

Indian Institute of Technology, Madras
CS6700: Reinforcement Learning
Reinforcement Learning Assignment-I Report

Rajan Kumar Soni - CS18S038

March 2020

Contents

1	SARSA	5
1.1	Implementation of Sarsa for Goal A and related plots	5
1.1.1	Related plots	5
1.2	Implementation of Sarsa for Goal B and related plots	6
1.2.1	Related plots	6
1.3	Implementation of Sarsa for Goal C and related plots	7
1.3.1	Related plots	7
1.4	Inference from Observation for SARSA	8
2	Q LEARNING	9
2.1	Implementation of Qlearning for Goal A and related plots	9
2.1.1	Related plots	9
2.2	Implementation of Qlearning for Goal B and related plots	10
2.2.1	Related plots	10
2.3	Implementation of Qlearning for Goal C and related plots	11
2.3.1	Related plots	11
2.4	Inference from Observation for Q learning	12
3	SARSA LAMBDA	13
3.1	Implementation of Sarsalambda for Goal A and related plots	13
3.1.1	Related plots	13
3.2	Implementation of Sarsalambda for Goal B and related plots	14
3.2.1	Related plots	14
3.3	Implementation of Sarsalambda for Goal C and related plots	15
3.3.1	Related plots	15

3.4	Implementation of Sarsa lambda for different lambdas for Goal A and related plots .	16
3.4.1	Related plots	16
3.5	Implementation of Sarsa lambda for different lambdas for Goal B and related plots .	19
3.5.1	Related plots	19
3.6	Inference from Observation for Sarsa Lambda	21
4	Policy Gradient	23
4.1	Chakra	23
4.1.1	Hyper parameter tuning	23
4.1.2	Value function visualisation	24
4.1.3	Trajectory and policy	25
4.1.4	Inference from observation	26
4.2	Visahmc	27
4.2.1	Hyper parameter tuning	27
4.2.2	Value function visualisation	28
4.2.3	Trajectory and policy	29
4.2.4	Inference from observation	30

Some common environment setup and observation for Puddle world

- Here wind is blowing so only the best action agent will take from that state. Next state depends on the action taken and the wind direction.
- Here we can see that every state does not learn best action. It happened because every time we are starting from the one of the four state and if the agent finds some of the best path then it does not explore more because they might not be reducing the total reward.
- In below figure you can see staircase structure, this is because I started with $\epsilon = 1$ means for first 500 episodes it randomly selects the action(it explores fully) after that step by step I decrease the epsilon value. Here point to be noted that environment is also helping in exploration.
- learning rate is fixed which is 0.5 through out the experiment.
- number of episodes taken 5000
- gamma value = .95

1 SARSA

1.1 Implementation of Sarsa for Goal A and related plots

1.1.1 Related plots

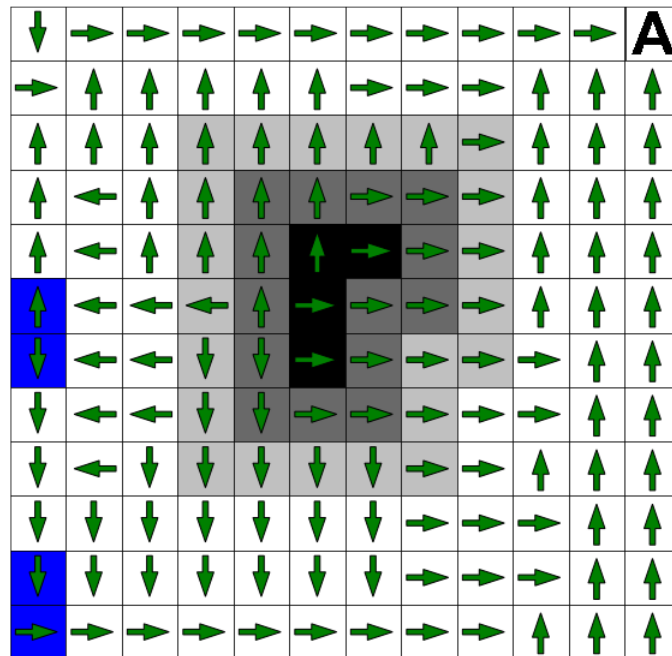
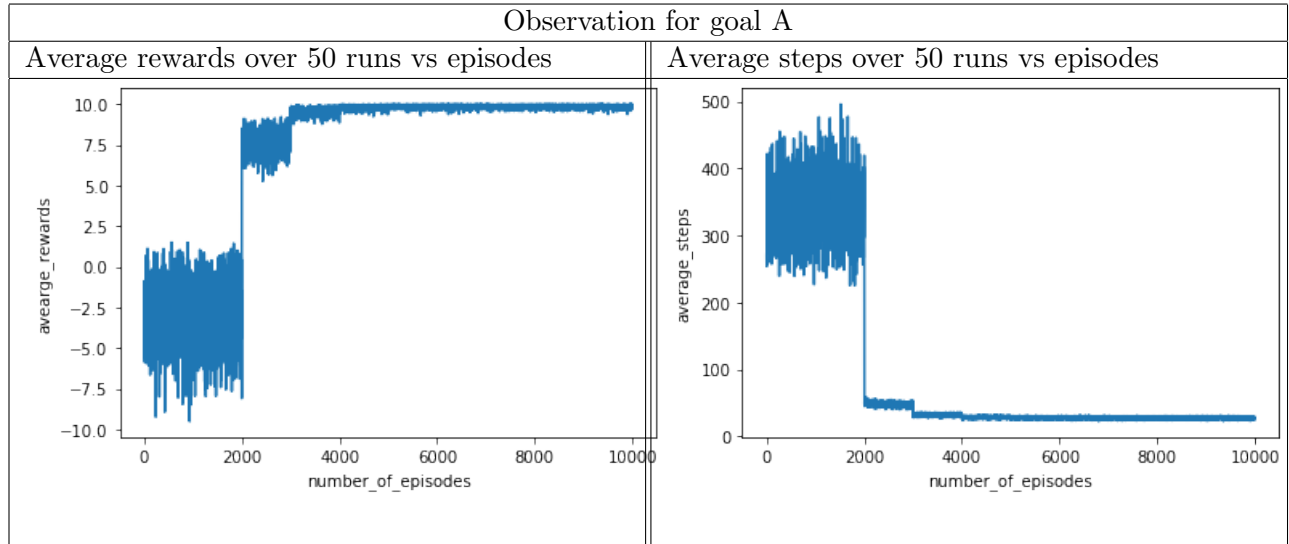


