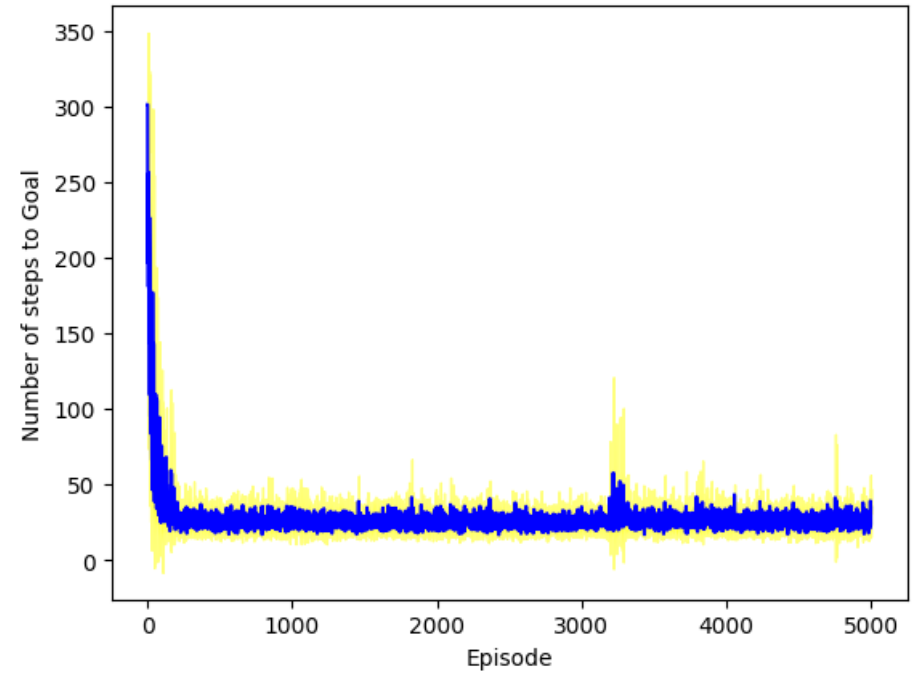


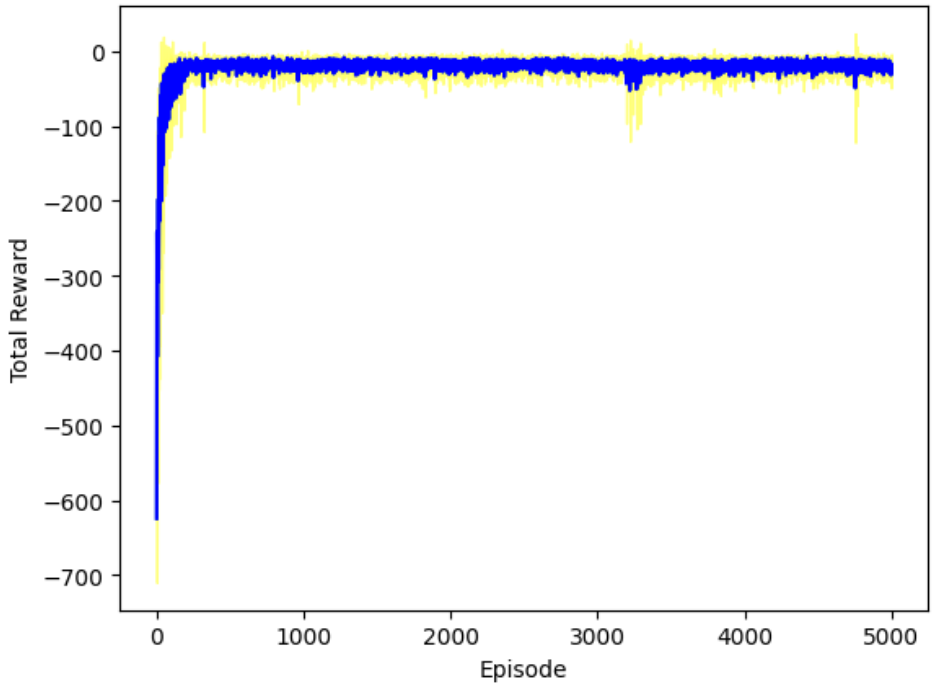
State Visit Count Plot



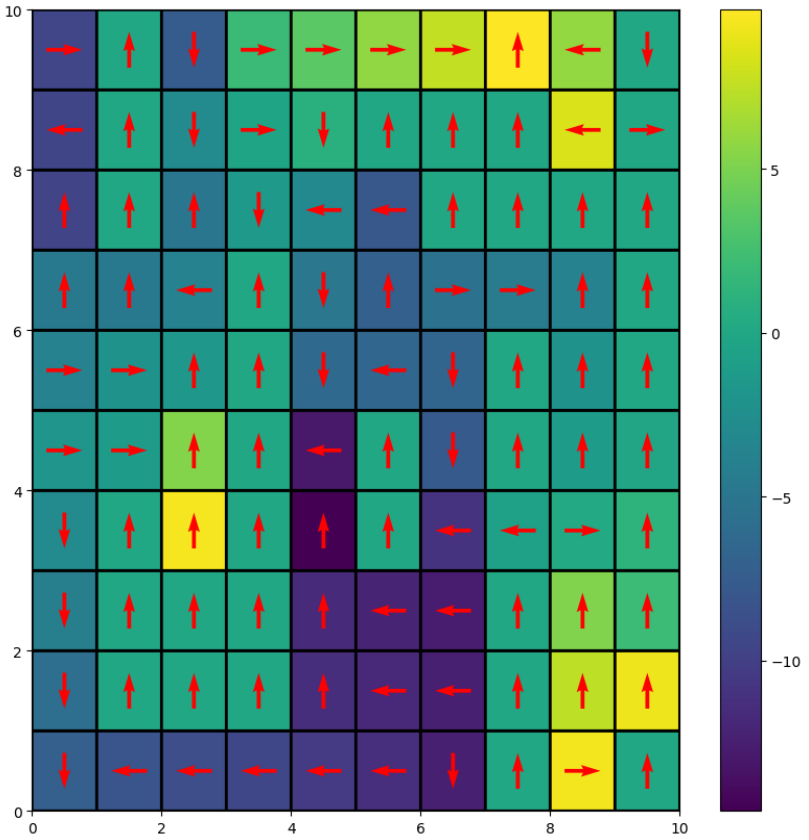
Plots for Q Learning

Wind=False **P**=0.7 **Softmax** **Start State**=[3,6]
Alpha= 0.283 **Gamma**=0.950 **Tau**= 0.234

