Assignment 1 - Reinforcement Learning - CS6700

1st Royyuru Sai Prasanna Gangadhar
Mechanical Engineering
IIT Madras
Chennai, India
me19b190@smail.iitm.ac.in

2nd Abhigyan Chattopadhyay
Electrical Engineering
IIT Madras
Chennai, India
ee19b146@smail.iitm.ac.in

***Abstract*—In this assignment, we explore SARSA and Q-Learning as methods to find the best possible path for a Grid World problem with given start & end states, with blockages and bad states in the way.**

The aim is to calculate the fastest path in the shortest time possible, with a variety of conditions that will change the experiment, the aim is to create a model that is robust to adapt to these different conditions and still give a good result.

***Index Terms*—reinforcement learning, q learning, sarsa, hyperparameter tuning**

I. Introduction

A. Temporal Difference Learning Algorithms

Temporal Difference (TD) learning is a type of machine learning algorithm that is commonly used in reinforcement learning tasks, where an agent learns to interact with an environment by taking actions that maximize a reward signal.

TD learning is a model-free approach, meaning that it does not require a complete model of the environment and instead learns directly from experience.

TD learning updates its estimates of the value function by using the current estimate as a starting point, enabling it to learn in an incremental and efficient manner. By iteratively updating its value function estimates based on the observed rewards and transitions between states, TD learning can converge to an optimal policy that maximizes the expected future reward for the agent, which is mathematically provable to be the most optimal policy.

B. Greedy vs Dynamic Programming

There are two main approaches that TD learning can take: greedy and dynamic programming, or a hybrid of the two.

In the greedy approach, the TD learner always chooses the action that is expected to yield the highest immediate reward, based on the current value function estimate. The greedy approach is computationally efficient, but may result in the learner getting stuck in a suboptimal policy.

In contrast, the dynamic programming approach considers all possible actions at each state and updates the value function estimates based on the expected return of each action. This approach is more computationally expensive but can lead to a better policy in the long run.

C. SARSA

Another approach is SARSA (State-Action-Reward-State-Action). It uses a value function to estimate the expected future reward of taking a particular action in a given state, and updates its estimates based on observed transitions from one state-action pair to another.

SARSA is not a purely greedy approach, as it uses either an epsilon-greedy policy or softmax policy to balance exploration and exploitation during the learning process.

D. Q-Learning

TD learning can also use a combination of both approaches, known as "Q-learning". In Q-learning, the learner can use a greedy strategy (like epsilon-greedy or softmax) to select actions during the learning process, but updates its value function estimates based on the maximum expected return across all possible actions. This approach balances computational efficiency with long-term performance.

E. Exploitation vs Exploration

SARSA and Q-Learning's behaviors are not always purely greedy, as they sometimes choose actions that are not optimal according to their current value function estimate. They can be seen as a compromise between pure greedy and dynamic exploration strategies, balancing short-term and long-term performance.

F. Policies for Exploration vs Exploitation

1) Epsilon-Greedy: The epsilon-greedy policy means that the agent takes a random action with probability epsilon (to explore the environment and potentially discover better policies), and takes the action with the highest expected reward with probability 1-epsilon (to exploit the current knowledge and maximize rewards).

2) Softmax: The softmax update policy incorporates a stochastic decision-making process based on the softmax distribution. The softmax distribution allows for a more flexible decision-making process that can explore different actions, even if they are not currently the most promising, and allows for more exploration rather than exploitation. It has significantly different results compared to the epsilon-greedy policy.

G. SARSA Update Policy

The SARSA update policy for the value function is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{1}$$

H. Q-Learning Update Policy

The Q-learning update policy for the value function is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{2}$$

II. Environment Description

A. The Grid-World Problem

III. Training

After understanding, the mathematical concepts of SARSA and Q-learning, it is time to implement it in python.

A. Implementation of SARSA to learn optimal policy

The following are the steps followed in the implementation of SARSA.

- Setup the environment: Environment has been configured with initial states, reward for multiple states (normal states, restart state, bad state and goal state), stochasticity is added to the environment with wind blowing and probability of good transition.
- The default number of epochs has been set to 1000. Because there is a stochasticity in the environment, it is important that the action value function has to been averaged over multiple experiments to get an unbiased estimate of the action value function. Default number of experiments have been set to 3.
- The agent is allowed to explore the given environment either by an epsilon-greedy policy or softmax
- The current action value function has been updated using the action value of the next immediate state, where the actions in present and following state are selected in an epsilon-greedy fashion. An episode terminates if the number of steps taken exceeds 100 or when the agent reaches the goal state. An experiment is terminated when number of episodes reach 1000.
- The following parameters cumulative reward, number of times a particular state is visited, and number of times a state is visited are being updated for each action taken by the agent
- The above five steps have been repeated $num_expts = 3$ and all the estimates are averaged over the three experiments to get an unbiased estimate of the optimal action value function.

B. Q- learning to learn the optimal policy

The following are the steps followed in the implementation of Q-learning.

- Setup the environment: Environment has been configured with initial states, reward for multiple states (normal states, restart state, bad state and goal state), stochasticity is added to the environment with wind blowing and probability of good transition.
- The default number of epochs has been set to 1000. Because there is a stochasticity in the environment, it is important that the action value function has to been averaged over multiple experiments to get an unbiased estimate of the action value function. Default number of experiments have been set to 3.
- The agent is allowed to explore the given environment either by an epsilon-greedy policy or softmax
- The current action value function has been updated using the maximum value of the current action value function of the next immediate state. Therefore, Q-learning is greedy with respect to the current estimate of the action value function. An episode terminates if the number of steps taken exceeds 100 or when the agent reaches the goal state. An experiment is terminated when number of episodes reach 1000.
- The following parameters cumulative reward, number of times a particular state is visited, and number of times a state is visited are being updated for each action taken by the agent
- The above five steps have been repeated $num_expts = 3$ and all the estimates are averaged over the three experiments to get an unbiased estimate of the optimal action value function.

IV. Hyper parameter Tuning

In theory, both SARSA and Q learning will converge to the true optimal action value function when learning rate is sufficiently small and the number of episodes and experiments tending to infinity. It is impractical to either have a very small α or run for very large number of experiments or episodes as the algorithm will take indefinite time to converge. To yield better results hyper parameter tuning has been done to obtain the best values of ϵ (the exploration rate), β (the temperature of the softmax distribution), α (the learning rate) and γ (the discount factor).

A. Epsilon Decay

One common modification to the ϵ-greedy method is to gradually decrease the value of ϵ over time. This is known as ϵ-decay or annealing, and is used here to encourage the agent to explore more at the beginning of learning and to exploit more later on.

It is not a novel idea to explore throughout the training phase as the agent might not learn any optimal policies because finite number of experiments.

For example, one way to decay ϵ is to set it to some initial value ϵ_0 at the beginning of learning, and then to decay it

linearly over a fixed number of time steps. This can be done using the formula:

$$\epsilon_t = \max(\epsilon_{\text{min}}, \epsilon_0 \cdot \mathbf{r}^i) \tag{3}$$

It is important to start exploiting the current learnt policy by decreasing epsilon as we continue to each further episode. We have initially started with some value of ϵ_0 and then exponentially decrease it down to a threshold of $\epsilon_{min} = 0.05$ at a rate of 0.99^i , where i is the current episode number.

B. Choosing the best set of hyper parameters

Tuning the hyperparameters of SARSA and Q-learning using the epsilon-greedy policy involves finding the optimal values of the learning rate (α), discount rate (γ), and exploration rate (ϵ) that lead to the best performance of the agent in a given environment.

Because there are infinitely many combinations to choose α, ϵ, β and γ, there needs to be a metric that helps us to analyze the relative performance of the agent over varied set of parameters.

The frequently used metric to evaluate how good a learnt policy is the **expected cumulative reward** in the optimal policy derived from the optimal action value function.

Thus, a new function calc_reward(Q) is been defined to calculate the expected reward under the current optimal policy Q. The cumulative reward returned is the averaged reward returned over 3 experiments to obtain an unbiased estimate of the cumulative reward.

$$r_{param_1} > r_{param_2}$$
$$\implies \text{select param}_1$$

Here, r is the reward value.

