CS - 6700 Reinforcement Learning

Programming Assignment -2

Mohammed Khandwawala EE16B117

Contents

1 Puddle Gridworld

Custom environment using OpenAI-Gym for Puddle Gridworld was created with the the reward distribution given below. Shaded squares represent the start states. Valid actions are movement of one block in any direction .Actions are taken with probability 0.9 and other three actions are selected with probability 0.1/3 each.

There is also a wind that pushes the playes one block east with probability 0.5 , this exist only in cast goals A and B.

Reward for reaching any of the goal state is +10. There is no effect of a action that takes player out of the grid.

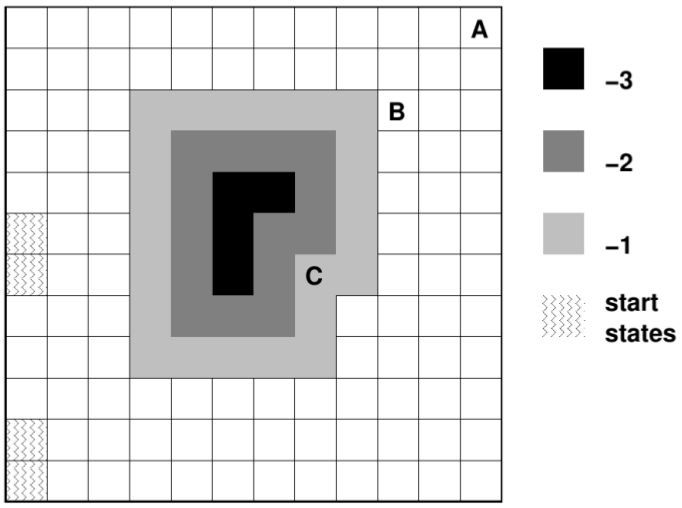

Figure 1: The puddle-world.

Figure 1: Details of the gridworld

2 Question 1

2.1 Q-Learning

2.1.1 Goal A

Below are the plots for average number of step and reward over 50 runs for Q learning. The Average number of steps converged to 21 and average reward converged at 10.Since the policy learned to avoid the puddle hence collect no negative reward is collected on the way to goal.

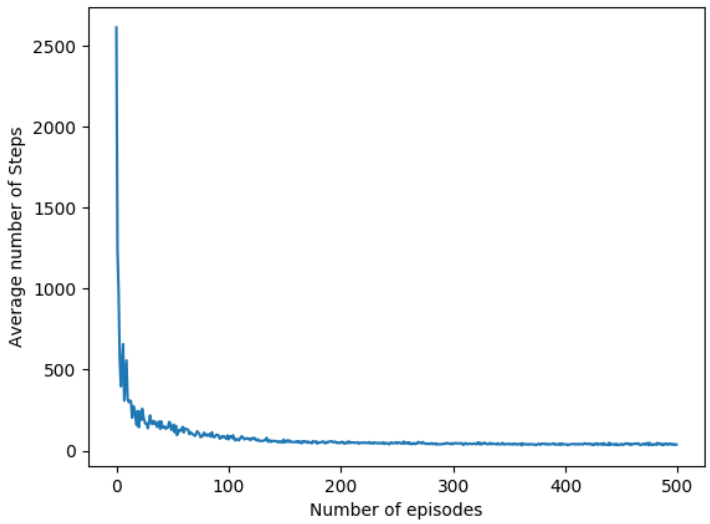

Figure 2: Average number of steps required plotted against the episode number. Averaged over 50 runs

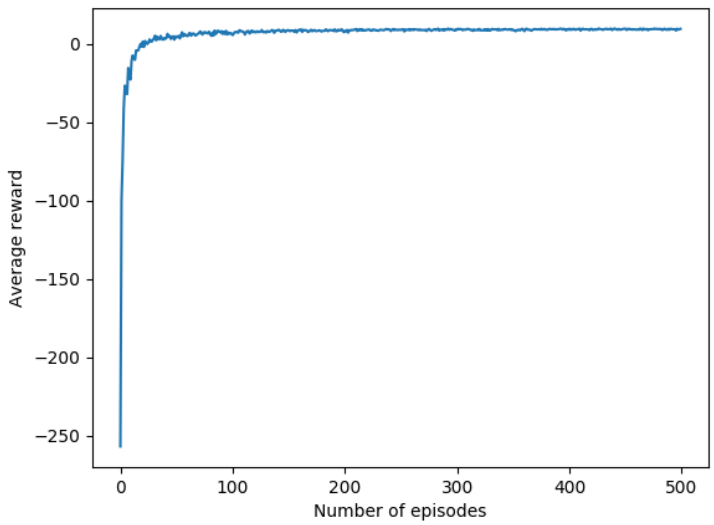

Figure 3: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the plot of policy obtained. The action plotted below are the selected by taking the most frequent action at a particular in the 50 policies obtained.

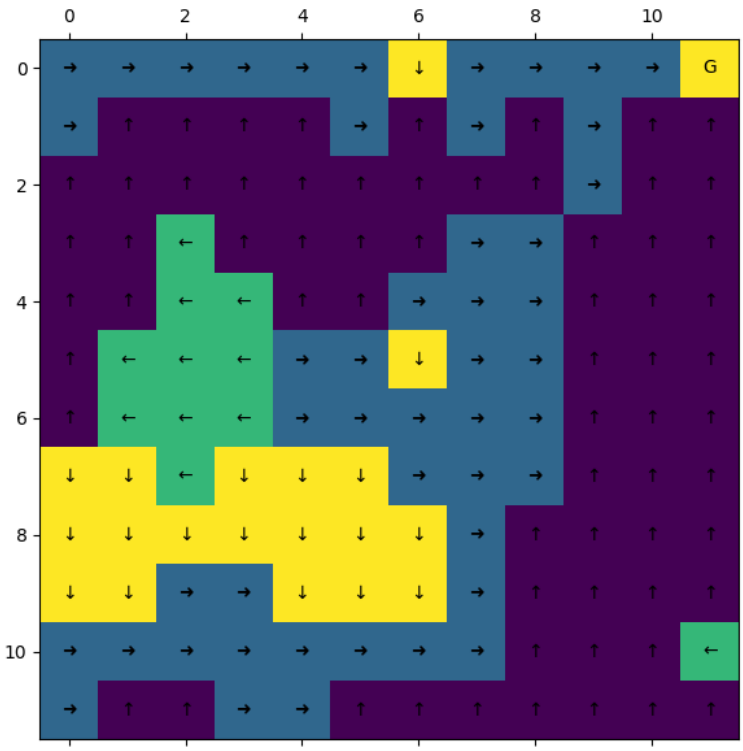

Figure 4: Visualization of the Policy learned by the Q-Learning. arrow represents the direction of action policy takes at that particular state

2.1.2 Goal B

Very similar to goal A the average number of steps to reach Goal B converged at 20 .And the reward converged at 10 as the policy learned to avoid the puddle.

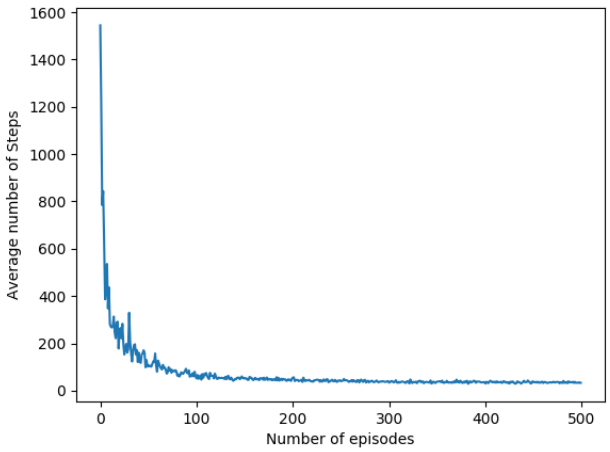

Figure 5: Average number of steps required plotted against the episode number. Averaged over 50 runs

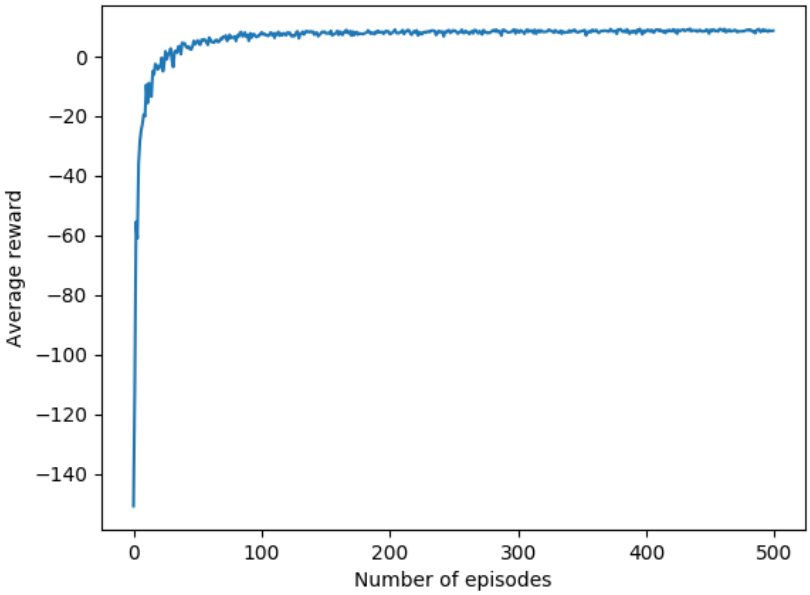

Figure 6: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the visualization of the policy learned by Q learning.