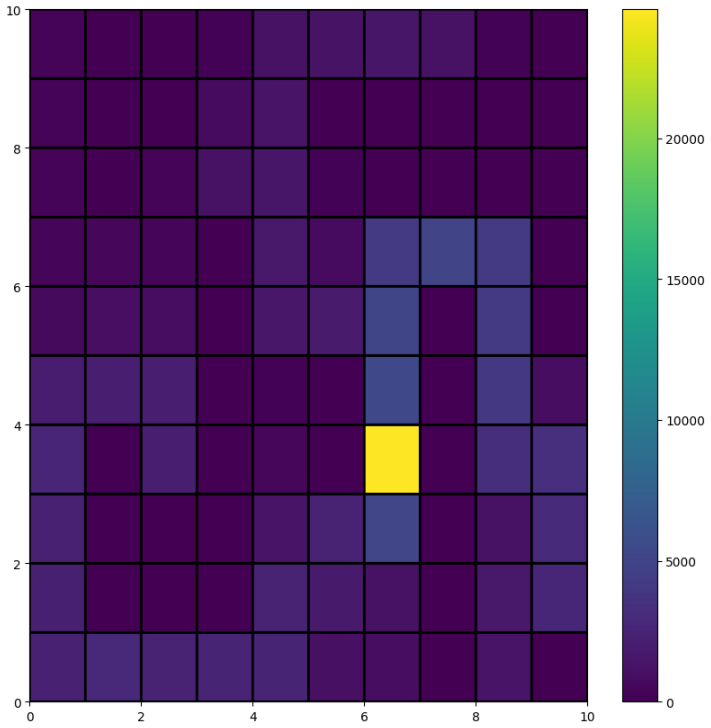




Episode 5000: Reward: -17.989899, Steps: 24.85, Qmax: 9.86, Qmin: -32.20

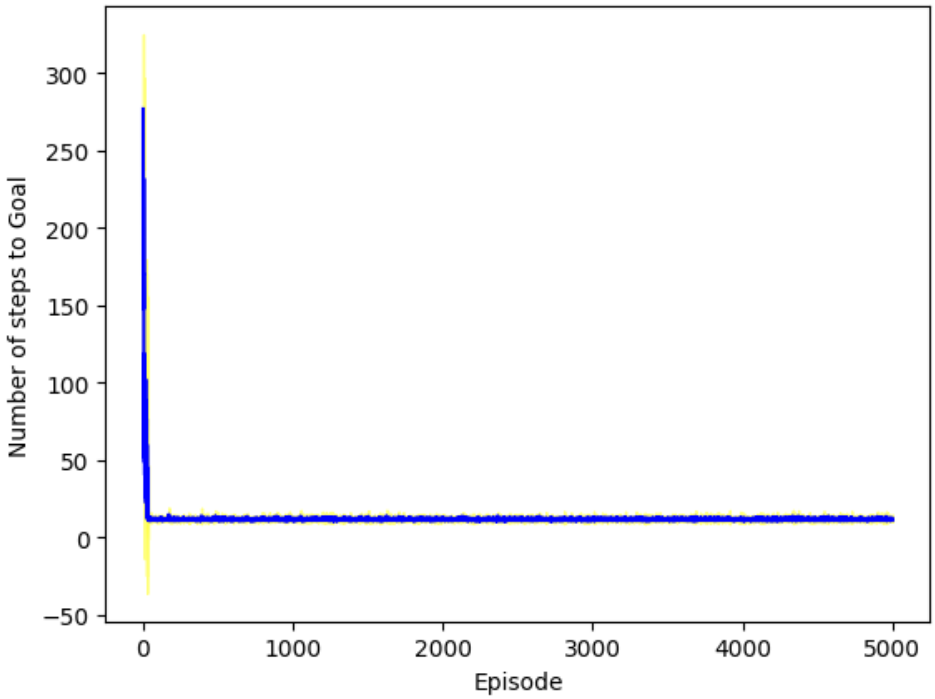


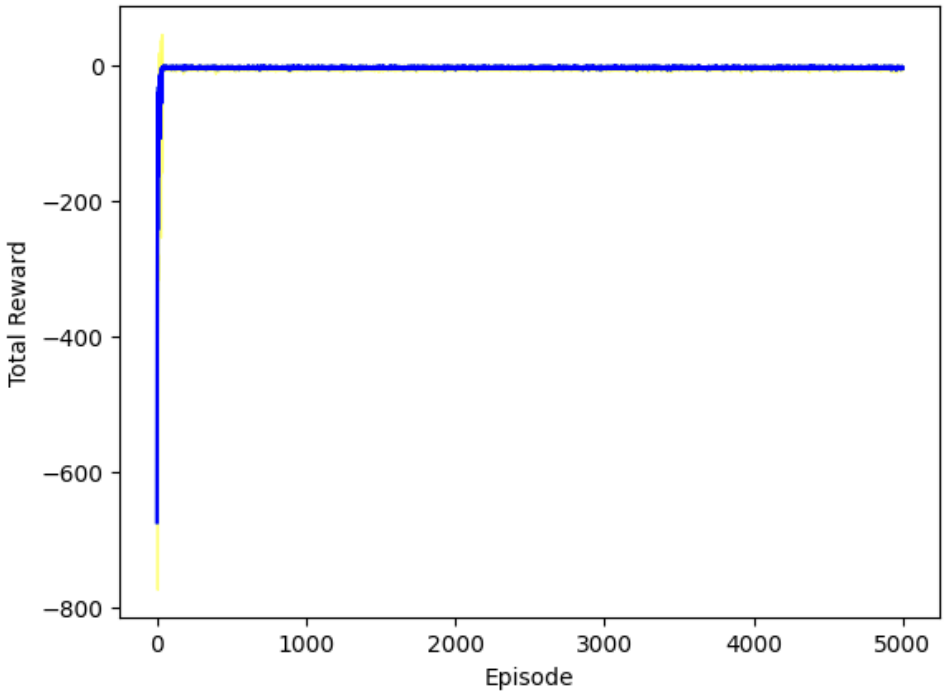