Figure 1: Policy learned

1.2 Implementation of Sarsa for Goal B and related plots

1.2.1 Related plots

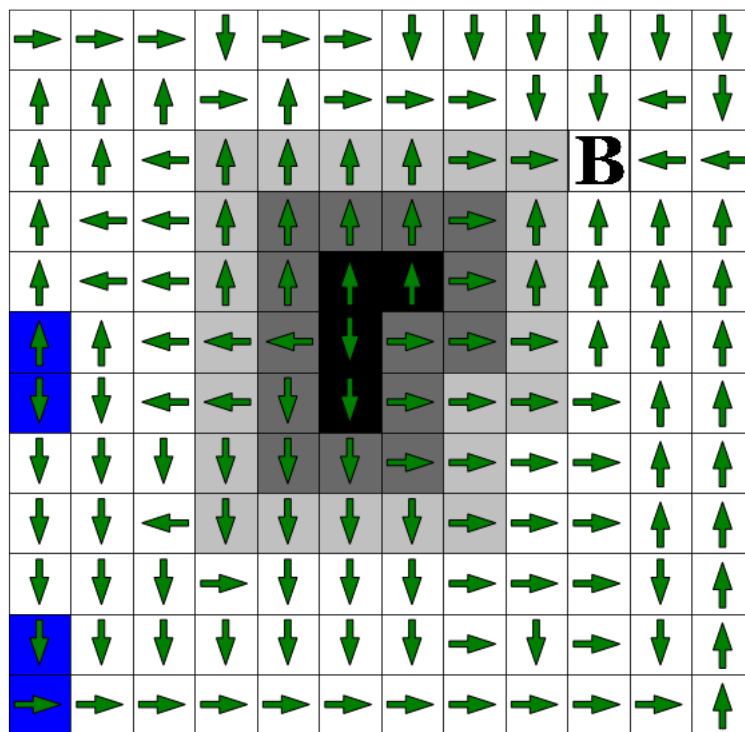
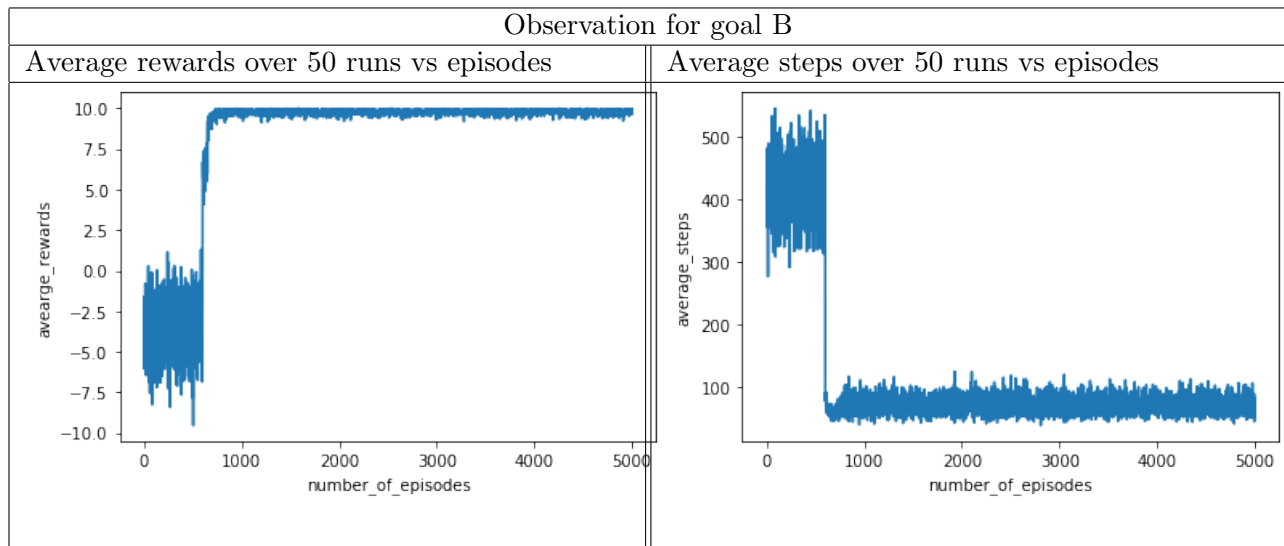


Figure 2: Policy learned

1.3 Implementation of Sarsa for Goal C and related plots

1.3.1 Related plots

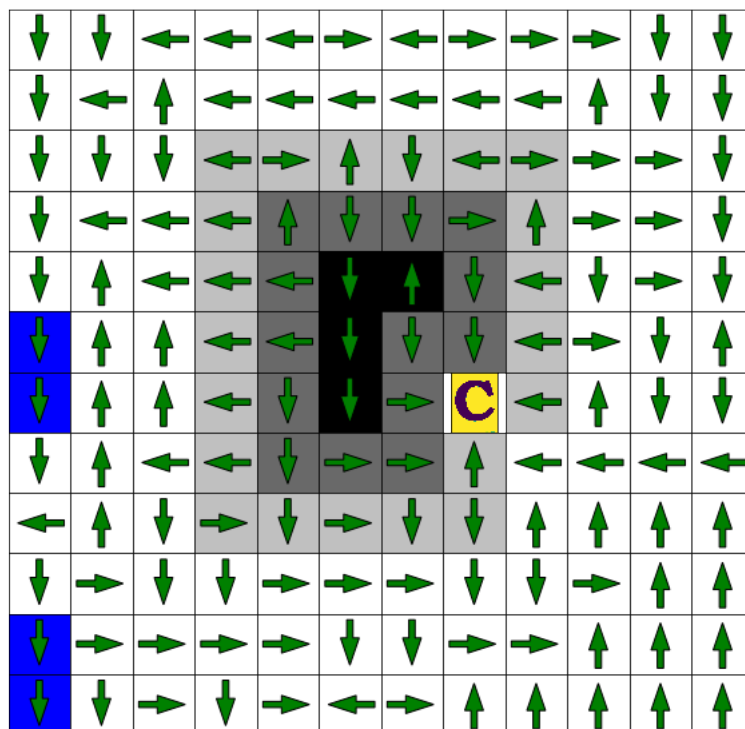
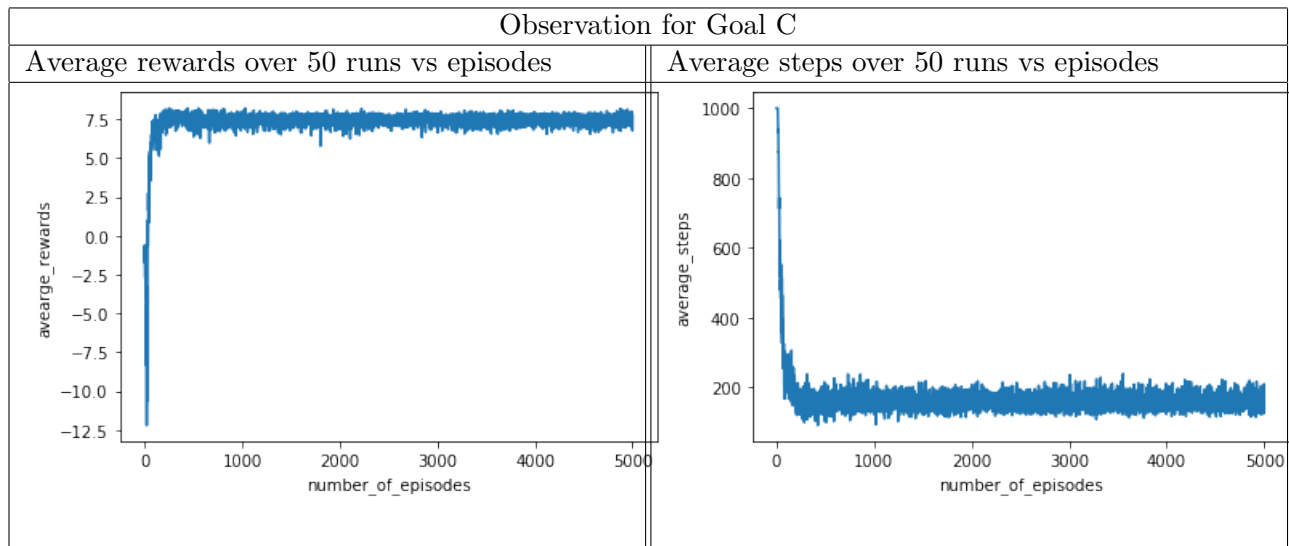