C. Iterative Grid-search to obtain best hyper parameters

Many inbuilt optimization algorithms in scikit-learn have been explored including $forest_minimize$, $gbrt_minimize$ and $gp_minimize$. Not only are these algorithms computationally intensive, but are not yielding satisfactory results. Thus, we opted to move to finite grid search with each hyper parameter taking maximum of six values. Since, there is lot of computation involved in the estimation of action value function, we have split the search space into two random sub spaces which are run on two colab notebooks and found the optimal hyper parameters in each of the sub-spaces using the cumulative reward as the evaluation metric. We then choose the set which yielded better cumulative reward of the two sets that are found independently. The following are the set of values that each hyper parameter can take:

From prior knowledge the values of epsilon and gamma must range from $(0, 1)$ and α must be as small as possible. The temperature beta of softmax can take any real value theoretically. Thus, the following values are randomly sampled from which best hyperparamters are chosen.

$$\epsilon = [0.3, 0.7, 0.99, 0.1, 0.5, 0.9]$$

$$\beta = [0.05, 5, 50, 0.1, 1, 10]$$

$$\gamma = [0.3, 0.6, 0.8, 0.1, 0.5, 0.9]$$

$$\alpha = [0.7, 1, 20, 0.1, 0.5, 0.9]$$

and the two sub spaces that are split from the above space are:

$$\epsilon = [0.1, 0.5, 0.9]$$

$$\beta = [0.1, 1, 10]$$

$$\gamma = [0.1, 0.5, 0.9]$$

$$\alpha = [0.1, 0.5, 0.9]$$

and

$$\epsilon = [0.3, 0.7, 0.99]$$

$$\beta = [0.05, 5, 50]$$

$$\gamma = [0.3, 0.6, 0.8]$$

$$\alpha = [0.7, 1, 20]$$

Overall, tuning the hyperparameters of SARSA and Q-learning requires a combination of intuition, experimentation, and careful evaluation of the agent's performance. It is an iterative process that may require multiple rounds of testing and refinement to find the optimal set of hyperparameters for a given task or environment.

V. RESULTS

Now, we present the results of our experiments aimed at finding and tuning the optimal hyperparameters for SARSA and Q-learning algorithms using both epsilon-greedy and softmax policies. Specifically, we focused on four hyperparameters: learning rate (α), discount rate (γ), exploration rate (ϵ), and temperature (β). We employed a grid search approach to systematically explore a range of values for each hyperparameter, and evaluated the performance of the agents on the Grid-World reinforcement learning task. Our results provide insights into the effects of these hyperparameter values on the performance of SARSA and Q-learning agents, and offer recommendations for selecting optimal hyperparameters in the Grid-World problem

Sr. No.	Probab of good trans	Wind	Initial state	action taken	epsilon/beta	gamma	alpha	Reward
0	1	True	[[3,6]]	'epsilon greedy'	0.5	0.9	0.1	-11.6
1	1	True	[[3,6]]	'softmax'	0.1	0.9	0.1	-10.3
2	1	True	[[0,4]]	'epsilon greedy'	0.5	0.9	0.5	-11.3
3	1	True	[[0,4]]	'softmax'	0.1	0.9	0.5	-8.66
4	1	False	[[3,6]]	'epsilon greedy'	0.9	0.9	0.1	-17.33
5	1	False	[[3,6]]	'softmax'	1	0.9	0.1	-8
6	1	False	[[0,4]]	'epsilon greedy'	0.9	0.9	0.1	-10
7	1	False	[[0,4]]	'softmax'	1	0.9	0.5	-13.66
8	0.7	True	[[3,6]]	'epsilon greedy'	0.1	0.9	0.1	-8.33
9	0.7	True	[[3,6]]	'softmax'	1	0.9	0.5	-11.33
10	0.7	True	[[0,4]]	'epsilon greedy'	0.5	0.9	0.1	-26
11	0.7	True	[[0,4]]	'softmax'	1	0.9	0.1	-7.33
12	0.7	False	[[3,6]]	'epsilon greedy'	0.1	0.9	0.1	-6.67
13	0.7	False	[[3,6]]	'softmax'	0.1	0.9	0.5	-16.33
14	0.7	False	[[0,4]]	'epsilon greedy'	0.1	0.9	0.5	-10
15	0.7	False	[[0,4]]	'softmax'	0.1	0.9	0.5	-12

TABLE I
HYPERPARAMETERS AND REWARD VALUES FOR SARSA

Sr. No.	Probab of good trans	Wind	Initial state	action taken	epsilon/beta	gamma	alpha	Reward
0	1	True	[[3,6]]	'epsilon greedy'	0.5	0.9	0.5	-11.33
1	1	True	[[3,6]]	'softmax'	0.1	0.9	0.9	-9.33
2	1	True	[[0,4]]	'epsilon greedy'	0.5	0.9	0.5	-7.66
3	1	True	[[0,4]]	'softmax'	0.1	0.9	0.1	-8
4	1	False	[[3,6]]	'epsilon greedy'	0.1	0.1	0.1	-9.66
5	1	False	[[3,6]]	'softmax'	1	0.5	0.1	-6.66
6	1	False	[[0,4]]	'epsilon greedy'	0.5	0.9	0.5	-11
7	1	False	[[0,4]]	'softmax'	10	0.9	0.9	-10
8	0.7	True	[[3,6]]	'epsilon greedy'	0.9	0.9	0.1	-9.66
9	0.7	True	[[3,6]]	'softmax'	1	0.9	0.1	-6.33
10	0.7	True	[[0,4]]	'epsilon greedy'	0.5	0.9	0.5	-7.66
11	0.7	True	[[0,4]]	'softmax'	1	0.9	0.5	-7
12	0.7	False	[[3,6]]	'epsilon greedy'	0.5	0.9	0.9	-11
13	0.7	False	[[3,6]]	'softmax'	0.1	0.9	0.1	-8.33
14	0.7	False	[[0,4]]	'epsilon greedy'	0.1	0.5	0.5	-12
15	0.7	False	[[0,4]]	'softmax'	0.1	0.9	0.9	-9.33