State Visit Count Plot



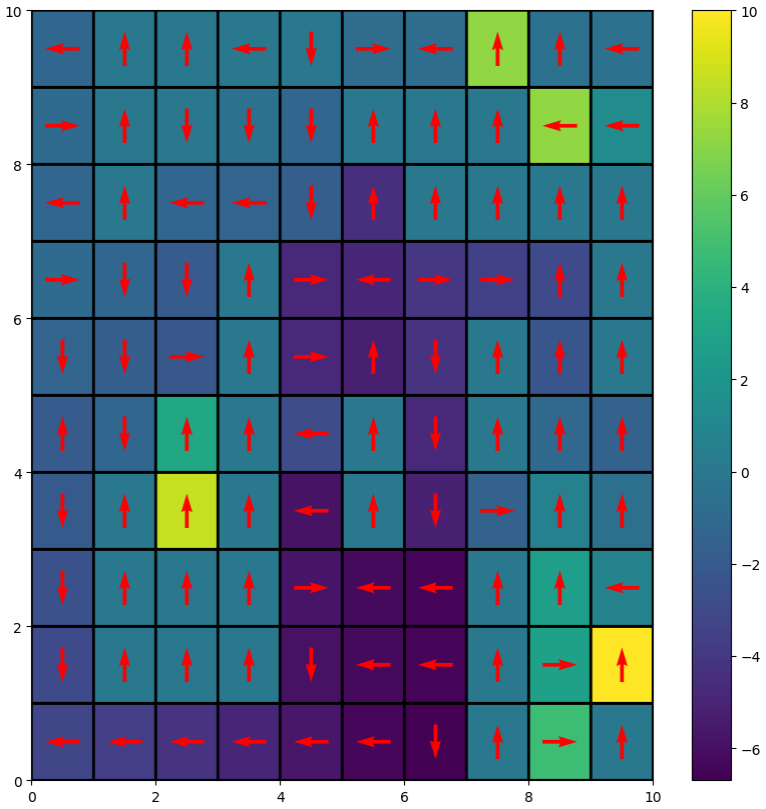
Plots for Q Learning

Wind=True **P**=1 **Softmax** **Start State**=[3,6]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.1000