Figure 3: Policy learned

1.4 Inference from Observation for SARSA

Referring to the above figure

- Average steps for goal A is 25
- Average steps for goal B is 29
- Average steps for goal C is 129
- As we know that reaching the goal agent gets 10 reward.
- **Average reward :** As we know that goal A is at the safest place and Sarsa learns by avoiding hazard so learning for goal A agent incurs more average reward. C performs worst because agent has to go in the puddle to reach the goal. But before reaching the goal agent also has to face negative rewards which agents wants to avoid. But because the environment non-deterministic with some probability and agent is also exploring so it reaches the goal. But average reward is less because goal itself is in puddle.
- **Average steps :** As we can see form the plot and and the result we got, average step increases as the goal keeps coming near to puddle word. Because using sarsa algo, agent tries to be as far as possible from the puddle world which leads to increase in average step size.
- **Policy :** As it is clear form the average steps taken to reach the goal. As the goal moves towards puddle average steps increases, it means learnt policy is not optimal. As we can see from the plot all the states does not learn the optimal policy. On of the reason might me possible that agent might not have visited that state that many number of times.
- it converges to good number of steps

2 Q LEARNING

2.1 Implementation of Qlearning for Goal A and related plots

2.1.1 Related plots

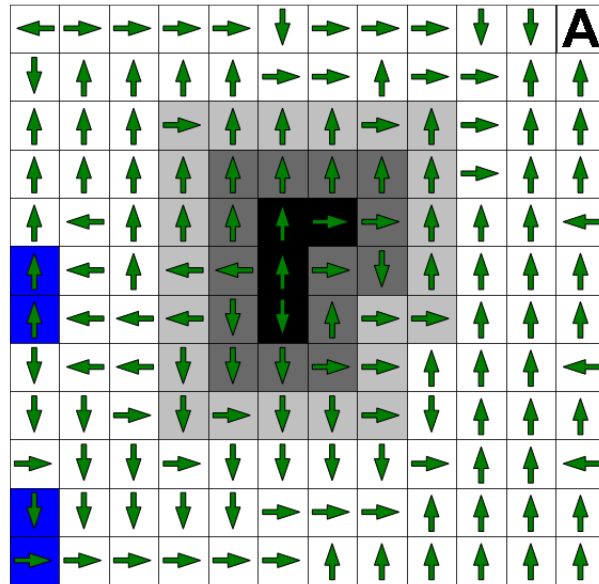
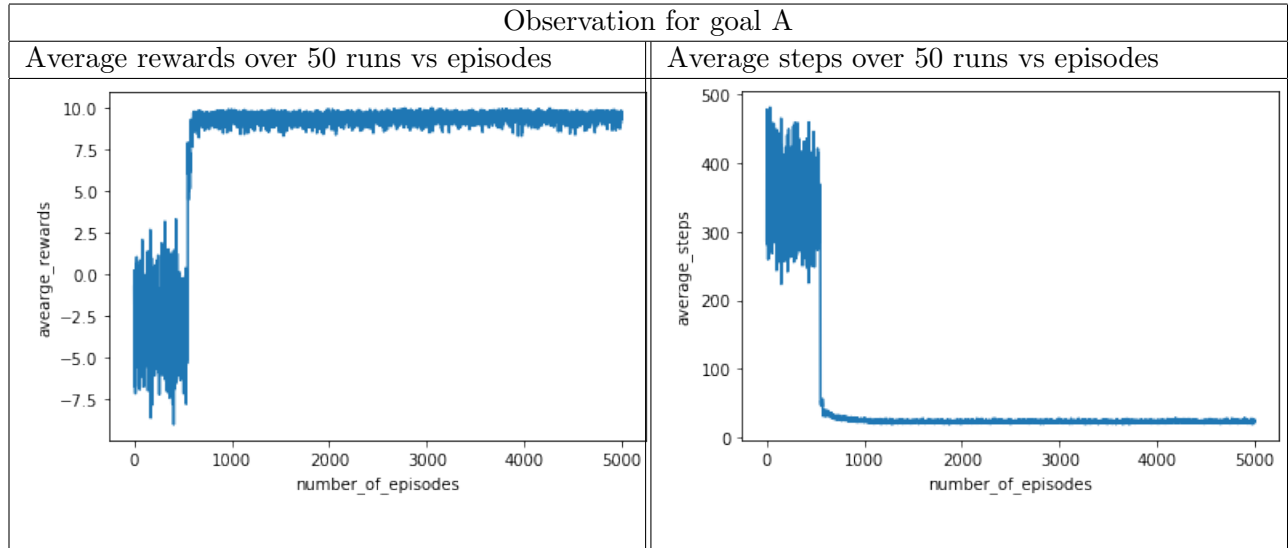


Figure 4: Policy learned

2.2 Implementation of Qlearning for Goal B and related plots

2.2.1 Related plots

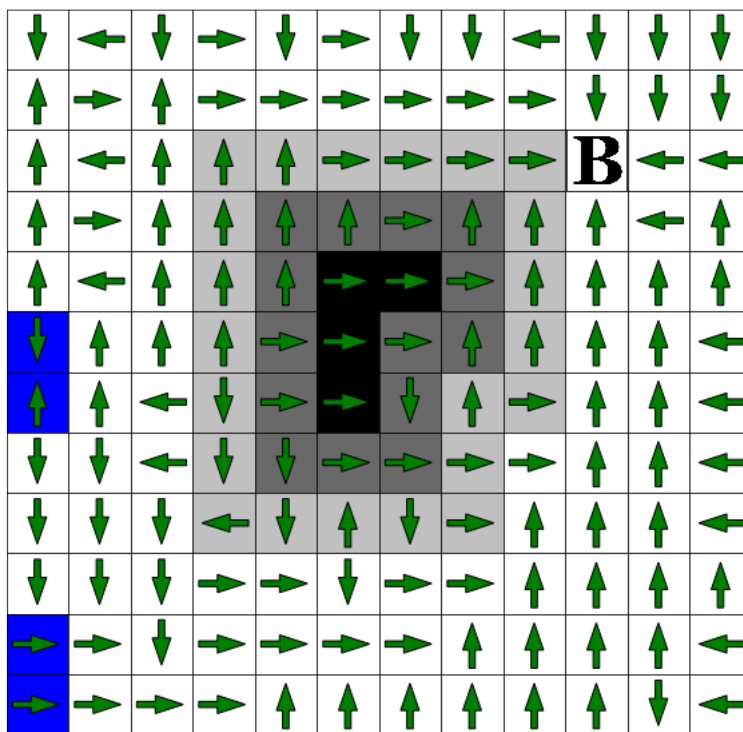
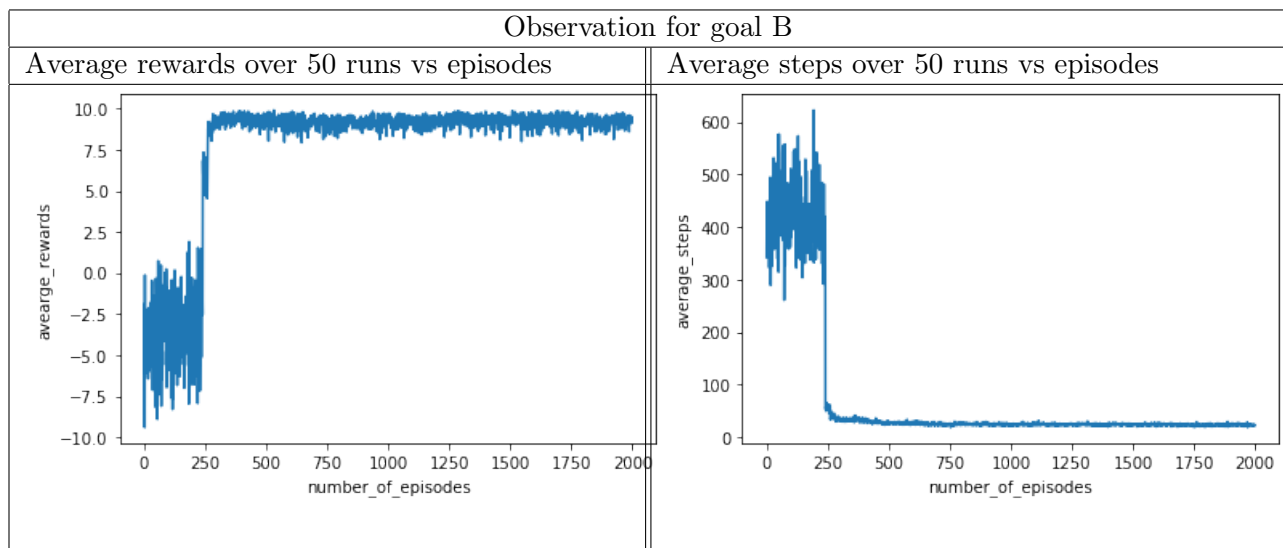