TABLE II
HYPERPARAMETERS AND REWARD VALUES FOR Q-LEARNING

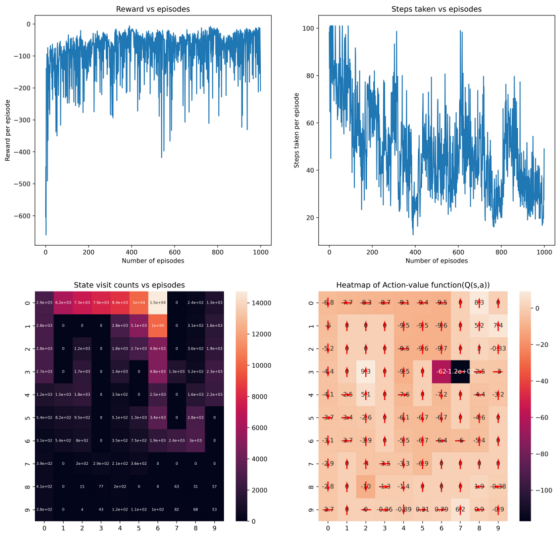

Fig. 1. SARSA Configuration 1 ϵ/β = 0.5 , γ = 0.9 , α = 0.1

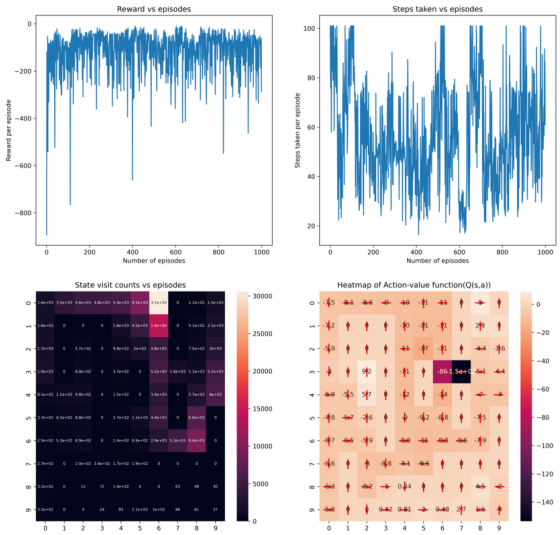

Fig. 3. SARSA Configuration 3 ϵ/β = 0.5, γ = 0.9 , α = 0.5

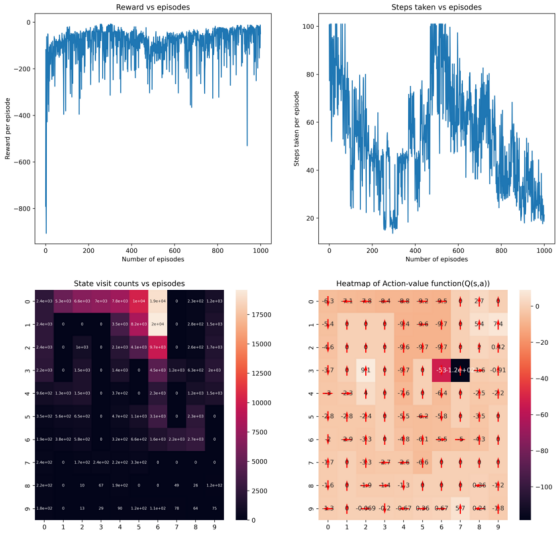

Fig. 2. SARSA Configuration 2 ϵ/β = 0.1 , γ = 0.9, α = 0.1

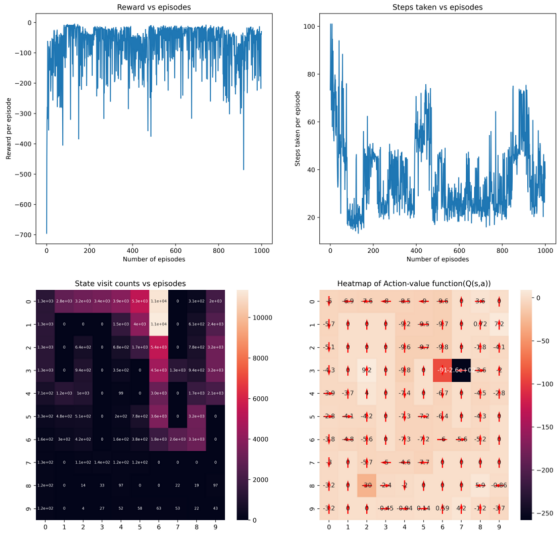

Fig. 4. SARSA Configuration 4 ϵ/β = 0.1 , γ = 0.9, α =0.5

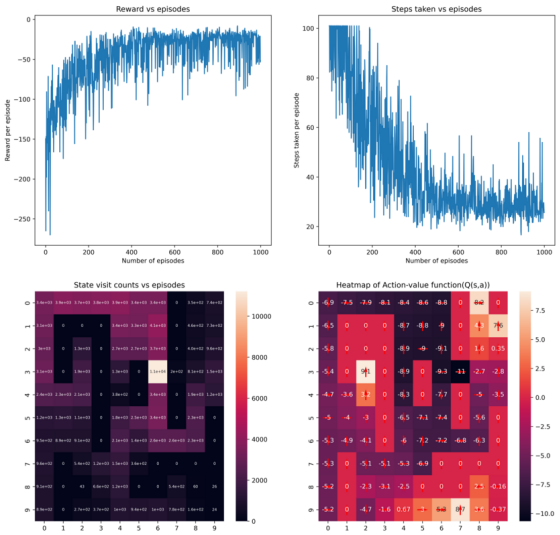

Fig. 5. SARSA Configuration 5 ϵ/β = 0.9 , γ = 0.9 , α = 0.1

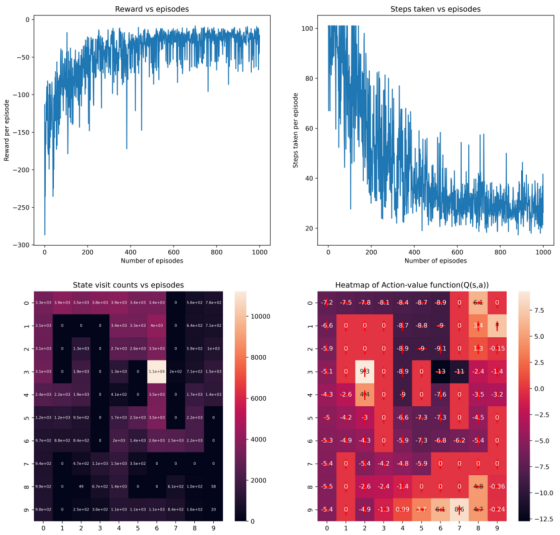

Fig. 7. SARSA Configuration 7 ϵ/β = 0.9 , γ = 0.9, α = 0.1

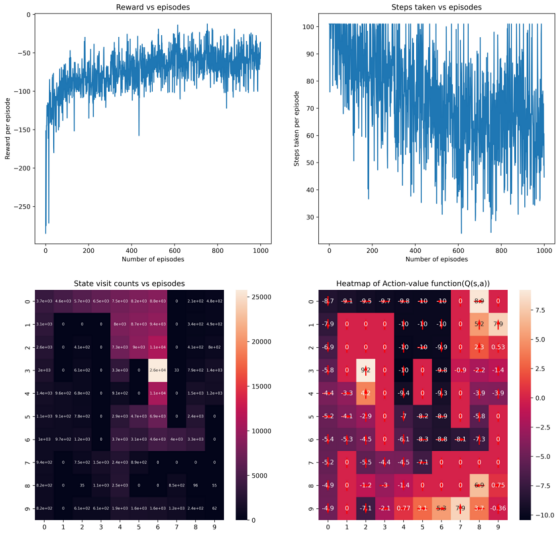

Fig. 6. SARSA Configuration 6 ϵ/β = 1 , γ = 0.9, α = 0.1