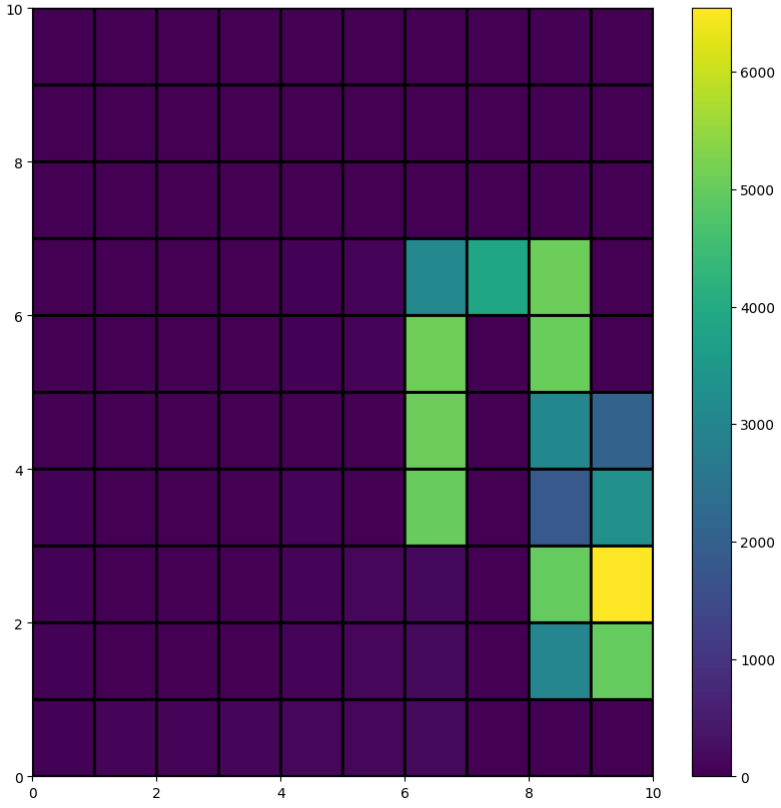




Episode 5000: Reward: -6.464646, Steps: 12.46, Qmax: 10.00, Qmin: -48.44

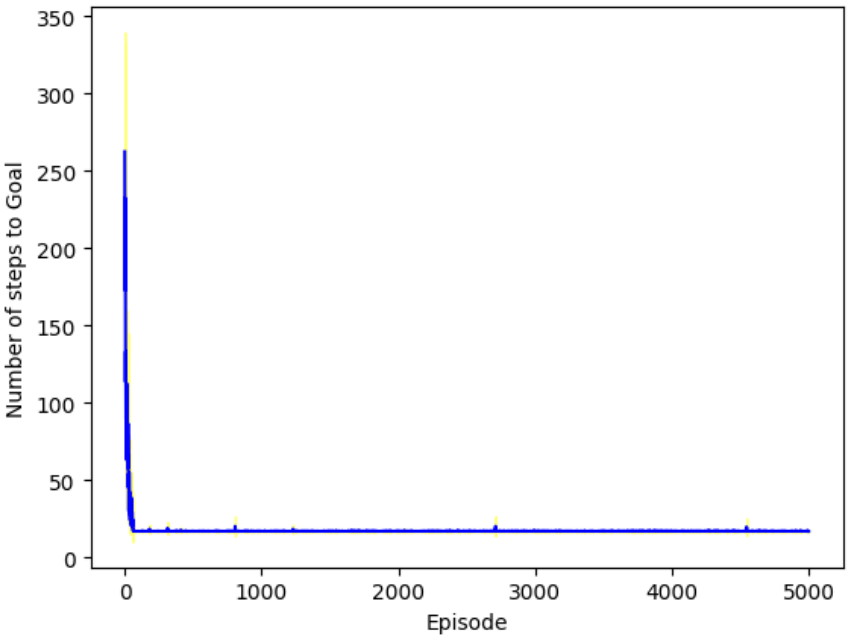


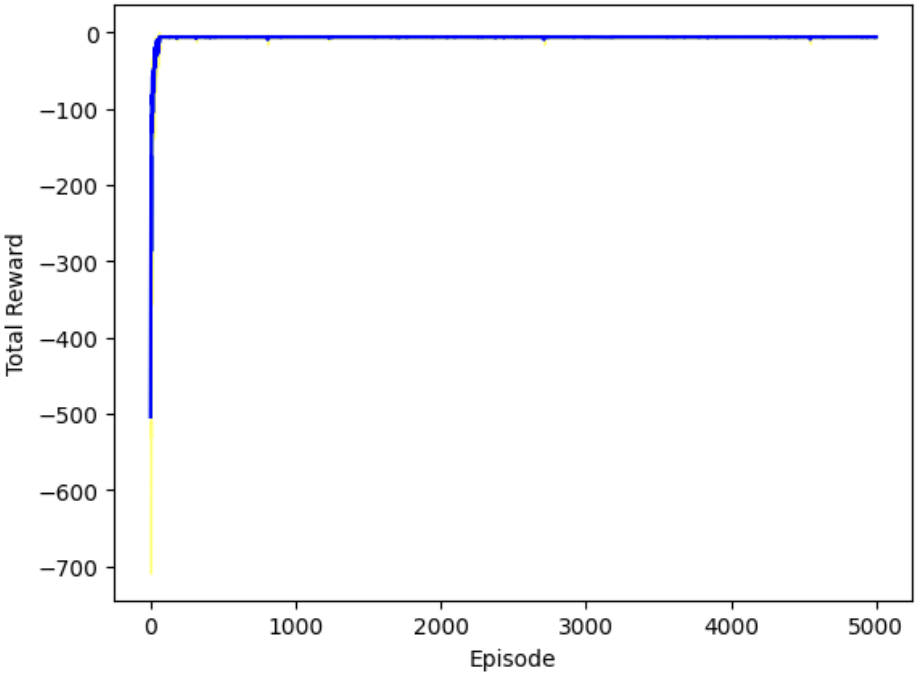
State Visit Count Plot



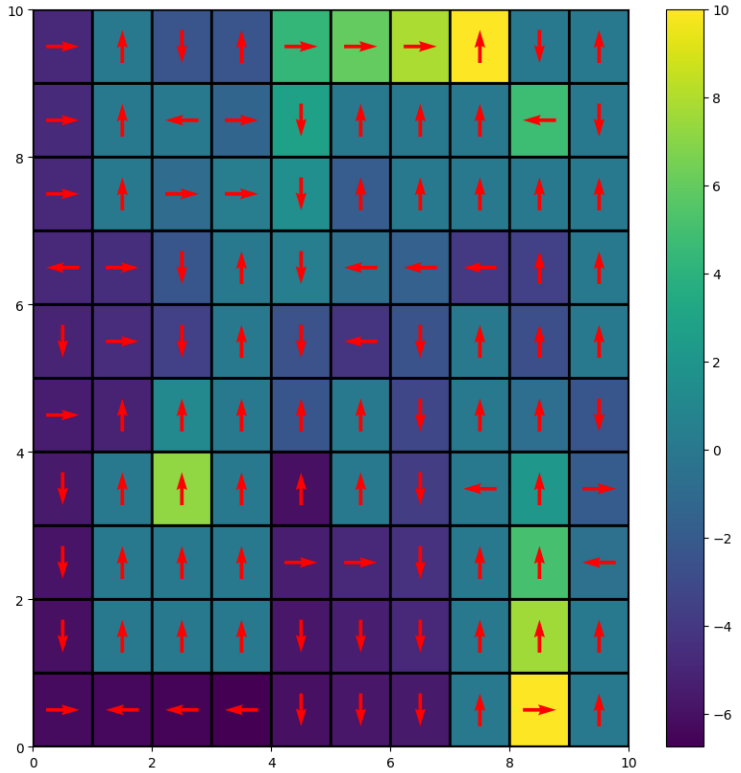
Plots for Q Learning

Wind=False **P**=1 **Softmax** **Start State**=[0,4]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.1005