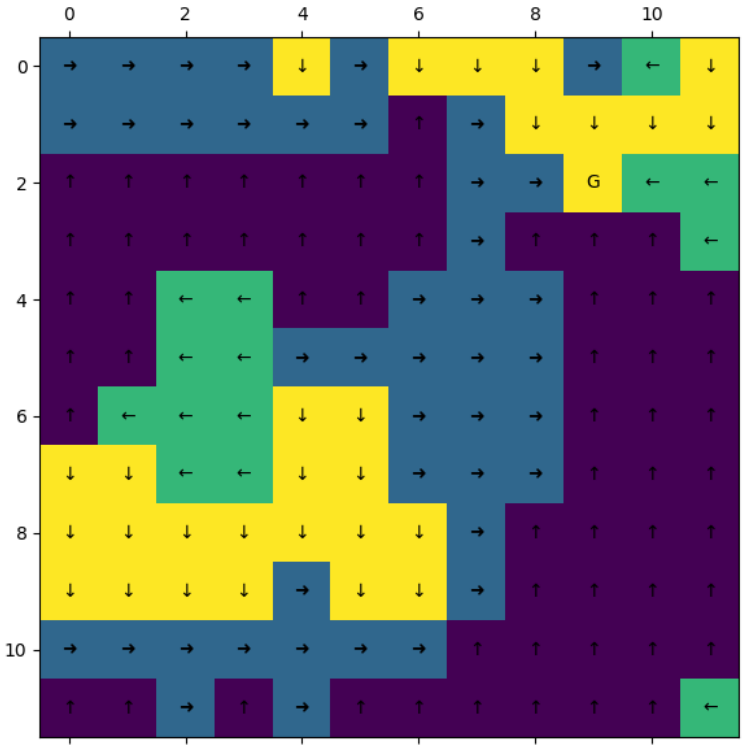

Figure 7: Visualization of the Policy learned by the Q-Learning. arrow represents the direction of action policy takes at that particular state

2.1.3 Goal C

For goal C the wind was removed as instructed in the question. The average number of steps converged to 16 .And the reward converged to 8.9.

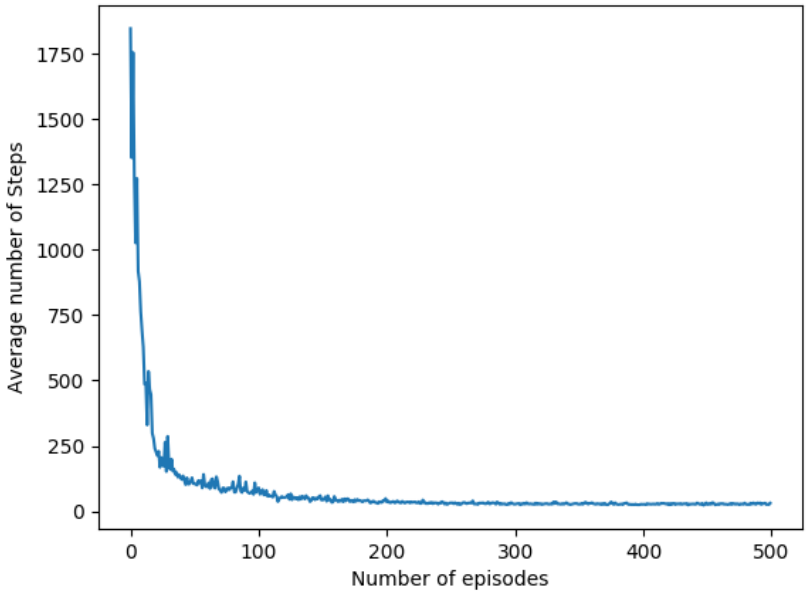

Figure 8: Average number of steps required plotted against the episode number. Averaged over 50 runs

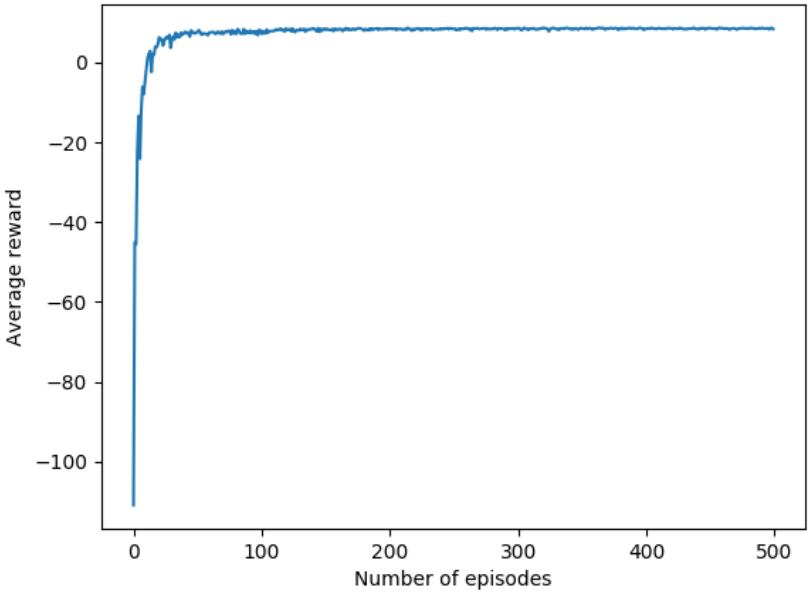

Figure 9: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the visualisation of the policy learned.

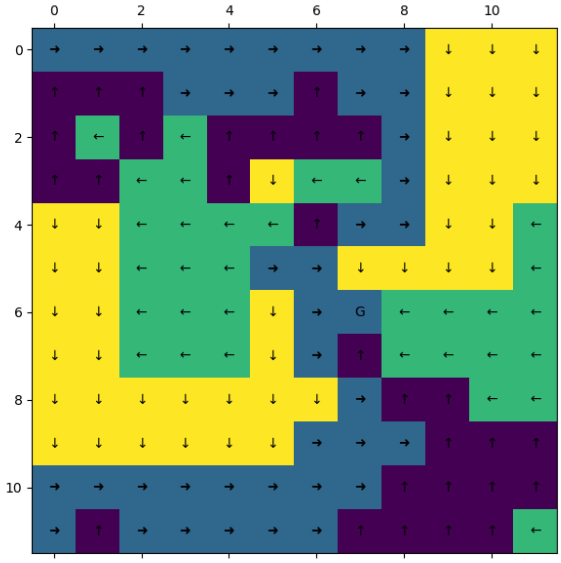

Figure 10: Visualization of the Policy learned by the Q-Learning. arrow represents the direction of action policy takes at that particular state

2.2 SARSA

2.2.1 Goal A

Similar to Q learning similar plots were obtained for SARSA. SARSA converged faster than Q-Learning 21. Although the reward values recorded at convergence are same as expected. Average number of steps taken are 26.5.

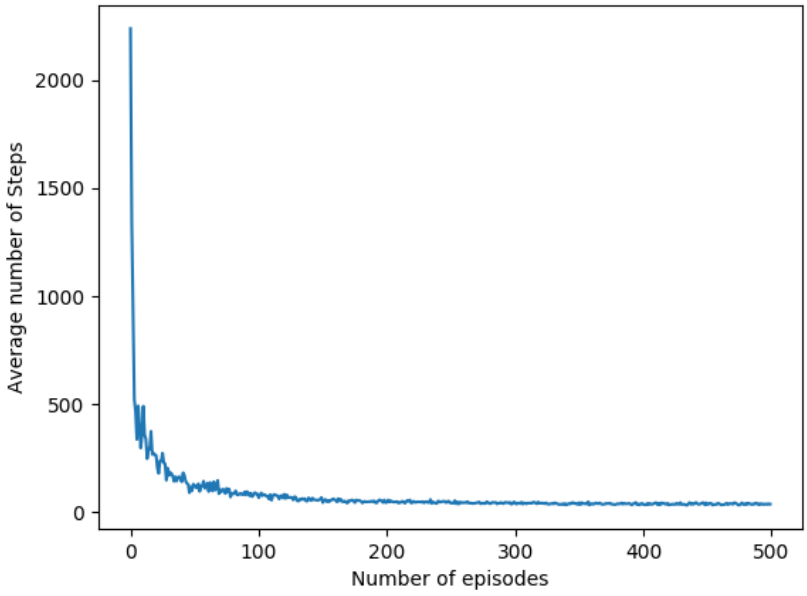

Figure 11: Average number of steps required plotted against the episode number. Averaged over 50 runs