Figure 5: Policy learned

2.3 Implementation of Qlearning for Goal C and related plots

2.3.1 Related plots

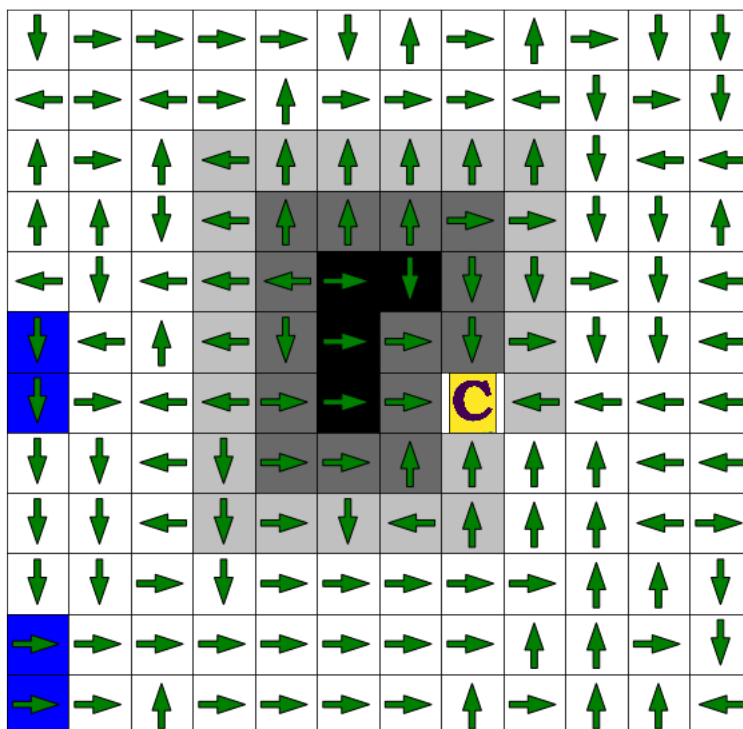
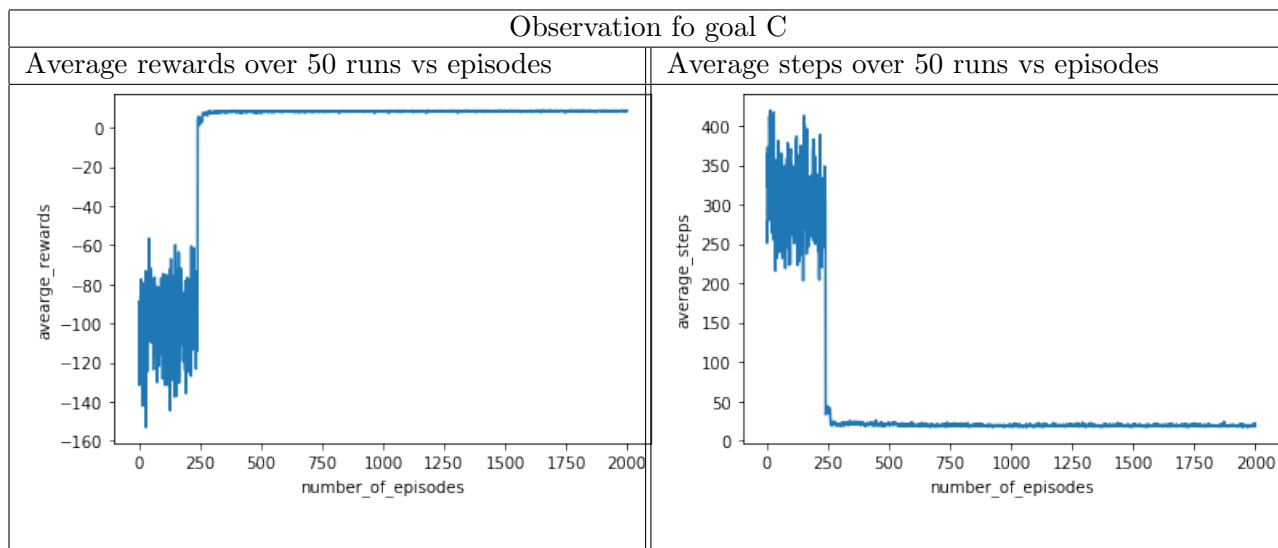


Figure 6: Policy learned

2.4 Inference from Observation for Q learning

Referring from the above figure:

- Average steps for goal A is 24
- Average steps for goal B is 22
- Average steps for goal C is 19
- As we know that reaching the goal agent gets 10 reward.
- **Average reward :** As we know that q learning is an off policy learning it moves in the environment according to some behaviour policy. And learning policy takes the action according to q^* As we know that goal A is at the safest place and q learning learns for goal A incurs more average reward. C performs good here because agent has to go in the puddle to reach the goal. Because it is a q learning agent does not avoid hazard . But before reaching the goal agent also has to face negative rewards. But because the environment is non-deterministic with some probability and agent is also exploring so it reaches the goal. But average reward is less because goal itself is in puddle.
- **Average steps :** As we can see from the plot and the result we got, average step decreases as the goal keeps coming near to puddle word. Because using q learning algo, agent does not hesitate to move inside the puddle.
- **Policy :** As it is clear from the average steps taken to reach the goal. As the goal moves towards puddle average steps decrease, it means learnt policy is not optimal. As we can see from the plot all the states does not learn the optimal policy. One of the reasons might be possible that agent might not have visited that state that many number of times.
- it converges to good number of steps