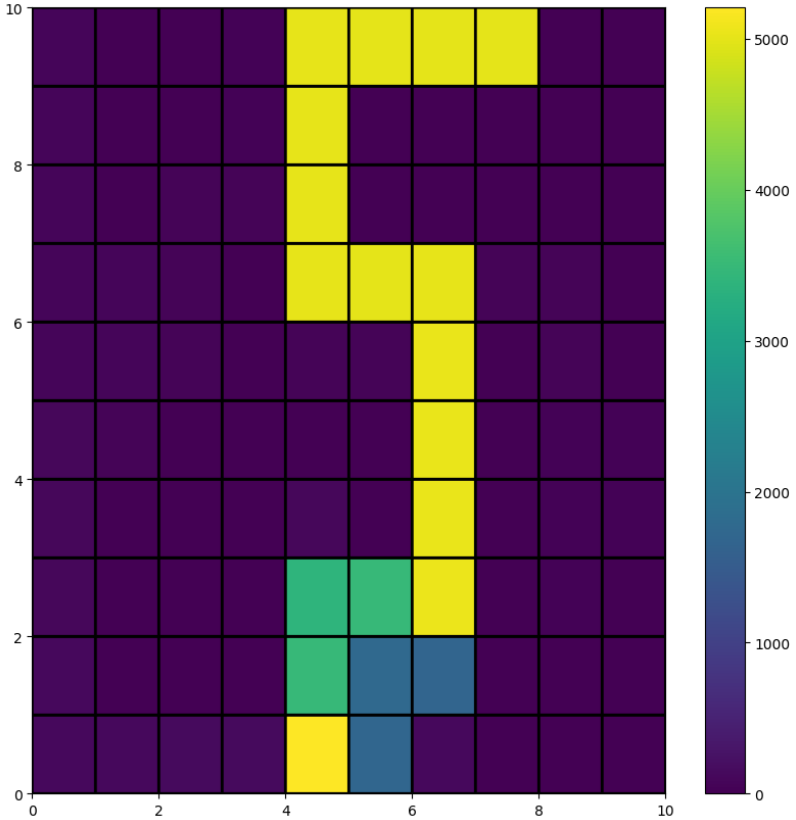




Episode 5000: Reward: -6.010101, Steps: 17.01, Qmax: 10.00, Qmin: -47.49

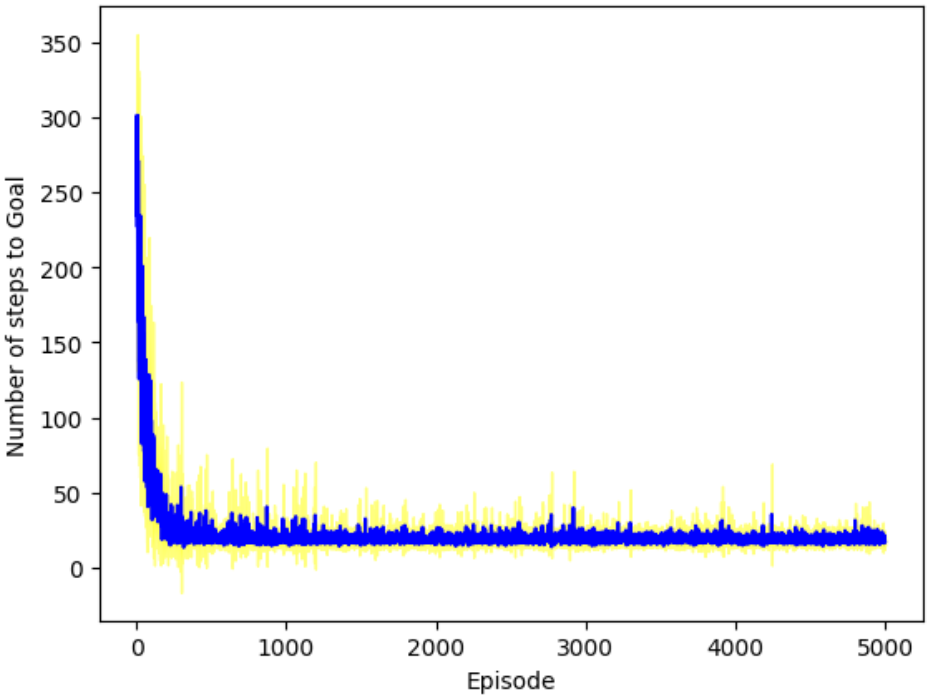


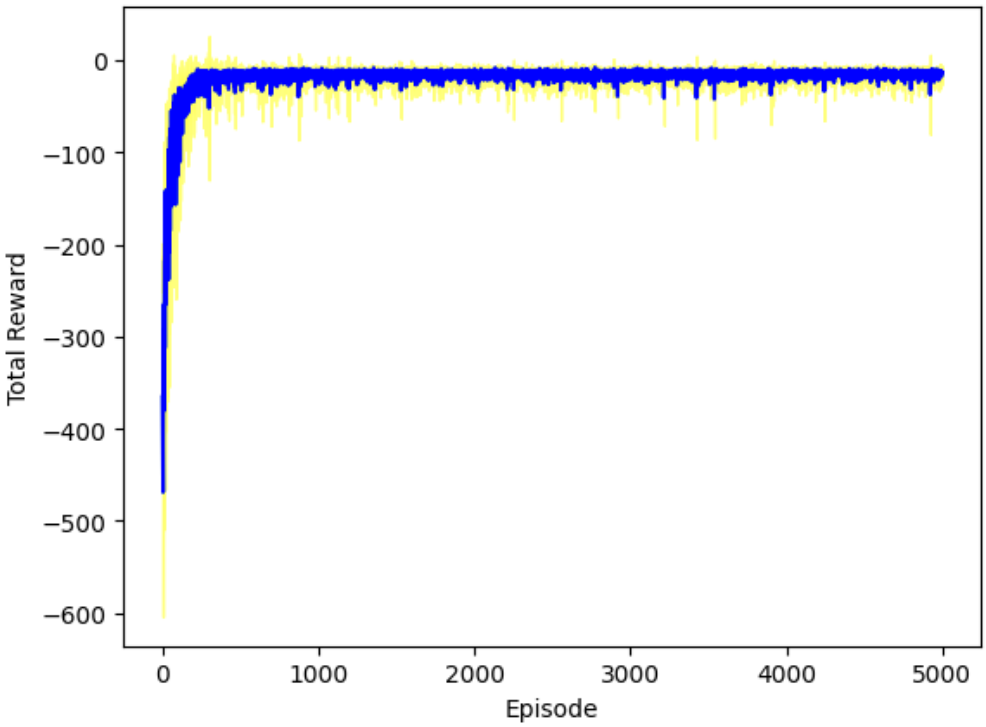