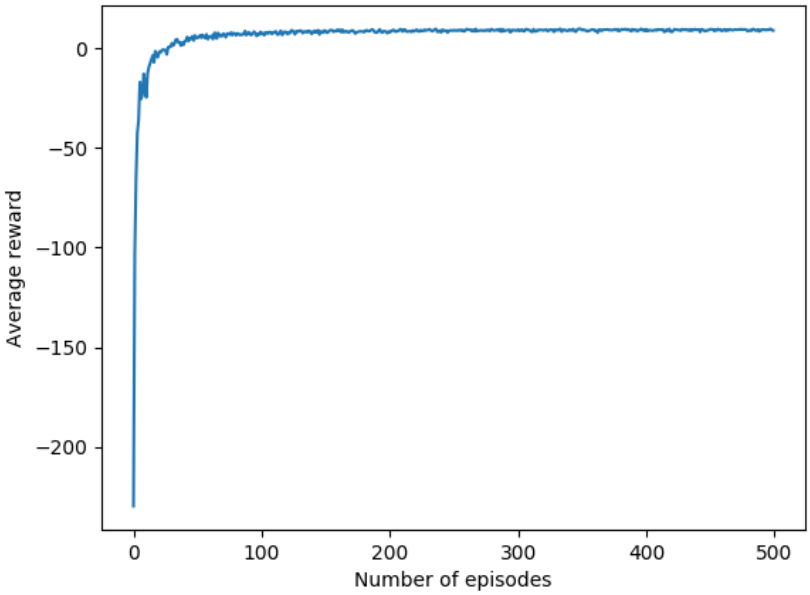

Figure 12: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the visualization of policy learned by SARSA to reach state A.

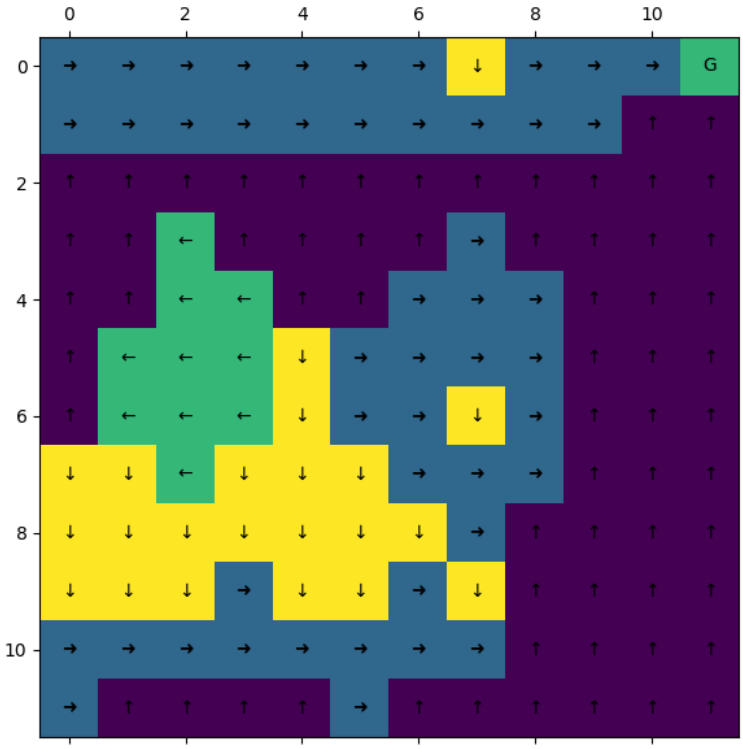

Figure 13: Visualization of the Policy learned by the SARSA. arrow represents the direction of action policy takes at that particular state

2.2.2 Goal B

Similar to Goal state A . SARSA converged at 20 steps and 10 reward.

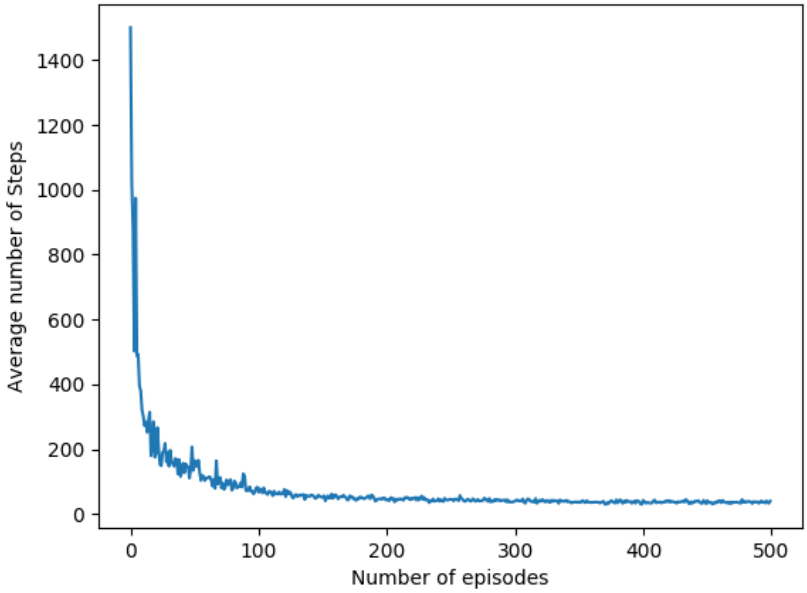

Figure 14: Average number of steps required plotted against the episode number. Averaged over 50 runs

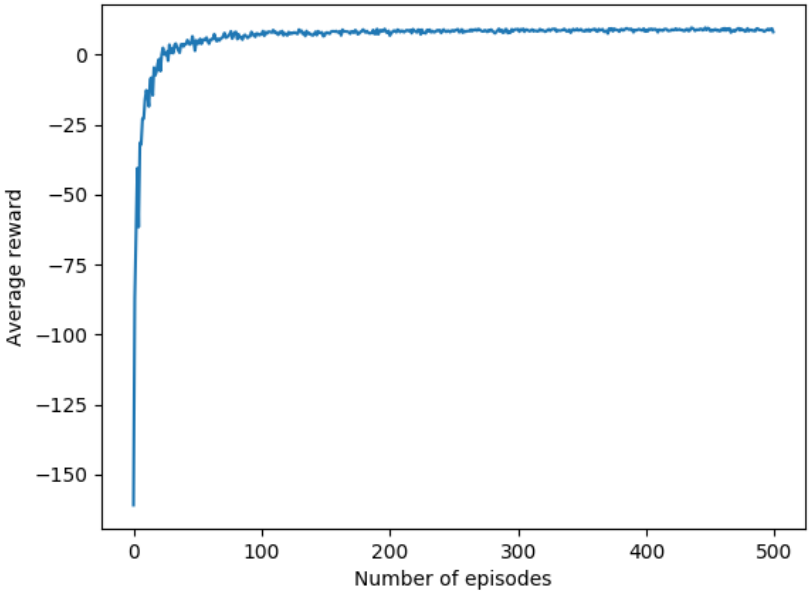

Figure 15: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the visualization of the policy.

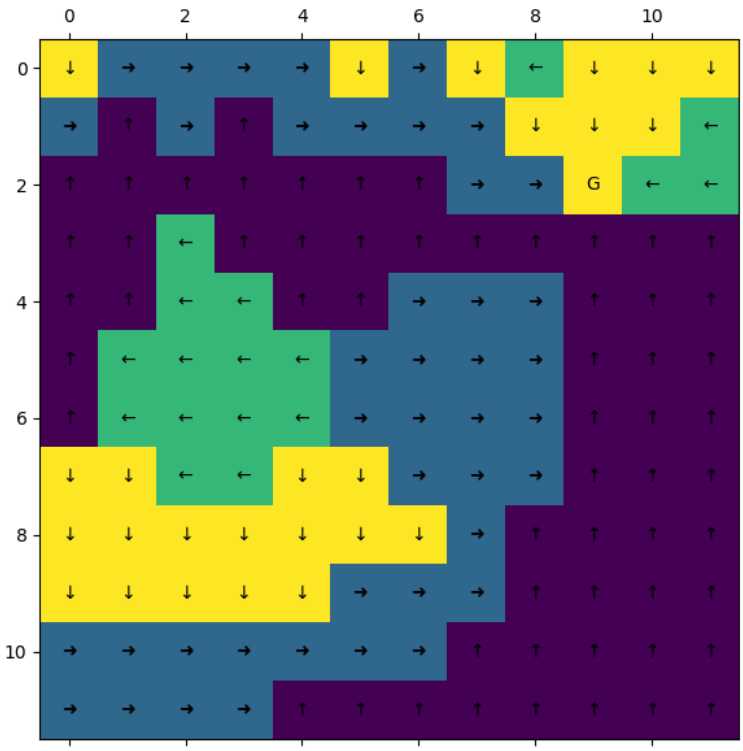

Figure 16: Visualization of the Policy learned by the SARSA. arrow represents the direction of action policy takes at that particular state

2.2.3 Goal C

SARSA to goal state C, Average number of steps converged at 16 . With average reward of 8.9.