3 SARSA LAMBDA

3.1 Implementation of Sarsalambda for Goal A and related plots

3.1.1 Related plots

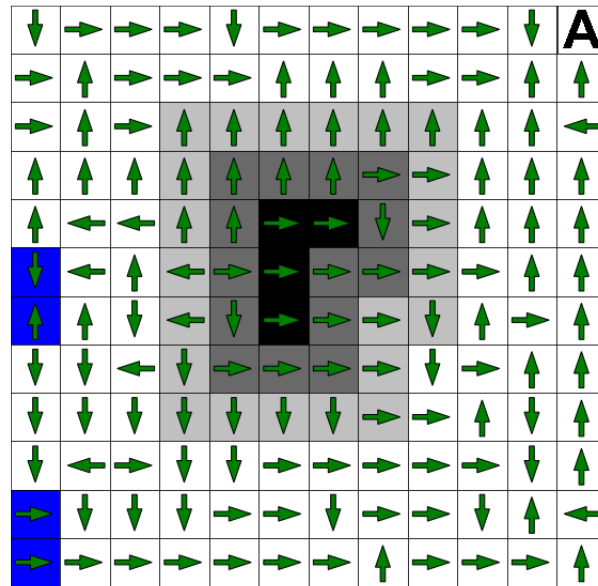
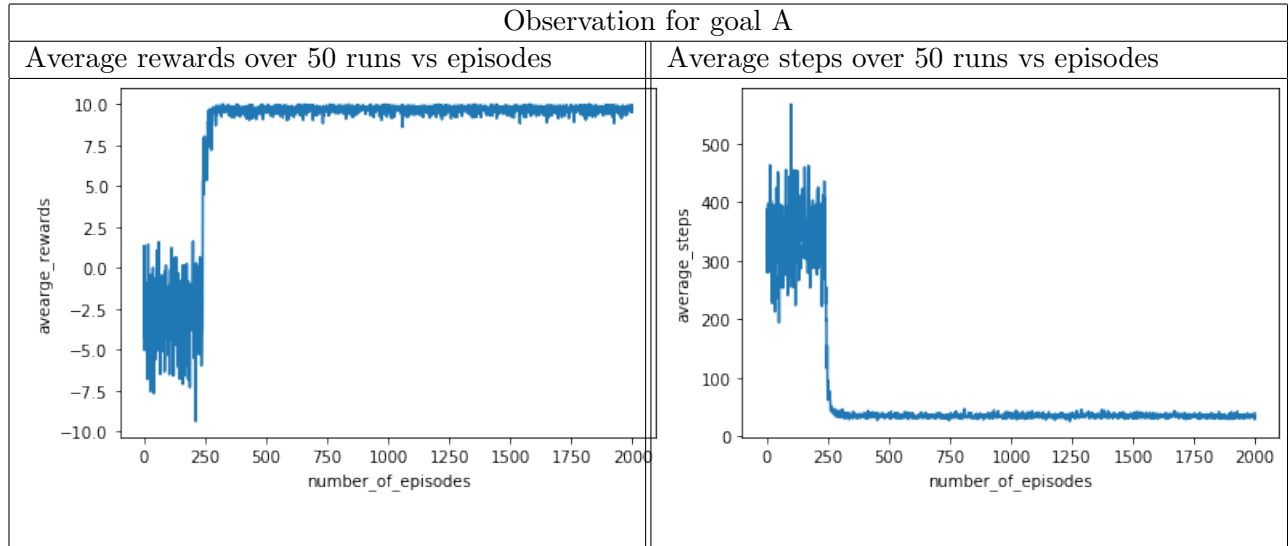


Figure 7: Policy learned

3.2 Implementation of Sarsalambda for Goal B and related plots

3.2.1 Related plots

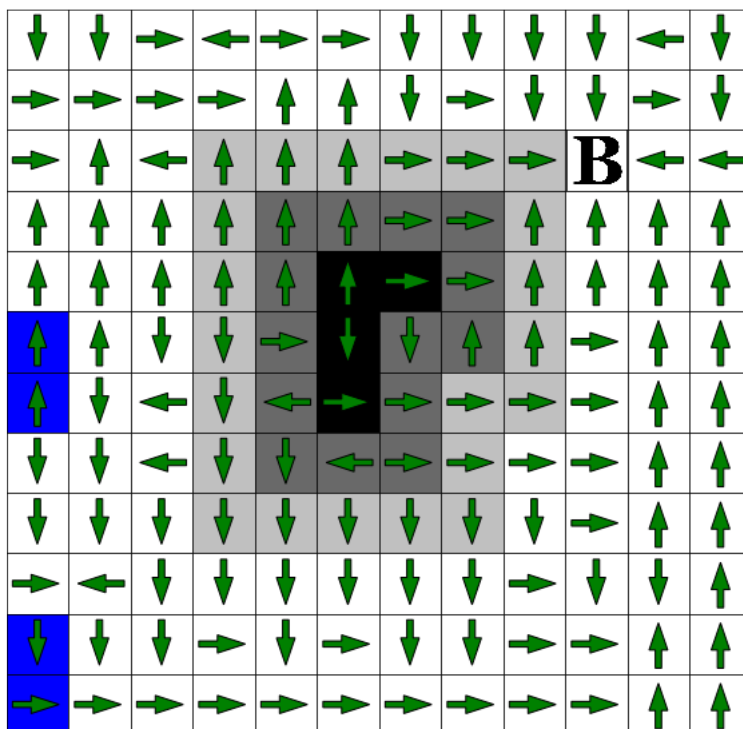
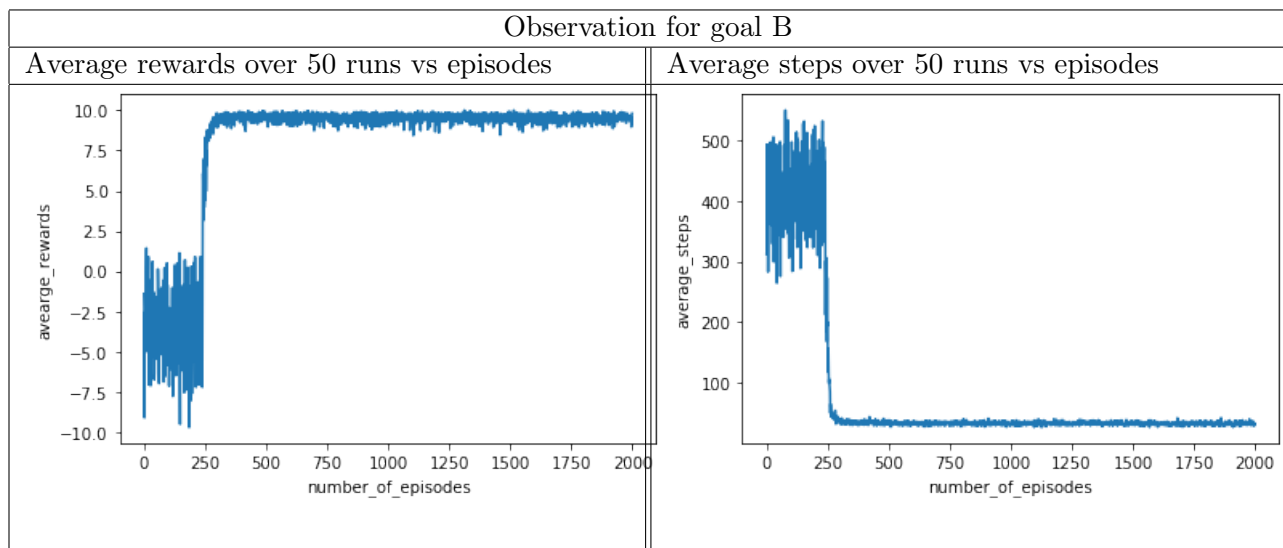