State Visit Count Plot



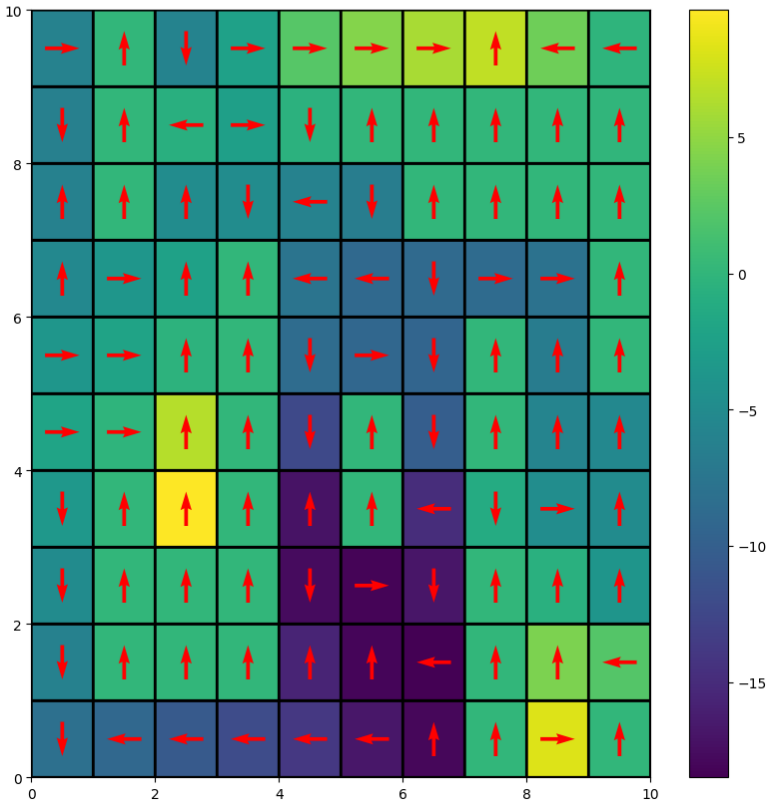
Plots for Q Learning

Wind=False **P**=1 **Softmax** **Start State**=[0,4]
Alpha= 0.1 **Gamma**=0.999 **Tau**= 0.1