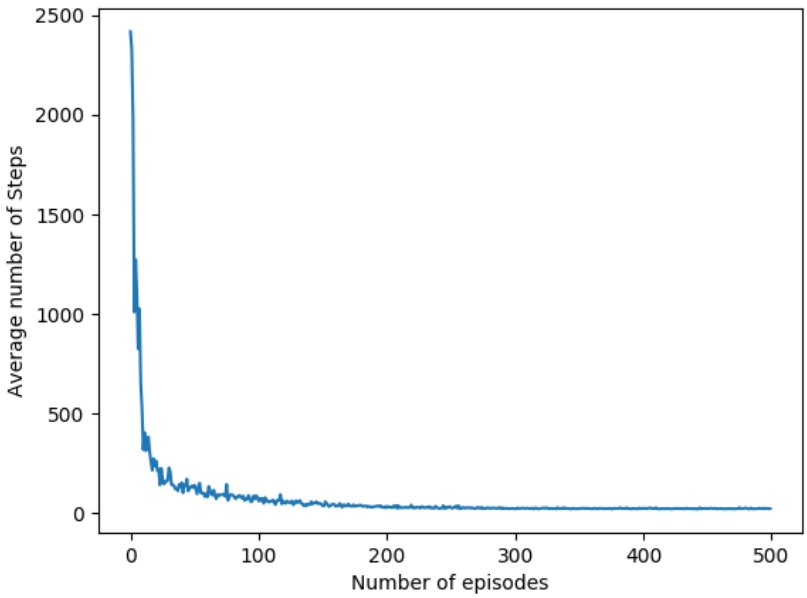

Figure 17: Average number of steps required plotted against the episode number. Averaged over 50 runs

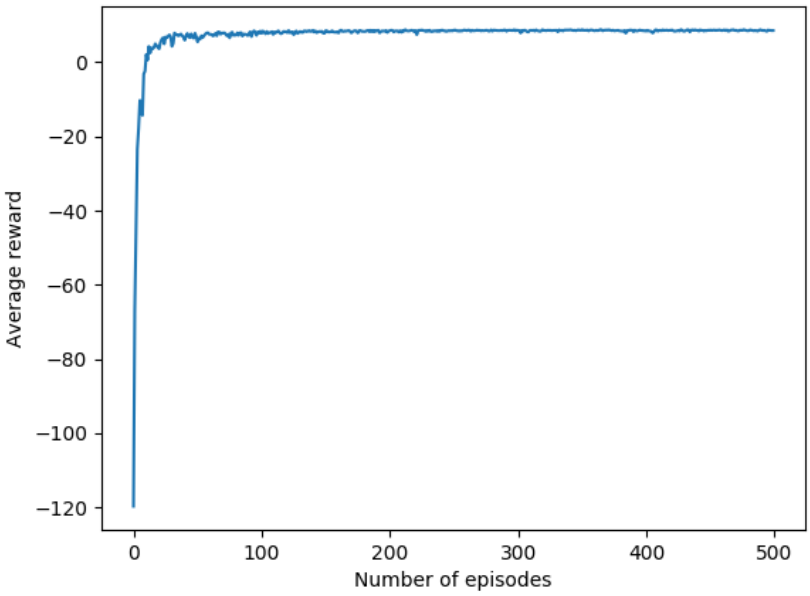

Figure 18: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the visualization of the policy learned by SARSA to reach goal state C.

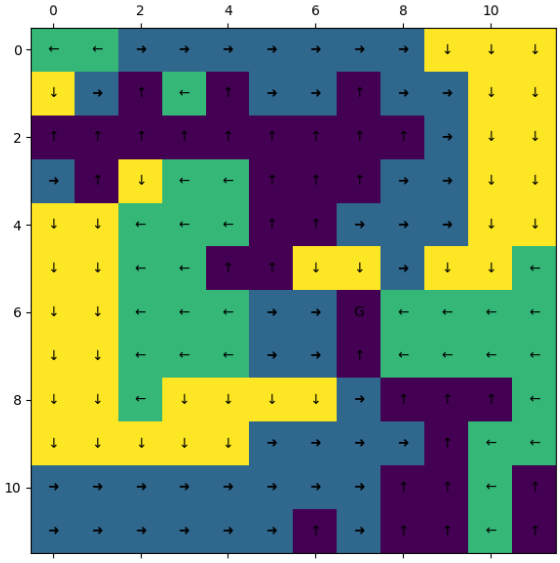

Figure 19: Visualization of the Policy learned by the SARSA. arrow represents the direction of action policy takes at that particular state

2.3 SARSA(λ)

2.3.1 Goal A

The below plot shows the average steps after 25 iterations for different values of λ . For lambda value of 0.5 number of steps seems to go down the best.

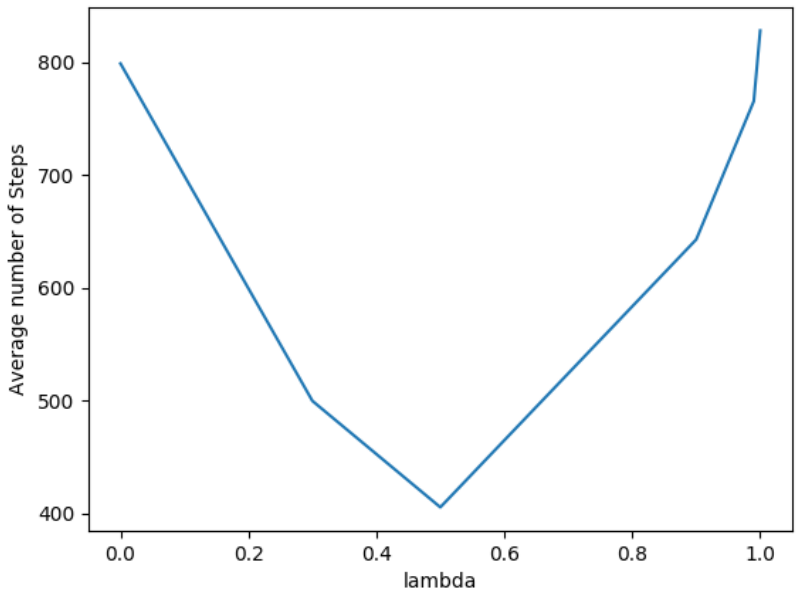

Figure 20: Comparing Average number of steps different values of lambda after 25 iterations of learning policy

Similarly for different values of lambda average reward after 25 iteration is also compared.

Average of 50 runs were taken.

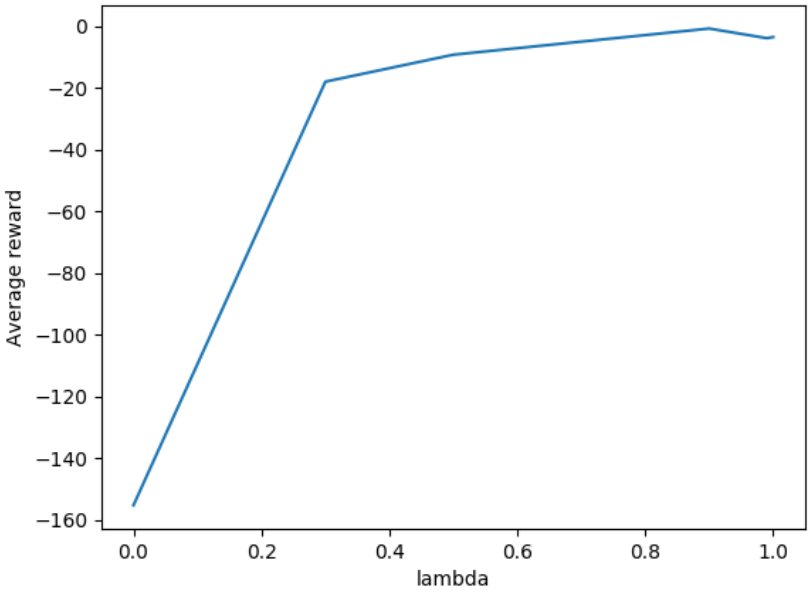

Figure 21: Comparing Average reward different values of lambda after 25 iterations of learning policy