Figure 8: Policy learned

3.3 Implementation of Sarsalambda for Goal C and related plots

3.3.1 Related plots

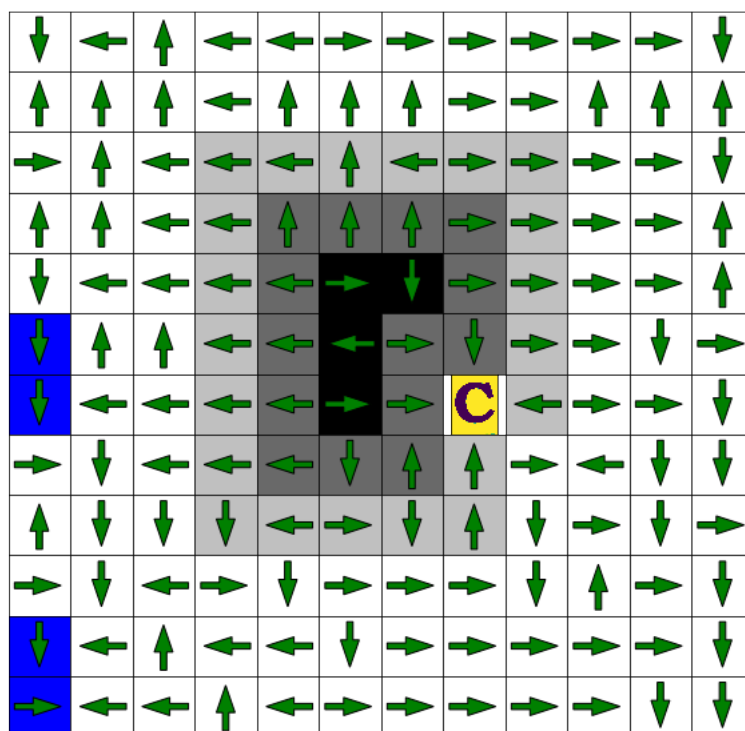
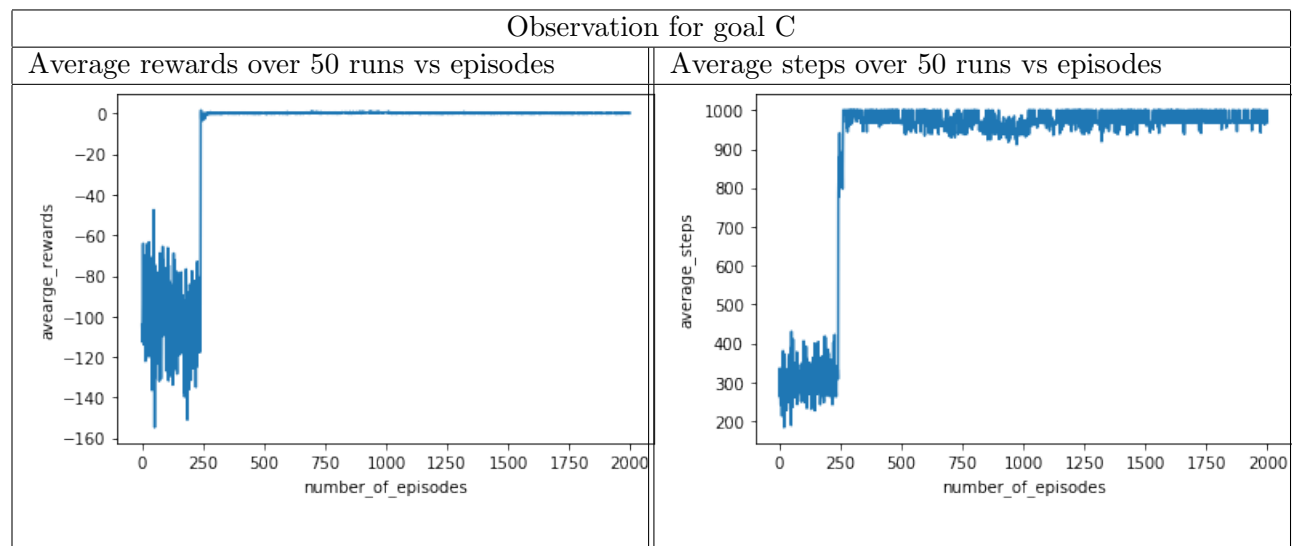


Figure 9: Policy learned

3.4 Implementation of Sarsa lambda for different lambdas for Goal A and related plots

3.4.1 Related plots

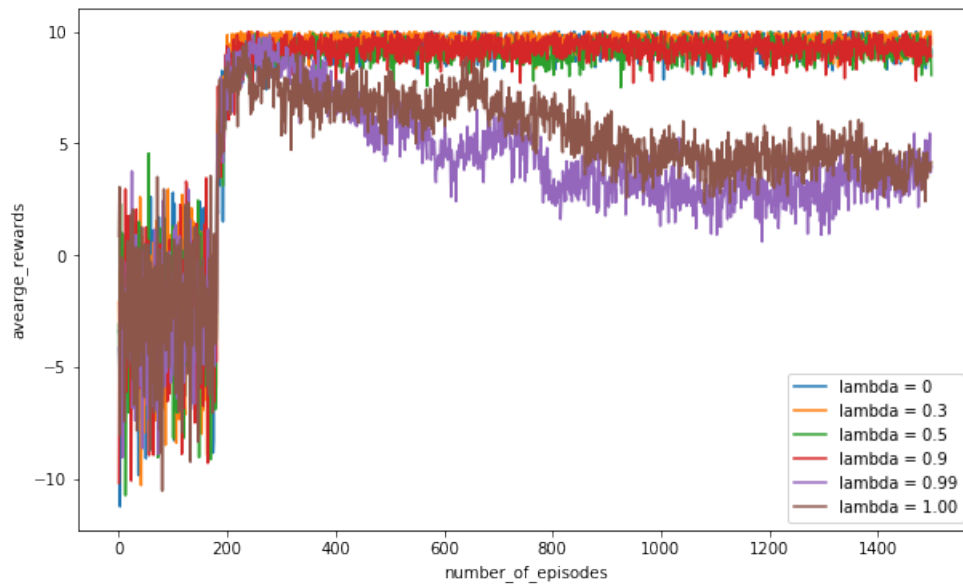


Figure 10: Average rewards over 50 runs vs episodes

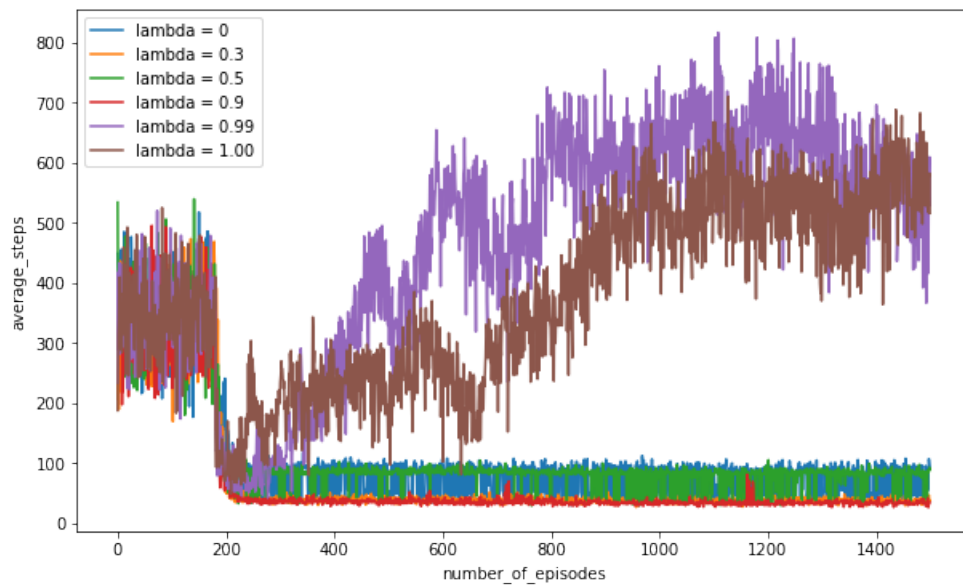
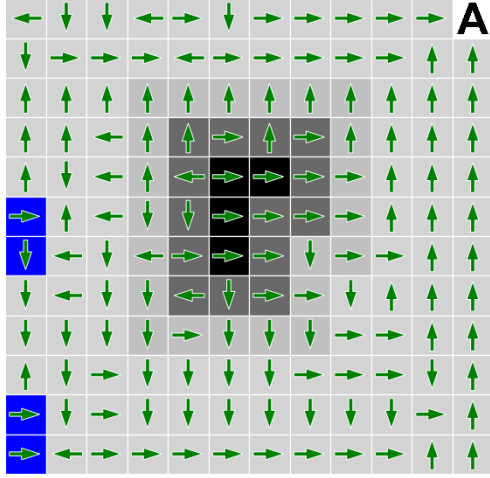