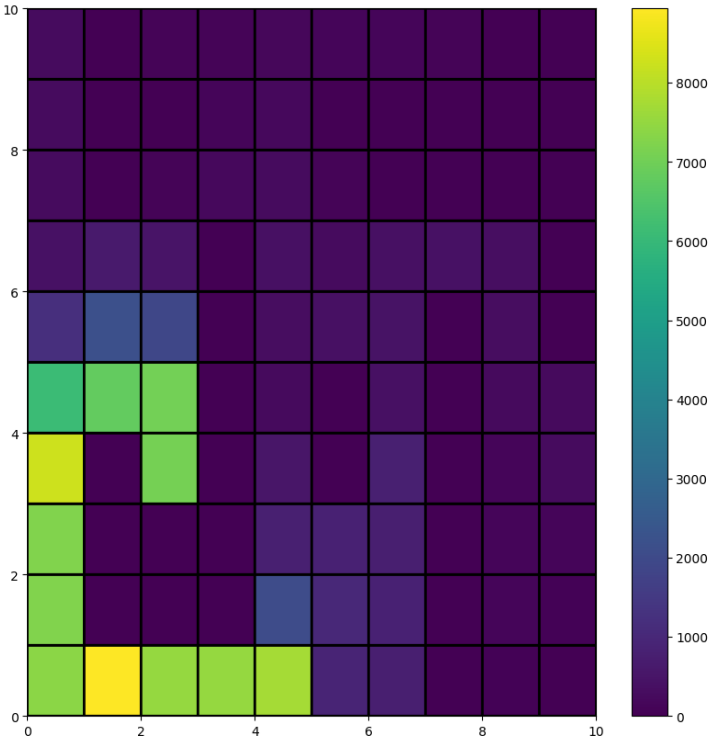




Episode 5000: Reward: -14.242424, Steps: 18.32, Qmax: 9.67, Qmin: -23.83

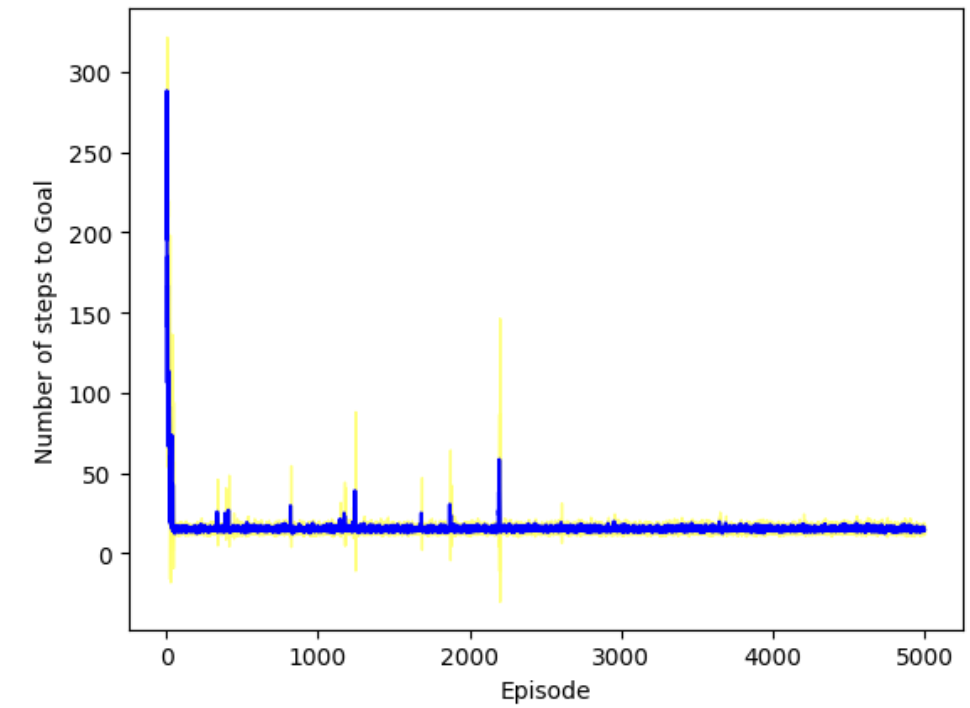


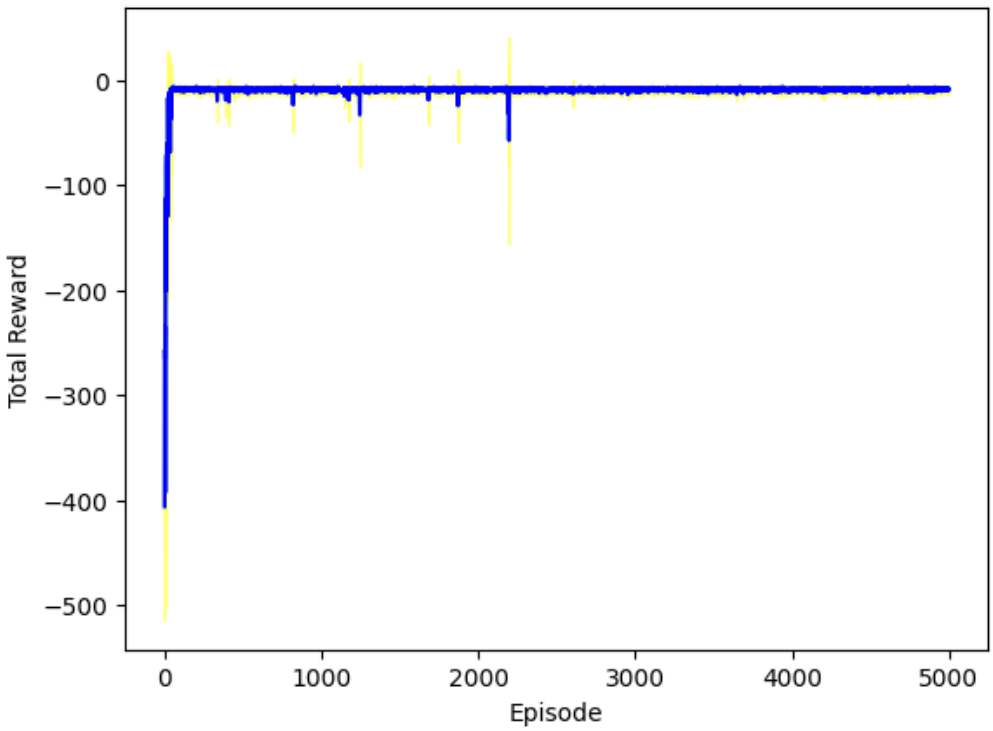
State Visit Count Plot



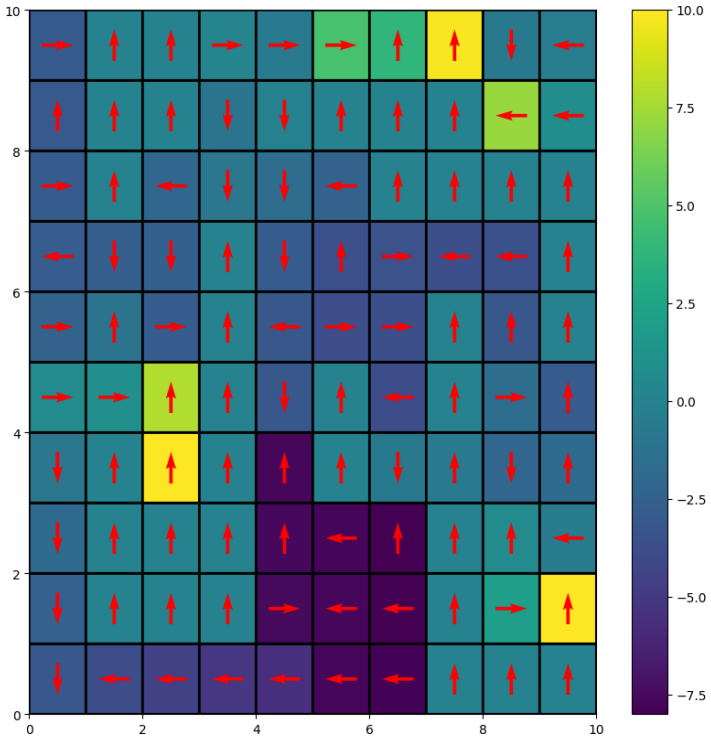
Plots for Q Learning

Wind=False **P**=1 **Softmax** **Start State**=[0,4]
Alpha= 0.474 **Gamma**=0.887 **Tau**= 0.100

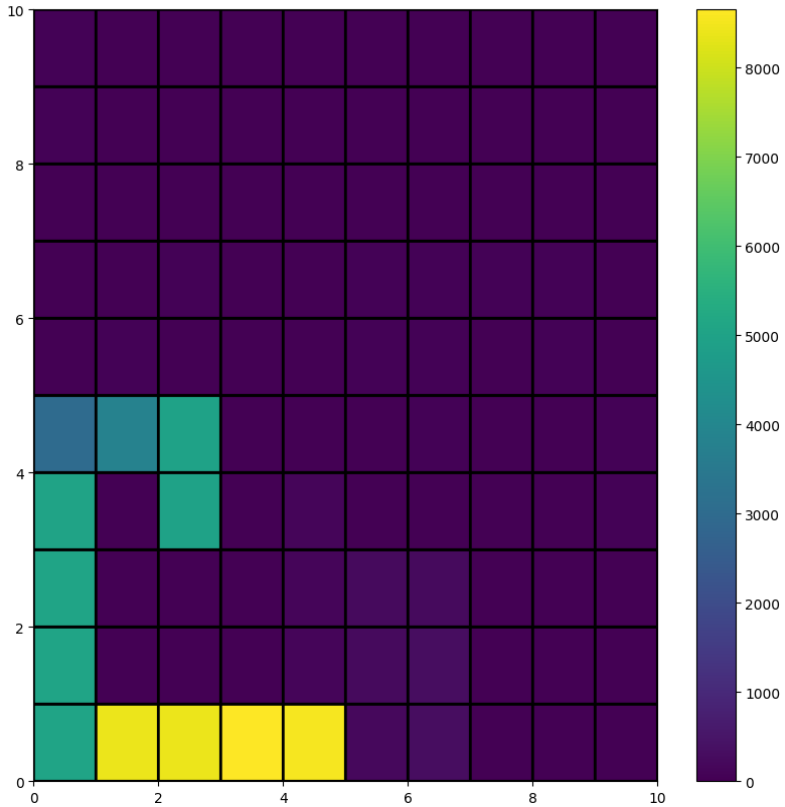




Episode 5000: Reward: -8.252525, Steps: 14.25, Qmax: 10.00, Qmin: -48.19



State Visit Count Plot



Inferences:

1) Optimal alpha values for P=0.7(stochastic) are lesser than that for deterministic cases. Decreasing alpha can help reduce variability and unpredictability in the stochastic environment by considering the expected value of future rewards over multiple possible outcomes. A large value of alpha may cause erratic changes to the value functions.

2) Q-learning performs better for the start state [3,6] than SARSA. This is because there are alot of bad states around [3,6]. SARSA is conservative and goes for the safer path, whereas Q-learning still searches for an optimal path.

3) In a deterministic and wind=False state, where stochasticity is low, the gamma value for Q-learning is lesser than SARSA. A higher gamma means that more weightage is given to future Q values. So, since SARSA is the more patient and conservative agent, it looks for cumulative rewards in the distant future over focusing on immediate rewards.

4) The gamma value for a stochastic environment is higher when compared to other environments. Higher gamma means that higher weightage is given to future Q-values. Long term expected values are preferred due to the stochastic nature of the environment.