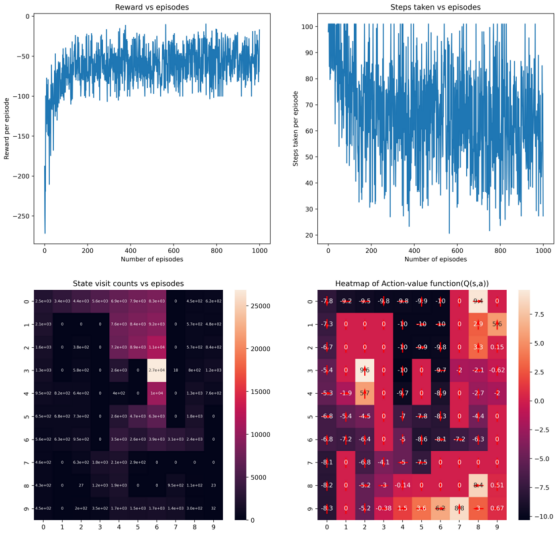

Fig. 8. SARSA Configuration 8 ϵ/β = 1, γ = 0.9, α = 0.5

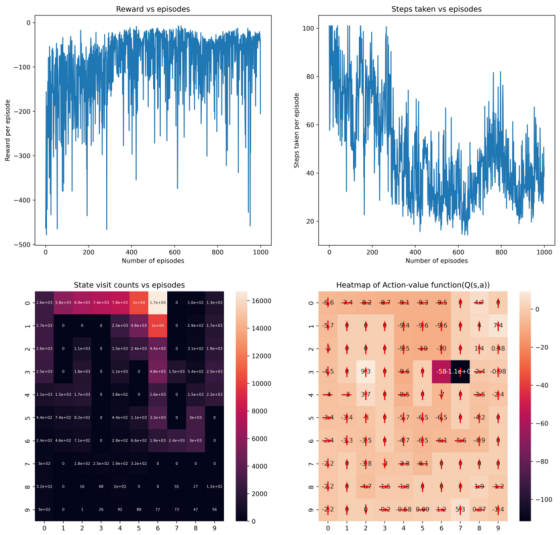

Fig. 9. Configuration 9 ϵ/β = 0.1, γ = 0.9 , α = 0.1

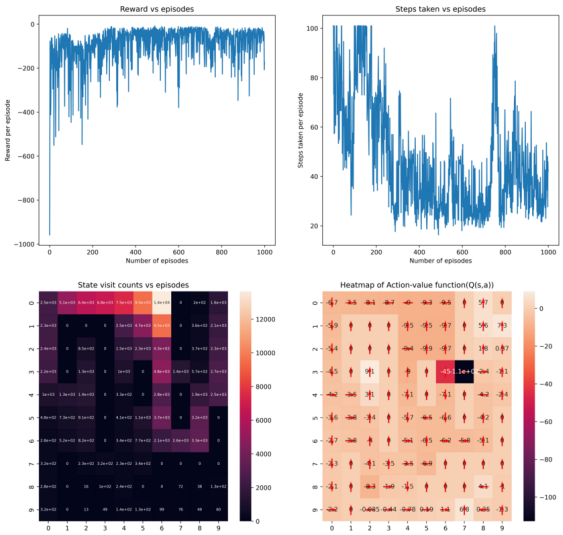

Fig. 11. SARSA Configuration 11 ϵ/β = 0.5, γ = 0.9 , $alpha$ = 0.1

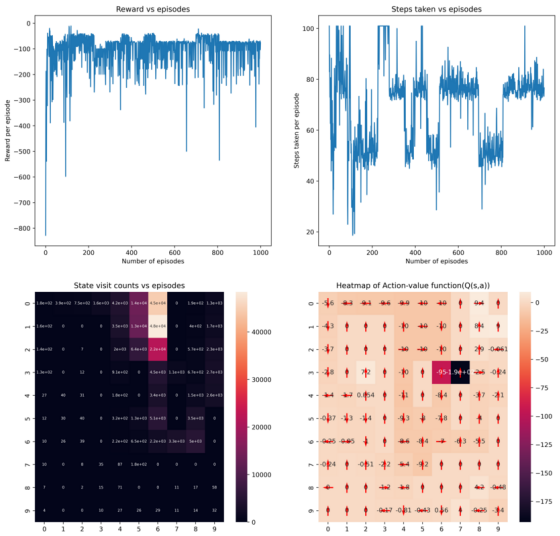

Fig. 10. SARSA Configuration 10 ϵ/β = 1, γ = 0.9 , α = 0.5

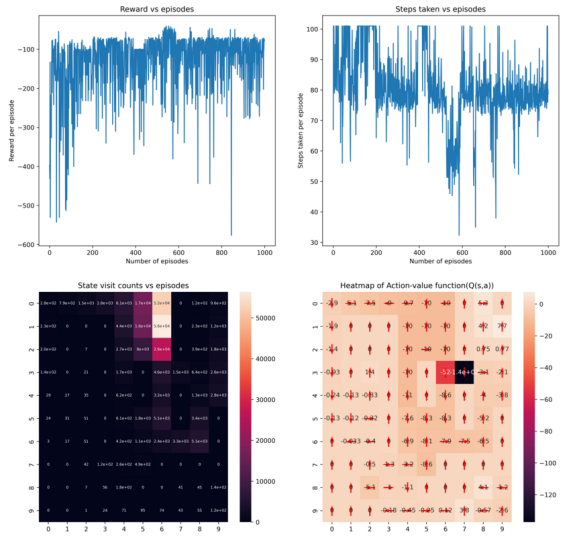

Fig. 12. SARSA Configuration 12 ϵ/β = 1, γ = 0.9 , α = 0.1

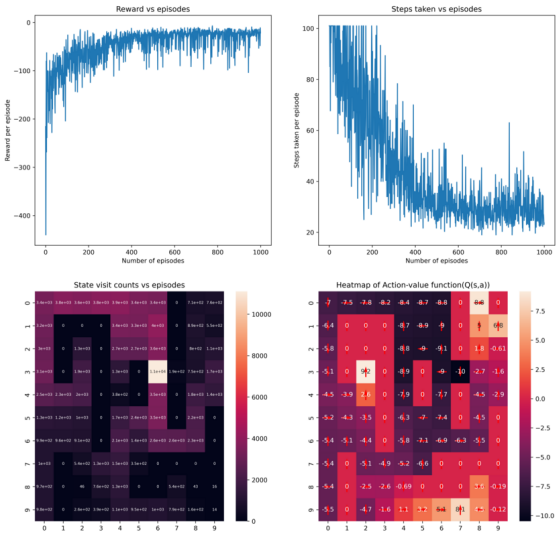

Fig. 13. SARSA Configuration 13 ϵ/β = 0.1 , γ = 0.9 , α= 0.1

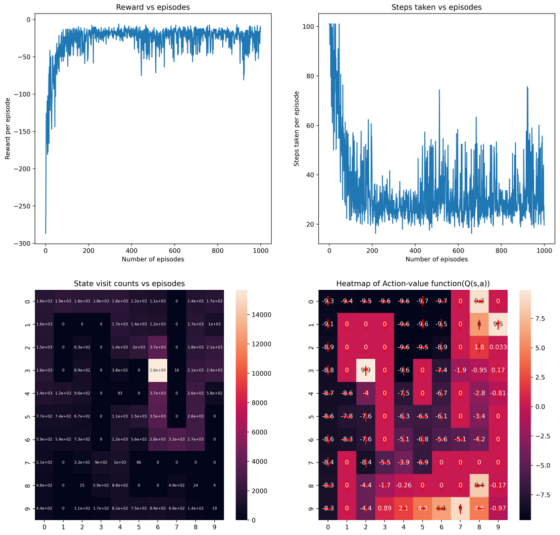

Fig. 14. SARSA Configuration 14 ϵ/β = 0.1 , γ = 0.9, α = 0.5

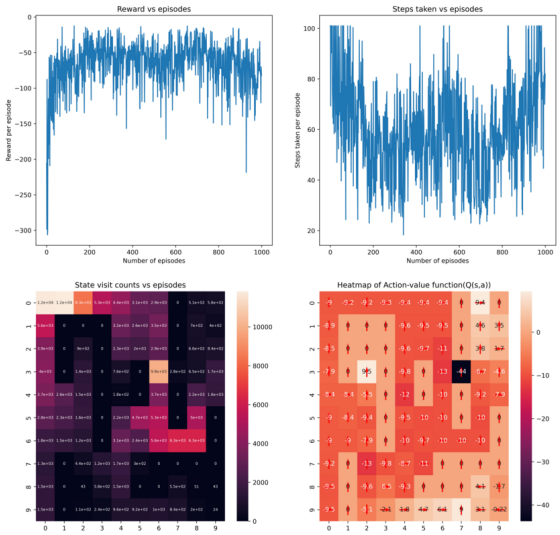