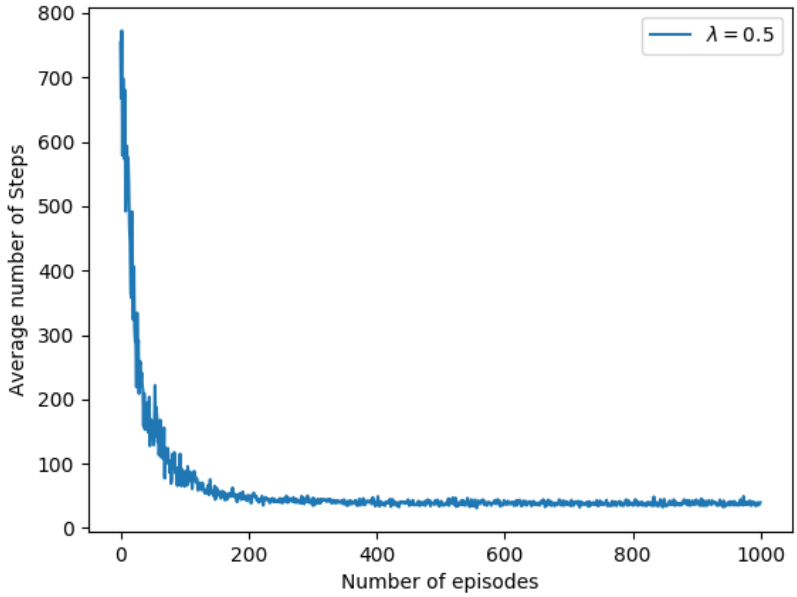

Figure 22: Average number of steps required plotted against the episode number. Averaged over 50 runs

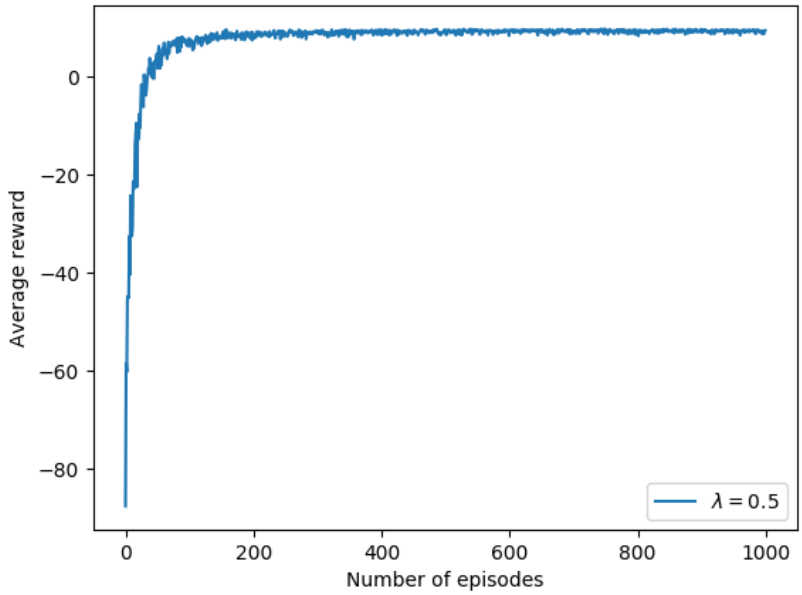

Figure 23: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the visualization of the policy learned by SARSA to reach goal state A.

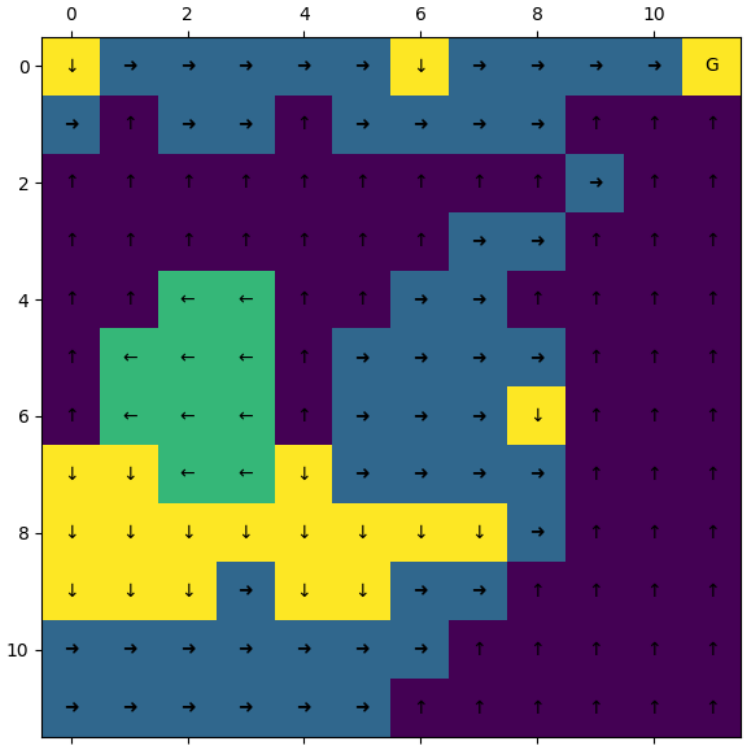

Figure 24: Visualization of the Policy learned by the **SARSA - λ**. arrow represents the direction of action policy takes at that particular state

2.3.2 Goal B

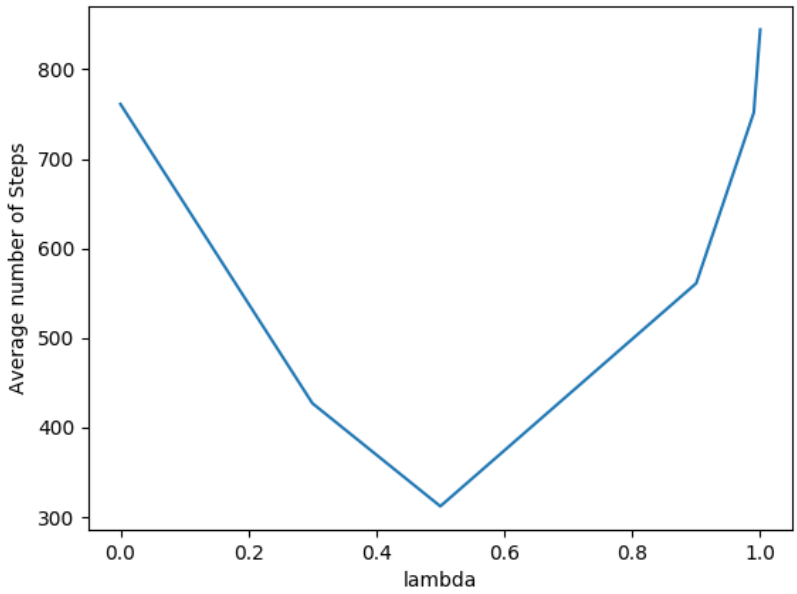

Figure 25: Comparing Average number of steps different values of lambda after 25 iterations of learning policy

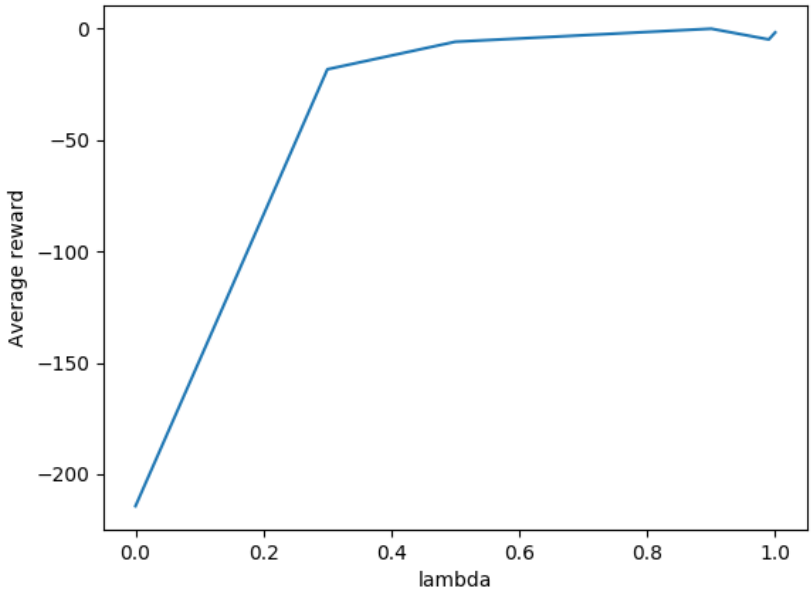

Figure 26: Comparing Average reward different values of lambda after 25 iterations of learning policy

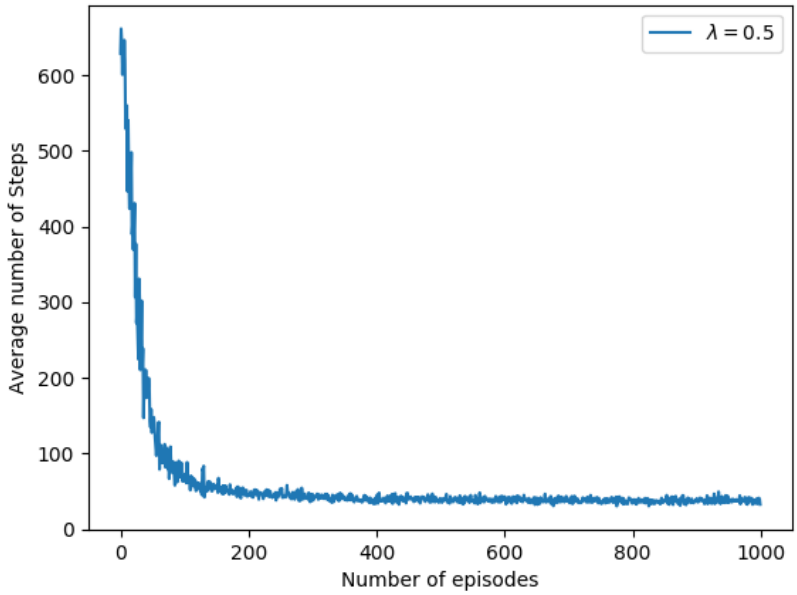

Figure 27: Average number of steps required plotted against the episode number. Averaged over 50 runs