5) In most cases, epsilon value is as low as 0.0001, which implies that we don't need too much exploration. Since we experiment over 5000 episodes, the policy has enough time to converge regardless of the epsilon value. Hence, we conclude that the optimal reward value can be achieved by epsilon = 0.0001.

6) As seen in the first iteration of softmax, we saw that larger tau values lead to a lot of higher negative rewards. This means that the policy does not converge to the optimal action values within 5000 episodes.Hence, we did a second iteration with a more greedy approach and less exploration. This second iteration resulted in better reward values and better tuned parameters

7) From the plots, it is evident that the epsilon greedy policy approaches closer to the optimal value faster and steeper than the softmax policy.

8) Because the wind blew consistently in the desired direction, the agent tended to move more frequently in that direction, ultimately discovering the goal in that same direction. This is clear from the state visit plot. This pattern of movement created a biased exploration, leading to increased unpredictability on one side. The state visit plot reflected this bias, showing a greater number of visits on the right-hand side due to the influence of the wind.
9) As the stochasticity increases, the average reward decreases.

Conclusion:

1) Hence, we have implemented SARSA and Q-learning algorithms for different variants of the environment and utilizing both epsilon greedy and softmax strategies for exploration.

2) We learned the difficulties and the computational costs involved in hyperparameter tuning.

3) We visualized the training results of each of these algorithms using plots and performed an analysis to compare the hyperparameters across different environments.