Fig. 15. SARSA Configuration 15 ϵ/β = 0.1 , γ = 0.9, α = 0.5

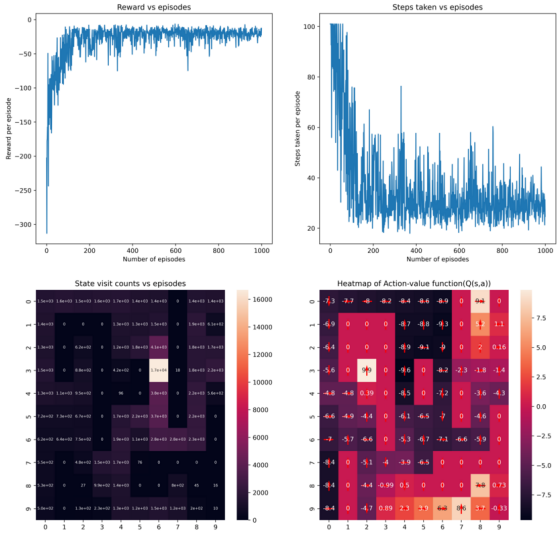

Fig. 16. SARSA Configuration 16 ϵ/β = 0.1 , γ = 0.9, α = 0.5

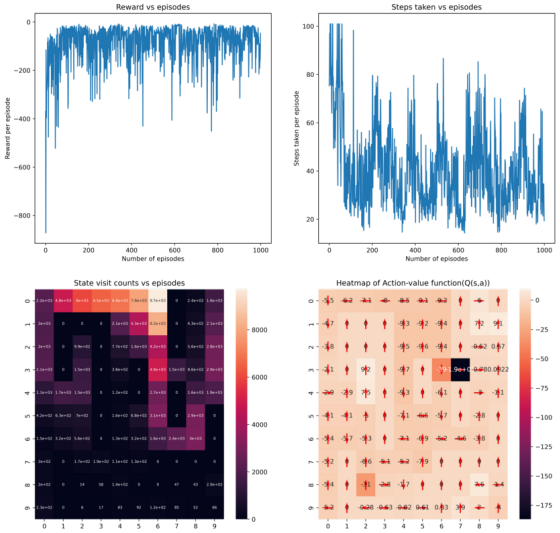

Fig. 17. Q-Learn Configuration 1 ϵ/β = 0.5 , γ = 0.9 , α = 0.5

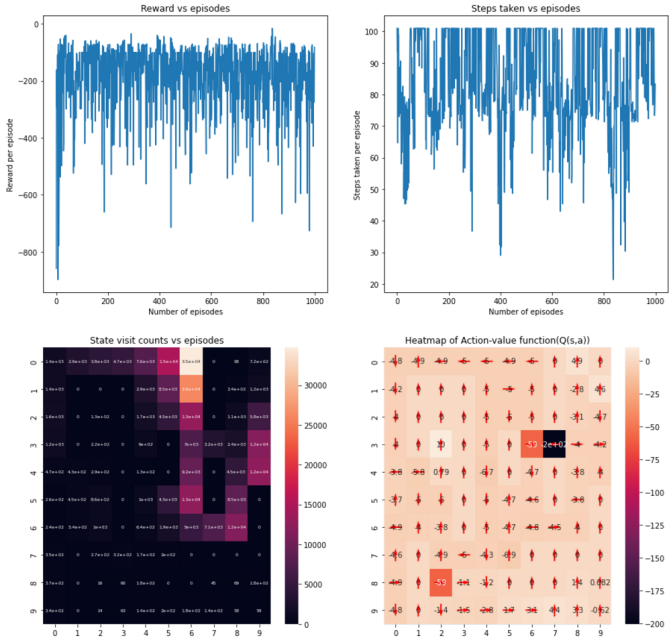

Fig. 19. Q-Learn Configuration 3 ϵ/β = 0.5 , γ = 0.9 , α= 0.5

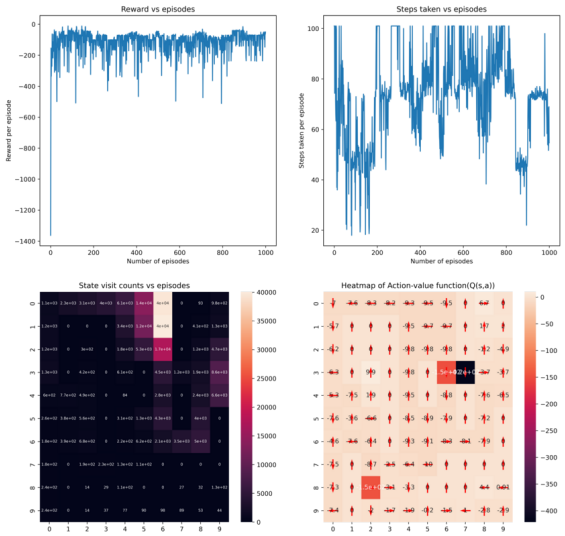

Fig. 18. Q-Learn Configuration 2 ϵ/β = 0.1 , γ = 0.9 , α = 0.9

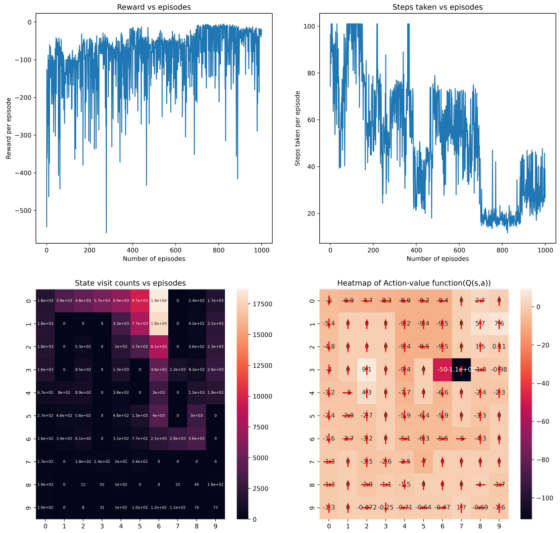

Fig. 20. Q-Learn Configuration 4 ϵ/β = 0.1 , γ = 0.9 , α = 0.1

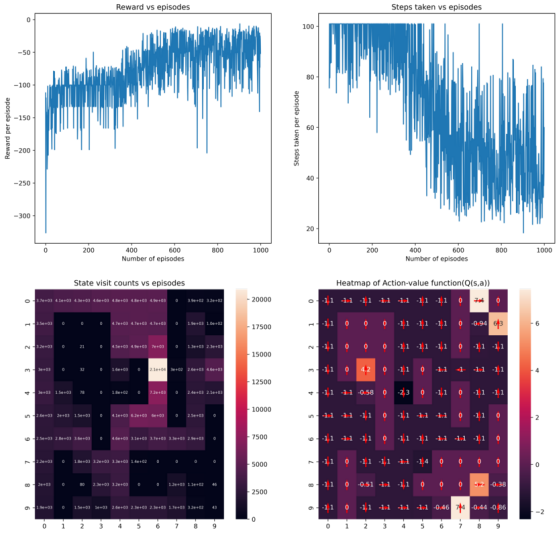

Fig. 21. Q-Learn Configuration 5 ϵ/β = 0.1 , γ = 0.1 , α = 0.1

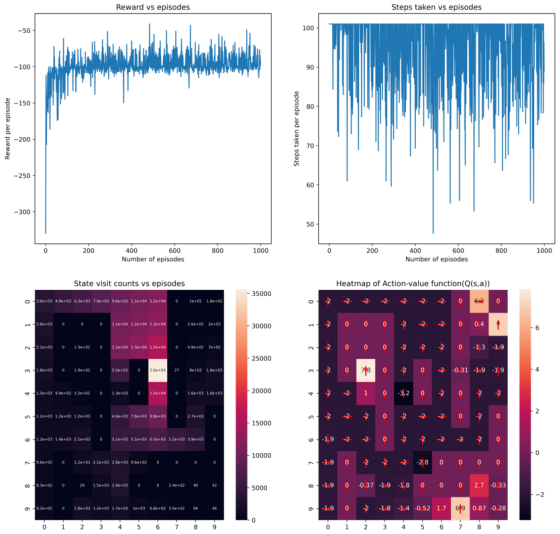