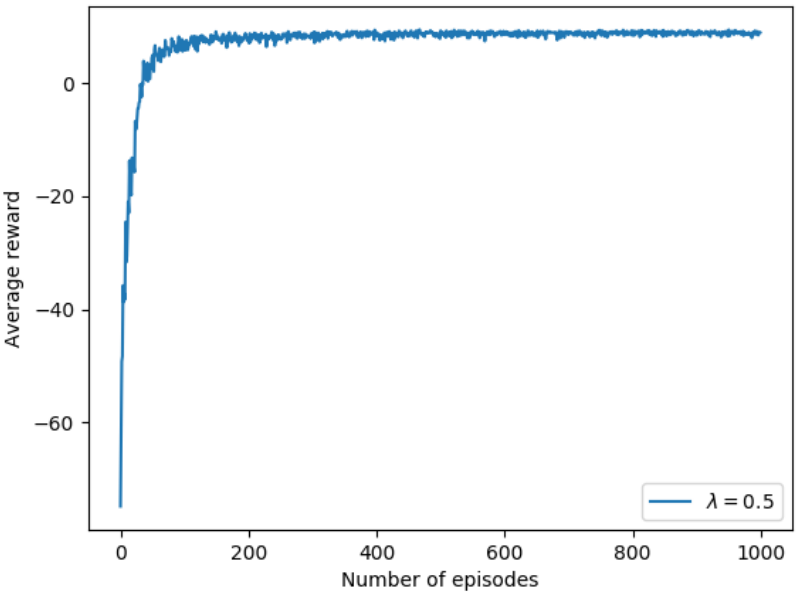

Figure 28: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the visualization of the policy learned by SARSA to reach goal state C.

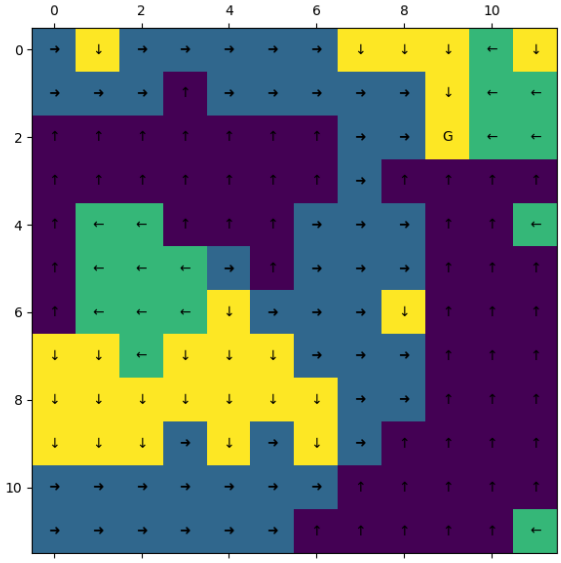

Figure 29: Visualization of the Policy learned by the **SARSA -λ)**. arrow represents the direction of action policy takes at that particular state

2.3.3 Goal C

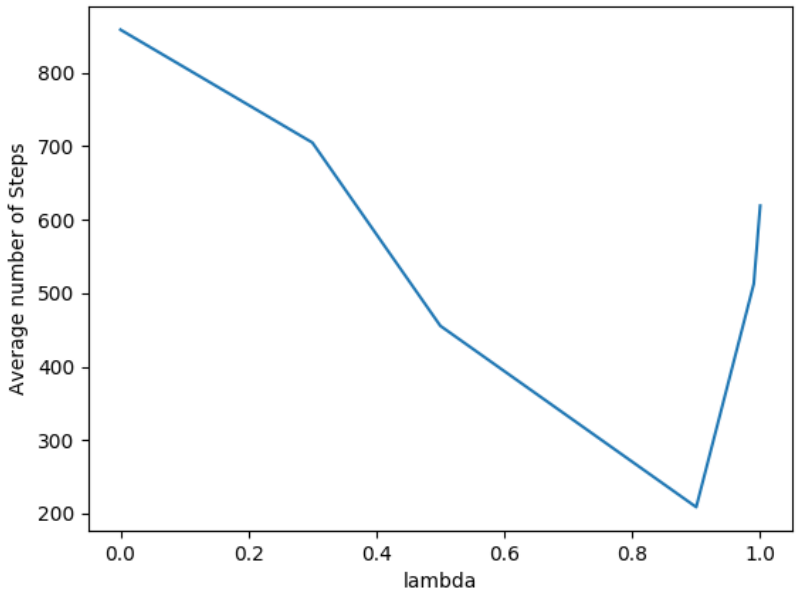

Figure 30: Comparing Average number of steps different values of lambda after 25 iterations of learning policy

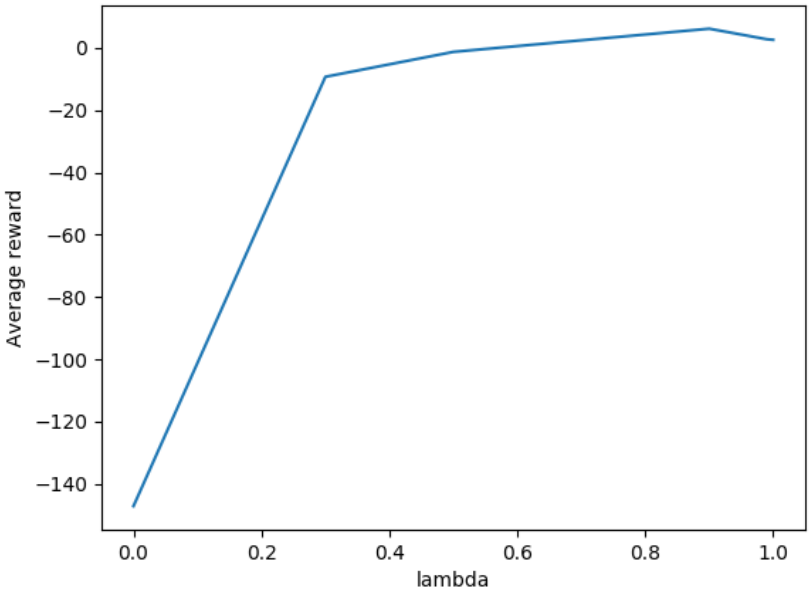

Figure 31: Comparing Average reward different values of lambda after 25 iterations of learning policy

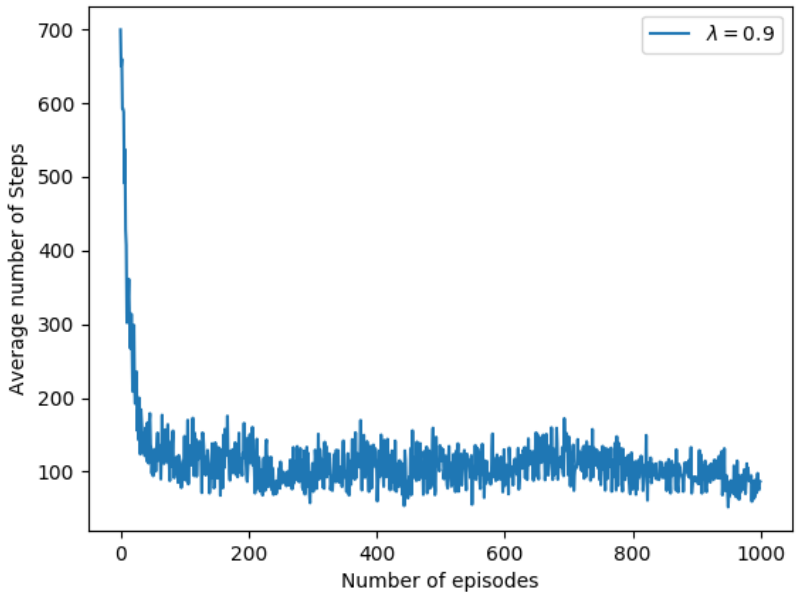

Figure 32: Average number of steps required plotted against the episode number. Averaged over 50 runs