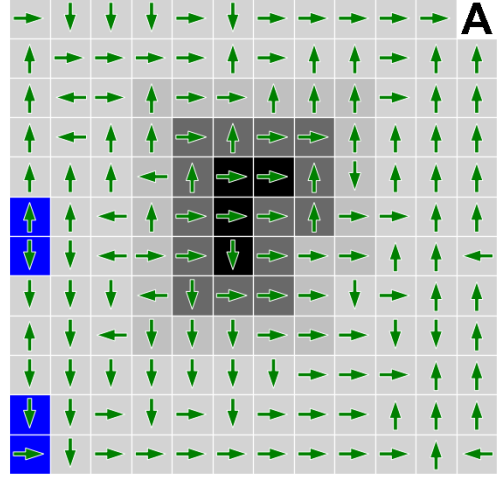
Figure 11: Average steps over 50 runs vs episodes

For Goal A Observation

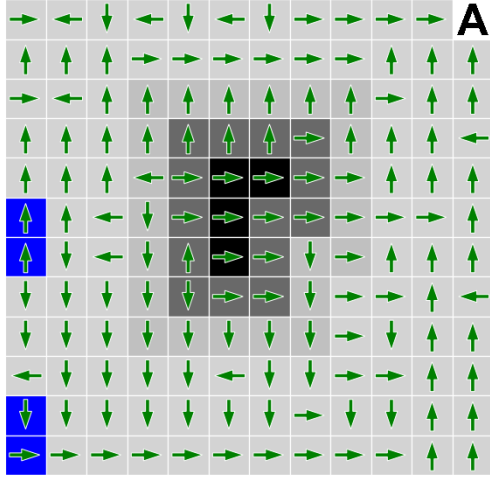
lambda = 0



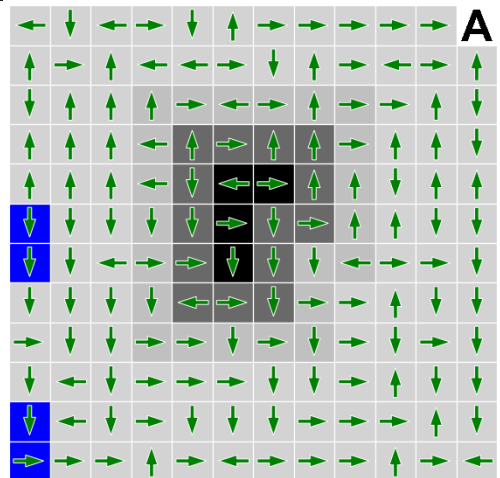
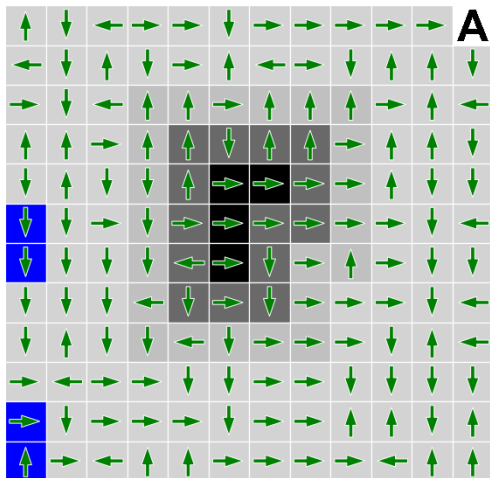
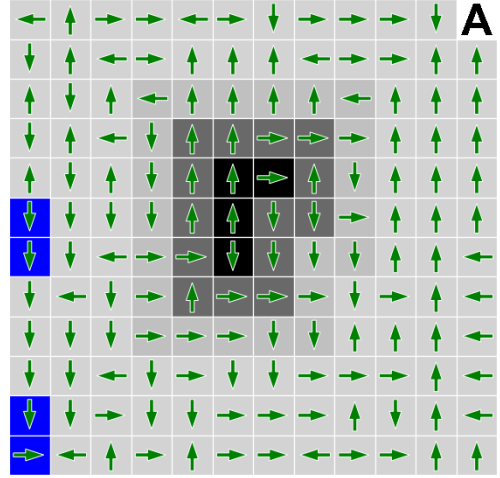
lambda = .3