Fig. 22. Q-Learn Configuration 6 ϵ/β = 1 , γ = 0.5 , α = 0.1

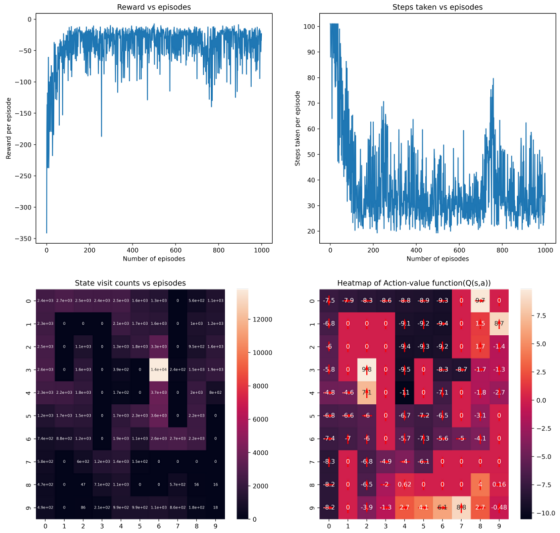

Fig. 23. Q-Learn Configuration 7 ϵ/β = 0.5 , γ = 0.9 , α = 0.5

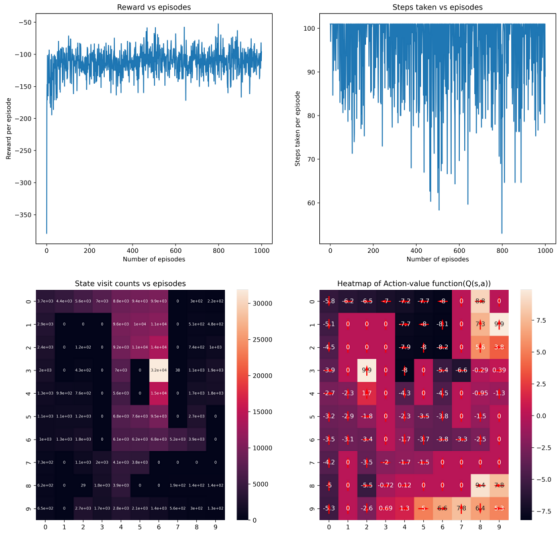

Fig. 24. Q-Learn Configuration 8 ϵ/β = 10 , γ = 0.9 , α = 0.9

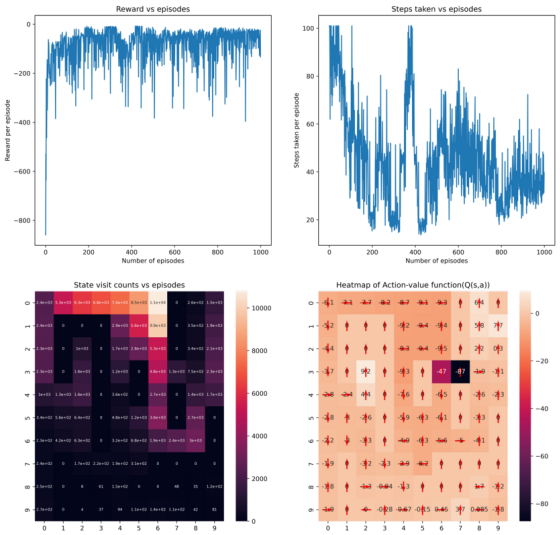

Fig. 25. Q-Learn Configuration 9 $\epsilon/\beta = 0.9$, $\gamma = 0.9$, $\alpha = 0.1$

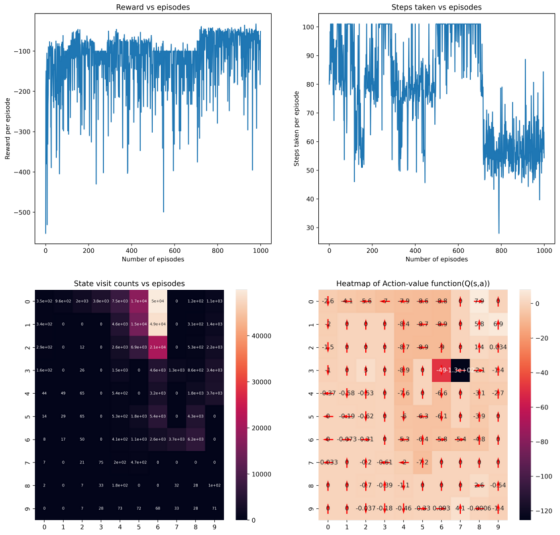

Fig. 26. Q-Learn Configuration 10 $\epsilon/\beta = 1$, $\gamma = 0.9$, $\alpha = 0.1$

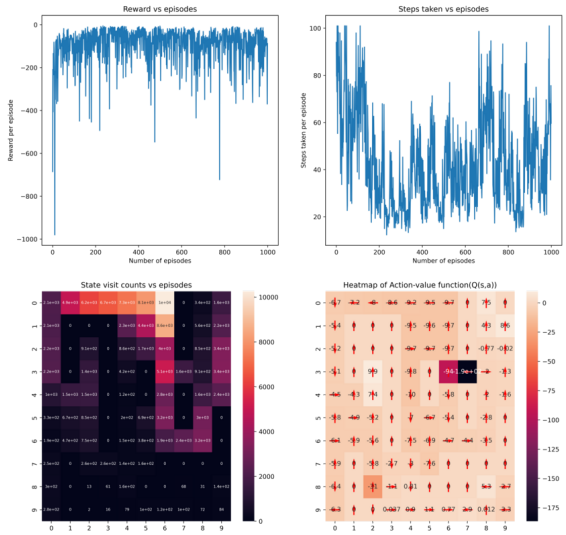

Fig. 27. Q-Learn Configuration 11 $\epsilon/\beta = 0.5$, $\gamma = 0.9$, $\alpha = 0.5$

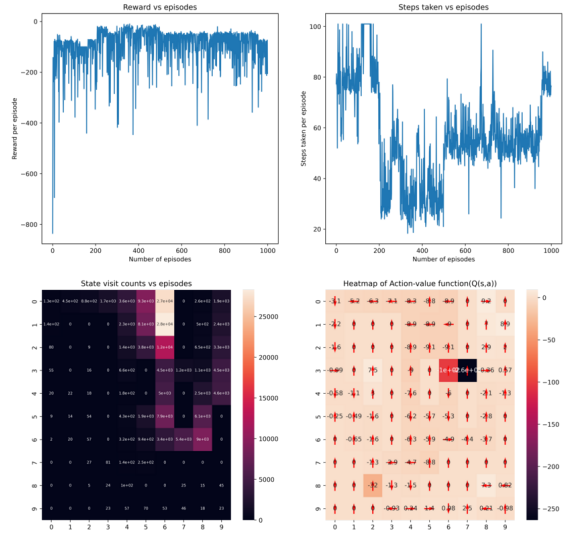

Fig. 28. Q-Learn Configuration 12 $\epsilon/\beta = 1$, $\gamma = 0.9$, $\alpha = 0.5$