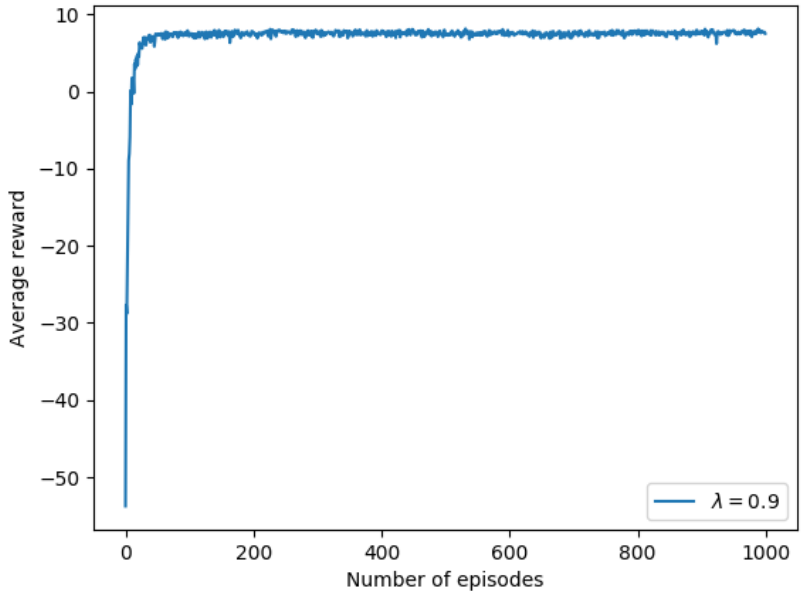

Figure 33: Average Reward collected in a episode plotted against the episode number. Averaged over 50 runs

Below is the visualization of the policy learned by SARSA-λ to reach goal state C.

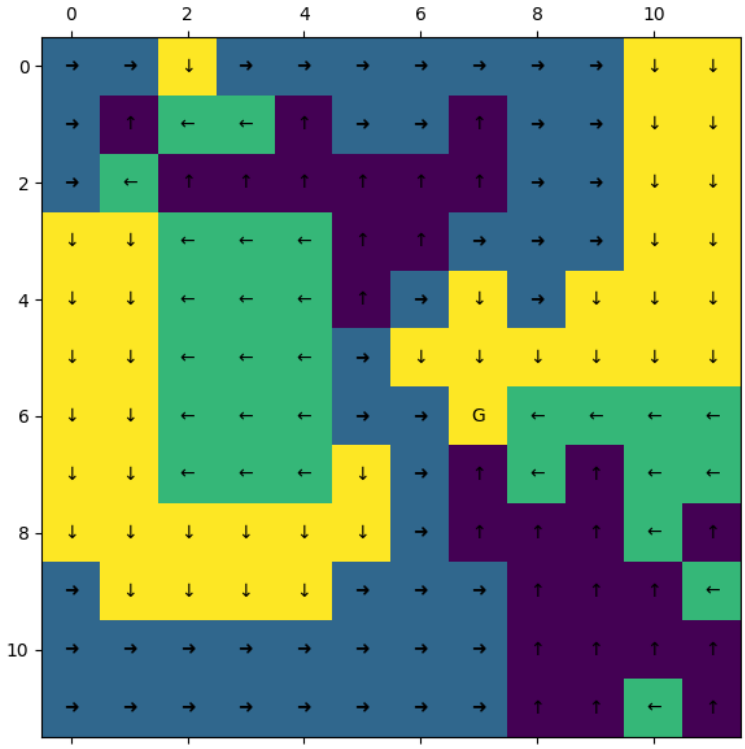

Figure 34: Visualization of the Policy learned by the **SARSA-λ** for gamma 0.9. arrow represents the direction of action policy takes at that particular state

3 Policy Gradients

3.1 Enviornment

The environment was implemented in openAI-gym using the template provided.

Actions continuous clipped between 0.025 and -0.025. Action are sampled according to the policy given as $\pi(s) = \mathcal{N}(\Theta^T S, \Sigma)$ Theta is the parameter for the policy such that $\Theta^T S$ is the mean of the normal distribution and the covariance matrix is identity. Θ is 3x2 and S is 1x3 . Additional dimension is augmented to take care of bias.

Whenever the player leaves the grid trajectory end playes gets a -1000 reward.
Reward at any position is euclidean distance to the center(Goal) for ***Chakra****.*
Reward at a position (x,y) is $0.5x^2 + 0.5y^2$ *for the* ***visham*** *environment*

For the gradient update

$$\nabla_\Theta log(\pi(a|S;\Theta)) = \Sigma^{-1}(a - \Theta^T S)S$$

$$\Sigma = \mathcal{I}$$

Theta is updated in batchwise manner. Gradient is accumulated over one batch of episodes then normalized and used to update Θ.

$$\Theta_{k+1} = \Theta_k + lr \times (\hat{A}(S,a)) \times \nabla_\Theta log(\pi(a|S;\Theta))$$

Advantage function is R_t - baseline(s). Baseline is a function of state and not action. Average discounted return is chosen as baseline.

The optimal policy learned for Chakra is

$$\begin{pmatrix} -1.11297217 & -0.01276917 \\ -0.00677354 & -1.11770226 \\ 0.01022734 & -0.00679685 \end{pmatrix}$$

The optimal policy learned for visham is when gamma is set to 0.9.

$$\begin{pmatrix} -2.00862838 & 0.00980043 \\ 0.0105143 & -2.02697878 \\ 0.0095315 & -0.00562391 \end{pmatrix}$$

The optimal policy learned for visham is when gamma is set to 0.01.

$$\big([[-1.30811914 - 0.02768532 - 0.00503418][0.0284237 - 0.77274371 - 0.02883167]]\big)$$

Below are the plots for of average number of steps needed to reach the goal. First Plot of for Chakra and second is for Visham.

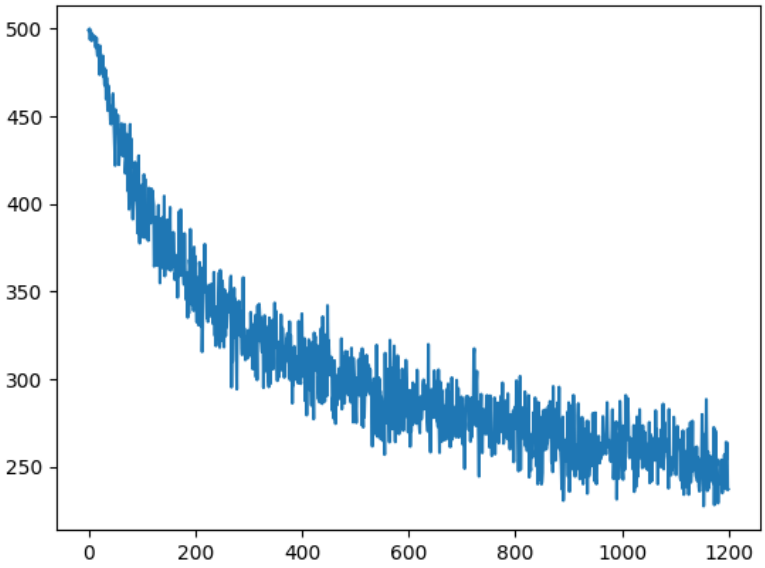

Figure 35: The above plot shows the number of steps in an episode for **Chakra** environment

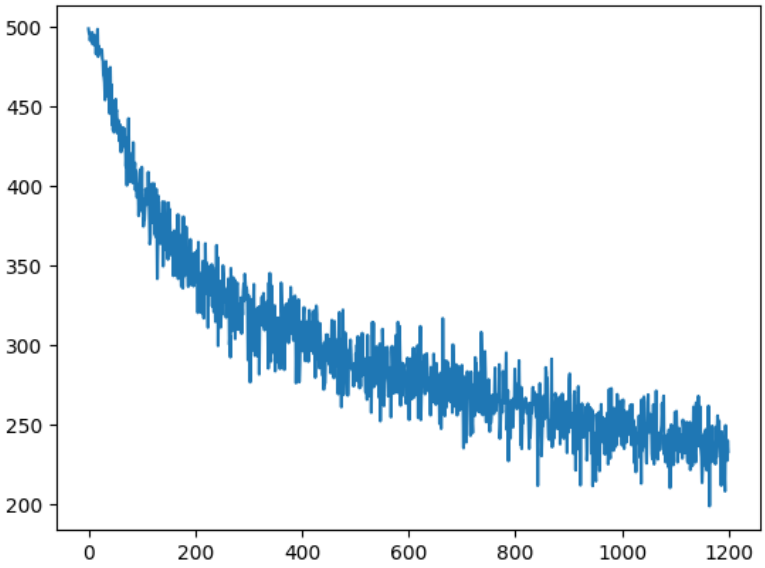

Figure 36: The above plot shows the number of steps in an episode for **Visham** environment

3.2 Questions and Deliverables

Carry out hyperparameter tuning: Tune the hyperparameters, like batch size, learning rate, discount factor, etc. What do you observe? Can you learn a policy faster?

In the discussion below it other hyper-parameters are kept fixed. **Batch Size** As the Batch size was decreased the the number of iterations required increases. But the total number of episode required remained almost same to arrive at the same policy. With smaller batch size at the theta (policy parameter) get updated in fewer number of episode therefore future episodes take less time

to end. But if the batch-size too small the gradient keep oscillating. Batch was selected to be 50 episodes.

Learning Rate : With Learning Rate kept very small takes very long time to arrive at the same Θ as in case of optimal Learning Rate. Increasing Learning Rate results in gradients oscillating and Θ getting stuck at non optimal values. Decaying Learning Rate also did not provide any advantage. Final learning rate of 10^{-2} was used.

Discount Factor : Tuning Discount Factor played more important role in Visham as the Discount factor was brougth closer to 0 policy became more elliptical. At higher Discount factor it looks more future rewards with changing the policy hence not looking at the immediate rewards. In Visham case Without any discounting and infinitesimally small step will lead to policy with action normal to the elliptical reward function. If we increase Discount Factor it more and more towards Chakra.

To make the learning faster can be made imposing -100 reward on going out of the frame this improved the learning speed. Adding positive reward on reaching the goal did not had the same effect.

Can you visualize a rough value function for the learned policy? If no, too bad. If yes, how would you go about it? Turn in the plots.

Value Function plotted below is computed using TD(0) update using state aggregation. 100 x 100 grid was used for aggregation

$$V(S) = V(S') + \alpha(R + \gamma V(S') - V(S))$$

Below are the plots of value function for Chakra. Value Function for chakra is a bowl shaped.

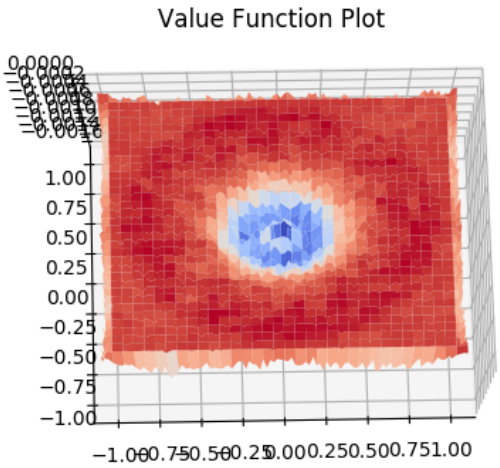

Figure 37: Value Function of Chakra

Below are the plots for the value function of Visham. First one is with value of gamma 0.9 and second is the value of 0.01.

Value function when gamma is 0.9 it resembles with chakra but when we decrease gamma to 0.01 value function becomes more eliptical.

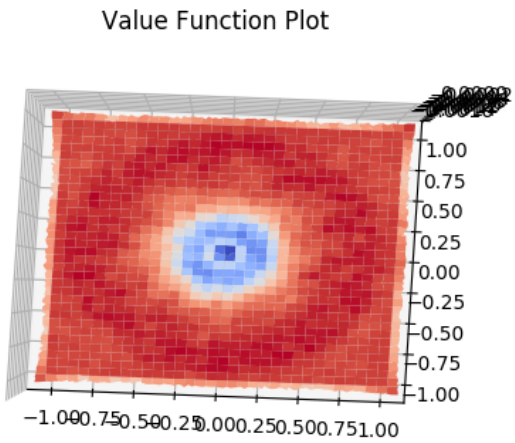

Figure 38: Value Function of Visham with gamma 0.9

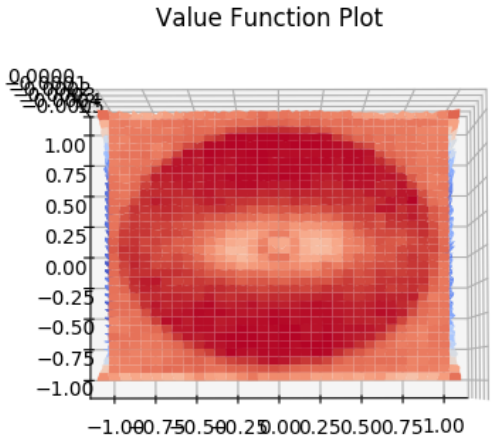

Figure 39: Value Function of Visham with gamma 0.01

What do you observe when you try plotting the policy trajectories for the learned agent? Is there a pattern in how the agent moves to reach the origin?

Below is the plot of $\Theta^T S$ over the given grid . For the case of chakra policy as any state S is to take action in the direction of origin or normal to the value function contour.

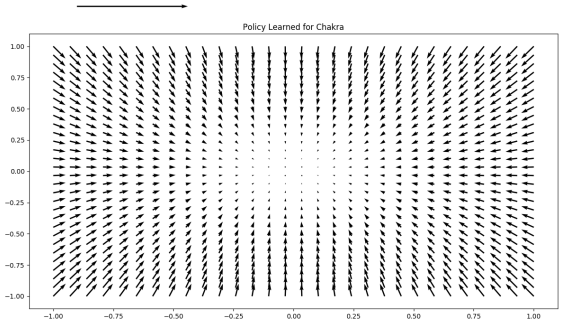

Figure 40: Policy visualization of chakra with gamma 0.9

For the case of visham for discounting of 0.9 the policy learned considers future rewards and hence it moves the same as chakra , However as we decrease the gamma policy starts to minimize cost immediately policy starts to look more eliptical and moves close to Visham value contour. (Although reward in distributed with ratio of major axis to minor is $\sqrt{10}$ where as learned policy is only the ratio is 2)

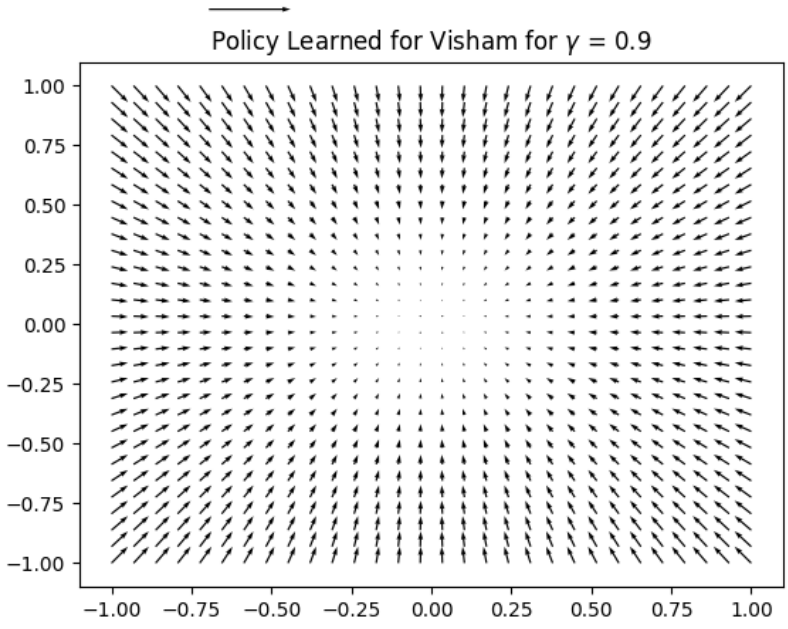

Figure 41: Policy visualization of Visham with gamma 0.9

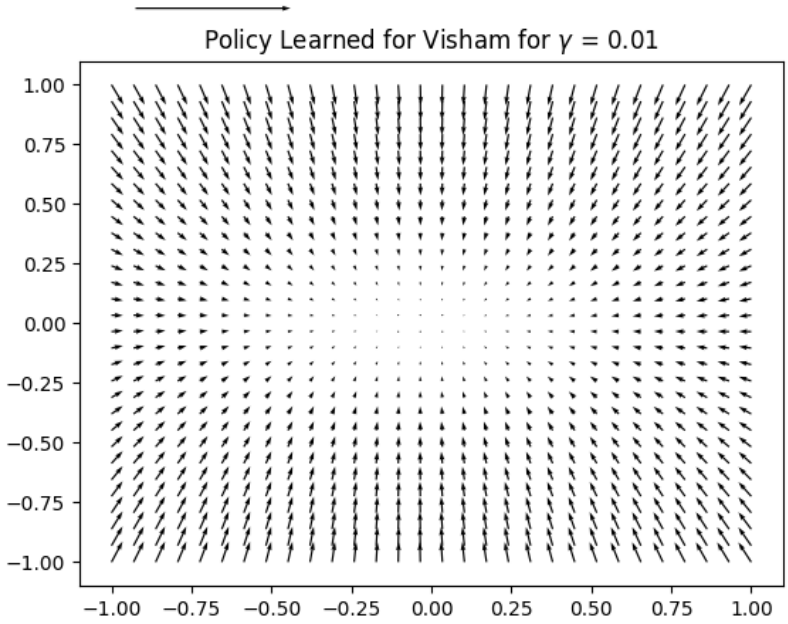

Figure 42: Policy visualization Visham with gamma 0.01

Below are the links for the video of trajectory

- Chakra
- Visham for gamma 0.9
- Visham for gamma 0.01