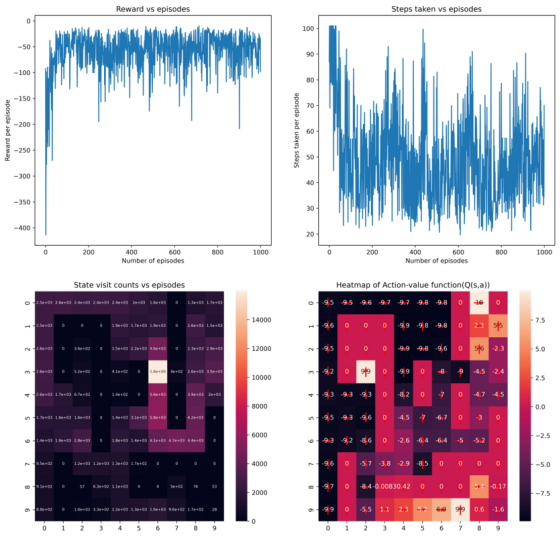

Fig. 29. Q-Learn Configuration 13 ϵ/β = 0.5 , γ = 0.9 , α = 0.9

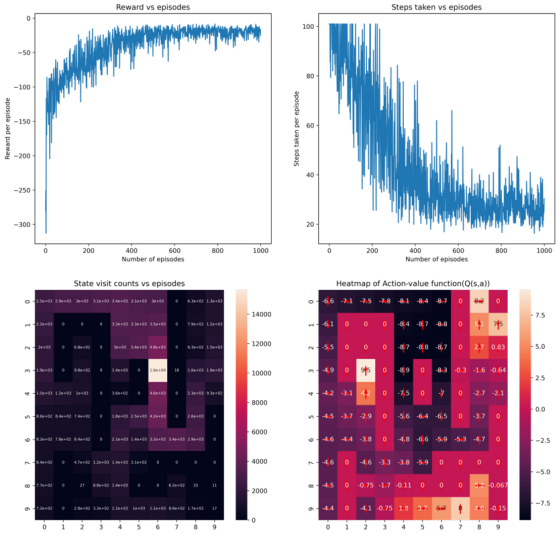

Fig. 30. Q-Learn Configuration 14 ϵ/β = 0.1 , γ = 0.9 , α = 0.1

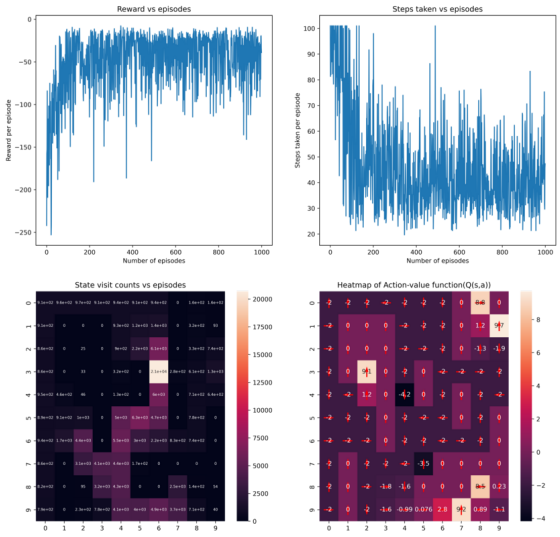

Fig. 31. Q-Learn Configuration 15 ϵ/β = 0.1 , γ = 0.5 , α = 0.5

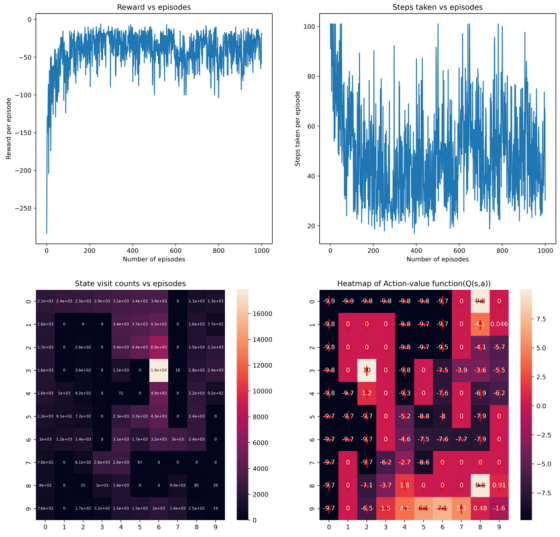

Fig. 32. Q-Learn Configuration 16 ϵ/β = 0.1 , γ= 0.9 , α= 0.9

VI. Conclusions from the learnt graphs

Intuition: The reward curves and number of states visited will have more variance when there is stochasticity in the environment. Additional stochaticity is induced in the environment if the transition is not deterministic, i.e p_good_trans not equal to one. The factors on which the reward curves and Action value curves depend on are:

By looking at the environment, our initial guess would be that goal state would be (8,7) if the start state is (3,6) and better goal state would be (2,2) for a initial state of (0,4) when the process is deterministic

- Probability of good transition
- Initial state
- Wind conditions

A. SARSA:

- Configuration 1: The stochasticity involved here is due to presence of wind. Thus the observed variation. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The policy has also reached the goal state [[8,7]] but only a few number of times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -120 at state [3,7], from which we can infer that it is really a very bad state.
- Configuration 2: The stochasticity involved here is due to presence of wind. During the initial phases of training the agent is more exploratory in nature thus, we see many a times it got terminated after 100 timesteps. Eventually, it learnt an optimal policy and thus there is a sharp decrease in number of steps taken towards the end. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The policy has also reached the goal state [[8,7]] but only a few number of times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -1200 at state [3,7], from which we can infer that it is really a very bad state and thus we can conclude that agent has learnt a reasonably good policy.
- Configuration 3: The stochasticity involved here is due to presence of wind. The agent of training the agent is more exploratory in nature thus, we see many a times it got terminated after 100 timesteps. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The policy has also reached the goal state [[8,7]] but only a few number of times. The reason that agent reached the goal state [0,9] most number of times is because there is bad state at the state [4,2], due to which there is a high negative reward. The action value function has a value -150 at state [3,7], from which we can infer that it is really a very bad state and thus we can conclude that agent has learnt a reasonably good policy, by avoiding the restart state
- Configuration 4: The stochasticity involved here is due to presence of wind. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The policy has also reached the goal state [[8,7]] but only a few number of times. The reason that agent reached the goal state [0,9] most number of times is because there is bad state at the state [4,2], due to which there is a high negative reward, which discourages the agent to reach the goal state at [[1,2]]. The action value function has a value -260 at state [3,7], from which we can infer that it is really a very bad state and thus we can conclude that agent has learnt a reasonably good policy, by avoiding the restart state and taking an alternate route to the state [[0,9]] via [[6,7]]
- Configuration 5: There is no stochasticity involved here is due to absence of wind and probaibilty of good transition is 1. The policy is reaching the goal state [[0,9]], [[2,2]] and [[8,7]] with reasonable frequencies. The action value function has a value -11 at state [3,7], from which we can infer that it is really a very bad state and thus we can conclude that agent has learnt a reasonably good policy, by avoiding the restart state and taking an alternate route to the state [[0,9]] via [[6,7]]. Thus both the reward curves and steps taken in each episode have converged to a reasonable value.
- Configuration 6: There is no stochasticity involved here is due to absence of wind and probability of good transition is 1. The policy is reaching the goal state [[0,9]], [[2,2]] and [[8,7]] with reasonable frequencies. Thus both the reward curves and steps taken in each episode have converged to a reasonable value. The policy is reaching the goal state [[8,7]] with relatively high frequency as it almost avoids any bad or restart state in its path
- Configuration 7: There is no stochasticity involved here is due to absence of wind and probability of good transition is 1. The policy is reaching the goal state [[0,9]], [[2,2]] and [[8,7]] with reasonable frequencies. The action value function has a value -11 at state [3,7], from which we can infer that it is really a very bad state and thus we can conclude that agent has learnt a reasonably good policy, by avoiding the restart state and taking an alternate route to the state [[0,9]] via [[6,7]].
- Configuration 8: There is no stochasticity involved here is due to absence of wind and probability of good transition is 1. The policy is reaching the goal state [[0,9]], [[2,2]] and [[8,7]] with reasonable frequencies. Thus both the reward curves and steps taken in each episode have converged to a reasonable value. The policy is reaching the goal state [[8,7]] with relatively high frequency as it almost avoids any bad or restart state in its path leading to higher reawards.
- Configuration 9: The stochasticity involved here is due to presence of wind and also the transition is not deterministic. Thus the observed variation in observed rewards. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The policy has also reached the goal state [[8,7]] but only a few number of times. From the optimal policy learnt, the agent has completely avoided

the restart state at [3,7]. The action value function has a value -120 at state [3,7], from which we can infer that it is really a very bad state. The agent is reaching the goals [0,9] and [2,2] most of the times. The agent successfully learnt a policy that completely avoided the restart state at [3,7], because of the high negative reward that is obtained if the agent lands on that state.

- Configuration 10: The stochasticity involved here is due to presence of wind and also the transition is not deterministic. The policy is reaching the goal state [[0,9]] most of the times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -190 at state [3,7], from which we can infer that it is really a very bad state. The agent is reaching the goals [0,9] .The agent successfully learnt a policy that completely avoided the restart state at [3,7], because of high stochasticity in the environment, the agent is more exploratory in nature to find an optimal policy which is reflected in the value of beta.
- Configuration 11: The stochasticity involved here is due to presence of wind and also the transition is not deterministic. The policy is reaching the goal state [[0,9]] most of the times and [2,2] a reasonable number of times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -110 at state [3,7], from which we can infer that it is really a very bad state. The agent successfully learnt a policy that completely avoided the restart state at [3,7] by reaching the goal state via [6,7]. Because of high stochasticity in the environment, the agent is more exploratory in nature to find an optimal policy which is reflected as a very high value of epsilon(0.5).
- Configuration 12: The stochasticity involved here is due to presence of wind and also the transition is not deterministic. The policy is reaching the goal state [[0,9]] most of the times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -140 at state [3,7], from which we can infer that it is really a very bad state. The agent successfully learnt a policy that completely avoided the restart state at [3,7] by reaching the goal state via [6,7]. Because of high stochasticity in the environment, the agent is more exploratory in nature to find the optimal policy which is reflected in the value of beta.
- Configuration 13: The stochasticity involved here is because of the fact that transition is not deterministic. The policy is reaching the goal state [[2,2]] most of the times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -10 at state [3,7], from which we can infer that it is really a very bad state. The agent successfully learnt a policy that completely avoided the restart state at [3,7] by reaching the goal state via [6,7]. Initially the agent is exploratory but then eventually learnt an optimal policy.
- Configuration 14: The stochasticity involved here is because of the fact that transition is not deterministic. The policy is reaching the goal state [[0,9]] most of the times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. Initially the agent is exploratory but then eventually learnt an optimal policy.
- Configuration 15: The stochasticity involved here is because of the fact that transition is not deterministic. The policy is reaching all the goal states [[0,9], [2,2],[8,7]] with relatively same frequencies. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -4.4 at state [3,7], from which we can infer that it is really a very bad state. The agent successfully learnt a policy that completely avoided the restart state at [3,7] by reaching the goal state via [6,7] or alternatively reaching other goal states.
- Configuration 16: The stochasticity involved here is because of the fact that transition is not deterministic. The policy is reaching all the goal states [[0,9], [2,2],[8,7]] with relatively same frequencies with [0,9] having a higher value. The agent is initially exploratory but later learnt an optimal policy

B. Q-Learning:

- Configuration 1: This is very similar to the SARSA case. Initially the graph is exploratory and then as observed in the steps taken vs episode graph, the trend is converging. The stochasticity involved here is due to presence of wind, explaining the observed variation. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The policy has also reached the goal state [[8,7]] more number of times than SARSA. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -190 at state [3,7], from which we can infer that it is really a very bad state, which is obvious for a restart state.
- Configuration 2: The stochasticity involved here is due to presence of wind. We see many a times it got terminated after 100 timesteps, leading to the conclusion that the agent was more exploratory. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The policy has also reached the goal state [[8,7]] but only a few number of times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7](Q[3,7] = -220)
- Configuration 3: This is very similar to the SARSA case. The stochasticity involved here is due to presence of wind. The agent of training the agent is more exploratory in nature thus, we see many a times it got terminated after 100 timesteps. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The reason that agent reached the goal state [0,9] most number of times is because there is bad state at the state [4,2], due to which there is a high negative reward. The action value function has a value -170 at state [3,7], from which we can infer that it is really a very bad state and thus we can

conclude that agent has learnt a reasonably good policy, by avoiding the restart state

- Configuration 4: This is very similar to the SARSA case. The stochasticity involved here is due to presence of wind. The policy is reaching the goal state [[0,9]] and [[2,2]] most of the times. The policy has also reached the goal state [[8,7]] but only a few number of times. The action value function has a value -110 at state [3,7], from which we can infer that it is really a very bad state and thus we can conclude that agent has learnt a reasonably good policy, by avoiding the restart state and taking an alternate route to the state [[0,9]] via [[6,7]]
- Configuration 5: There is no stochasticity involved here is due to absence of wind. The policy is reaching the goal state [[0,9]] and [[8,7]] with reasonable frequencies.
- Configuration 6: There is no stochasticity involved here is due to absence of wind and probability of good transition is 1. The policy is reaching the goal state [[0,9]], [[2,2]] and [[8,7]] with reasonable frequencies. Reaching the state [[8,7]] is more favourable for the agent as it will avoid any bad state or restart state
- Configuration 7: This is very similar to the SARSA case. There is no stochasticity involved here is due to absence of wind and probability of good transition is 1. The policy is reaching the goal state [[0,9]], [[2,2]] and [[8,7]] with reasonable frequencies. The action value function has a value -8.7 at state [3,7], from which we can infer that it is a bad state and thus we can conclude that agent has learnt a reasonably good policy, by avoiding the restart state and taking an alternate route to the goal state [[0,9]] via [[6,7]]. The algorithm initially is exploratory in nature but later learns an optimal policy, which is reflected both in reward curves and number of steps.
- Configuration 8: This is similar to the SARSA case. There is no stochasticity involved here is due to absence of wind and probability of good transition is 1. The policy is reaching the goal state [[0,9]], [[2,2]] and [[8,7]] with reasonable frequencies. The policy is reaching the goal state [[8,7]] and [[0,9]] with relatively high frequencies as it almost avoids any bad or restart state in its path leading to better.
- Configuration 9: The stochasticity involved here is due to presence of wind and also the transition is not deterministic. Thus the observed variation in observed rewards. We observe that the variance in reward per episode is less than the variance in SARSA. This is because of the fact that Qlearning is more greedy compared to SARSA. The policy is reaching the goal state [[0,9]] most of the times. The policy has also reached the goal state [[8,7]] but only a few number of times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -120 at state [3,7], from which we can infer that it is really a very bad state. The agent is reaching the goals [0,9] and [2,2] most of the times. The agent successfully learnt a policy that completely avoided the restart state at [3,7], because of the high negative reward that is obtained if the agent lands on that state.
- Configuration 10: This configuration learns much faster than the SARSA case and becomes less exploratory as the number of episodes increases. The stochasticity involved here is due to presence of wind and also the transition is not deterministic. The policy is reaching the goal state [[0,9]] most of the times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -130 at state [3,7], from which we can infer that it is really a very bad state. The agent is reaching the goals [0,9] .The agent successfully learnt a policy that completely avoided the restart state at [3,7], because of high stochasticity in the environment, the agent is more exploratory in nature to find an optimal policy which is reflected in the value of beta.
- Configuration 11: This is quite similar to the SARSA case, and is very similar in the exploitative vs exploratory nature. It is stochastic due to wind and also the transition is not deterministic. The policy is reaching the goal state [[2,2]] most of the times and [0,9] a reasonable number of times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -190 at state [3,7], from which we can infer that it is really a very bad state. The agent successfully learnt a policy that completely avoided the restart state at [3,7] by reaching the goal state via [6,7]. Because of high stochasticity in the environment, the agent is more exploratory in nature to find an optimal policy which is reflected as a very high value of epsilon (0.5).
- Configuration 12: This is similar to the SARSA case, but slightly more exploitative. The stochasticity involved here is due to presence of wind and also the transition is not deterministic. The policy is reaching the goal state [[0,9]] most of the times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -260 at state [3,7], from which we can infer that it is really a very bad state. The agent successfully learnt a policy that completely avoided the restart state at [3,7] by reaching the goal state via [6,7]. Because of high stochasticity in the environment, the agent is more exploratory in nature to find the optimal policy which is reflected in the value of beta.
- Configuration 13: This is very similar to the SARSA case. The stochasticity involved here is because of the fact that transition is 0.7 and non deterministic. The policy is reaching the goal state [[0,9]] most of the times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. The action value function has a value -9 at state [3,7], from which we can infer that it is really a very bad state. The agent successfully learnt a policy that completely avoided the restart state at [3,7] by reaching the goal state via [6,7]. Initially the agent is exploratory but then eventually learnt an optimal policy.

- Configuration 14: This is very similar to the SARSA case, but the SARSA case learns much more quickly. By itself, it is also a very quick learner, and becomes monotonically more exploitative as the number of episodes increases. The stochasticity involved here is because of the fact that transition is not deterministic. The policy is reaching the goal state [[2,2]] most of the times. From the optimal policy learnt, the agent has completely avoided the restart state at [3,7]. Initially the agent is exploratory but has quickly learnt an optimal policy.
- Configuration 15: This is similar to the SARSA case, but learns more quickly. The stochasticity involved here is because of the non-deterministic transition of 0.7. The policy reaches all the goal states [[0,9], [2,2],[8,7]] with similar frequencies, with [9,0] being the most frequent. The agent successfully learns a policy that completely reaches the goal state [0,9] and alternatively reaches the other goal states too.
- Configuration 16: This is much more stochastic compared to the SARSA case, as the transition is not deterministic. The policy reaches all the goal states [[0,9], [2,2],[8,7]] with relatively same frequencies with [2,2] having a higher value. The agent has learnt the optimal policy by 1000 episodes, however, it is learning slower than SARSA for the same hyperparameter values.

VII. General Conclusions

- QLearning is greedy with respect to the next action action value
- SARSA returns a more safer path(has higher expected reward) , but where as Q Learning try to return the shortest path possible.
- The value of hyperparameters are highly dependent upon the probability of good transition, presence of wind and the initial state of the agent, which is evident from the above analysis
- More stochasticity in the environment needs more exploration by the agent to converge to the optimal policy.