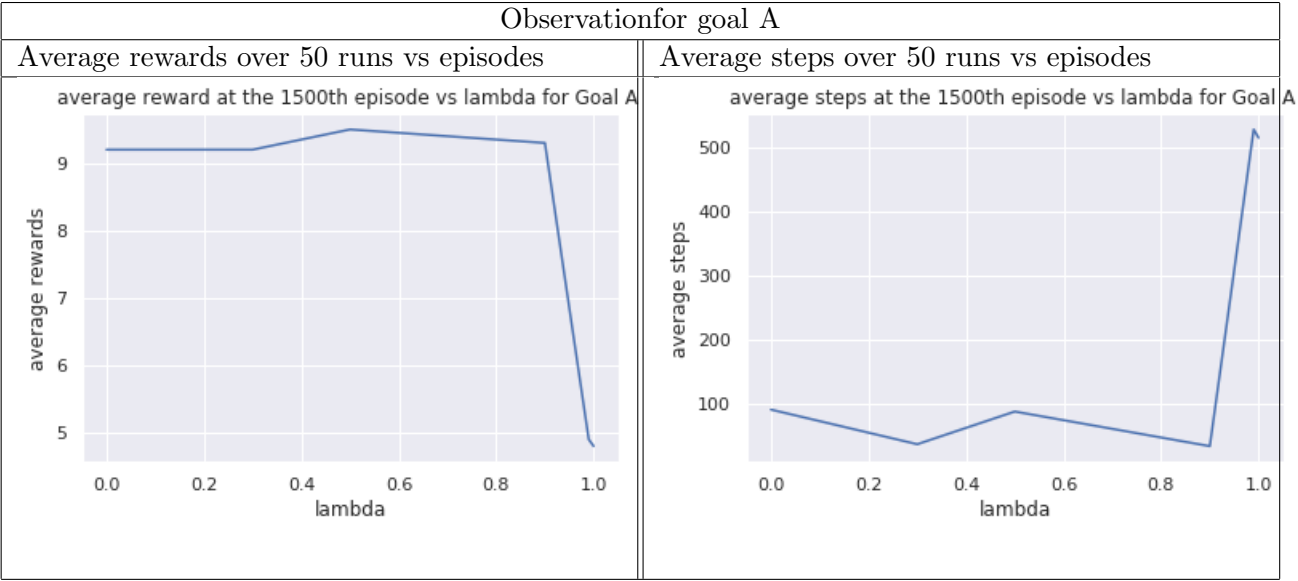


lambda = .5



lambda = .9





3.5 Implementation of Sarsa lambda for different lambdas for Goal B and related plots

3.5.1 Related plots

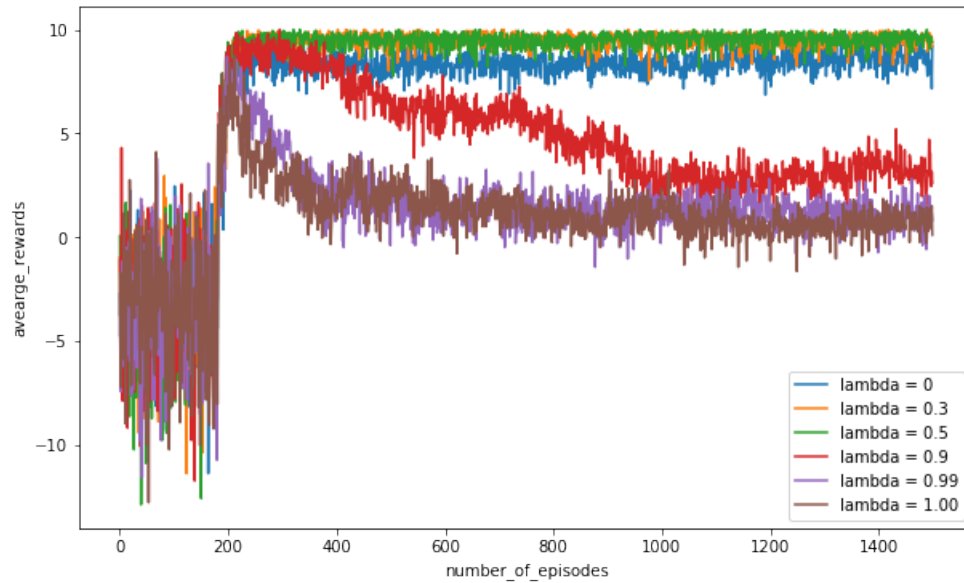


Figure 12: Average rewards over 50 runs vs episodes

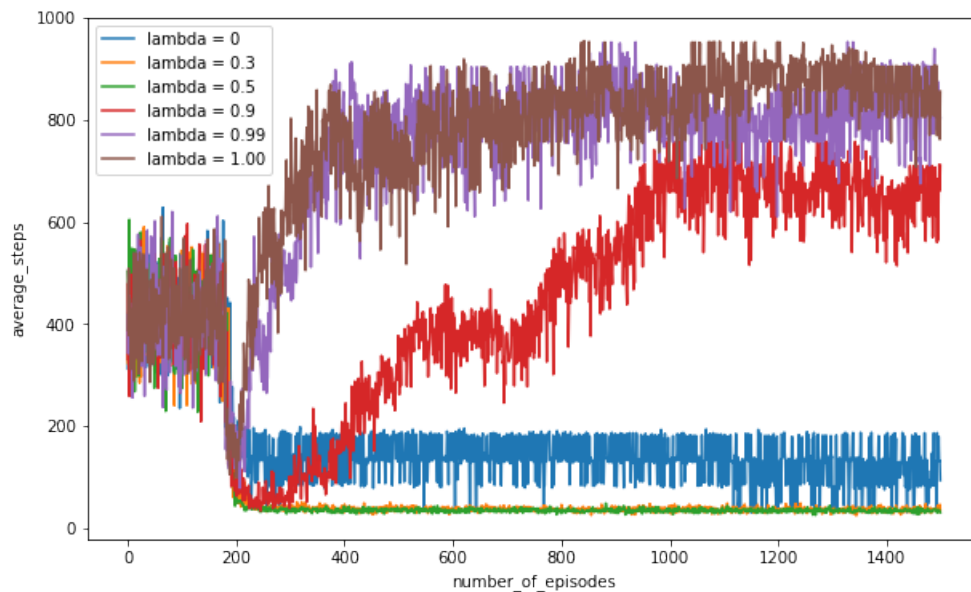
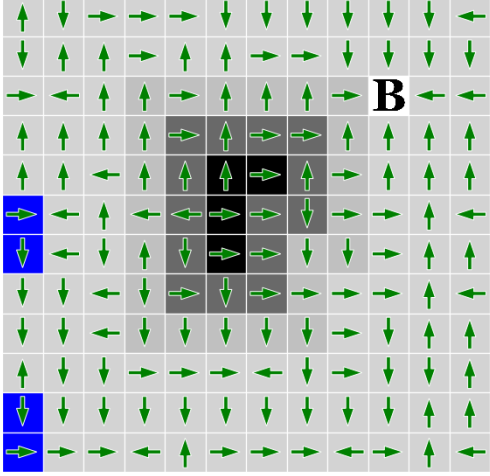
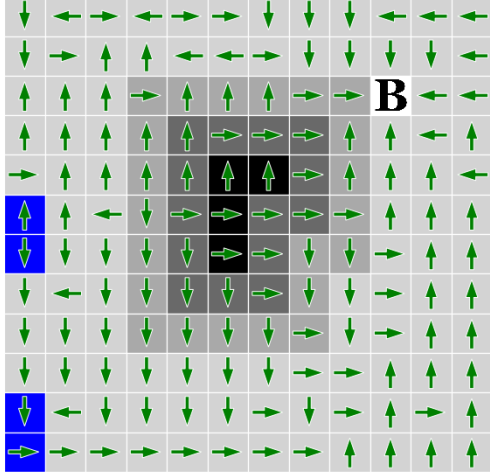
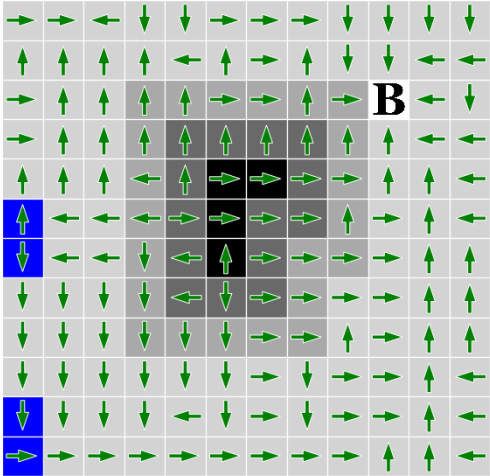
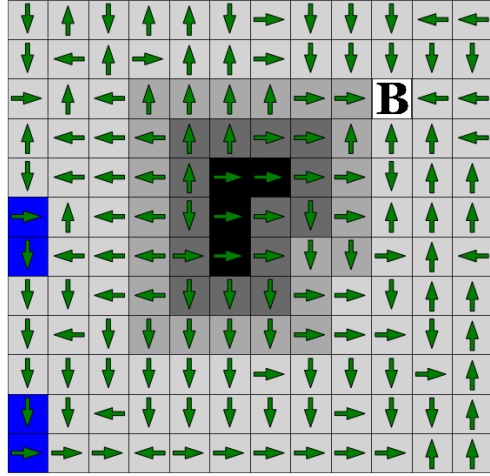
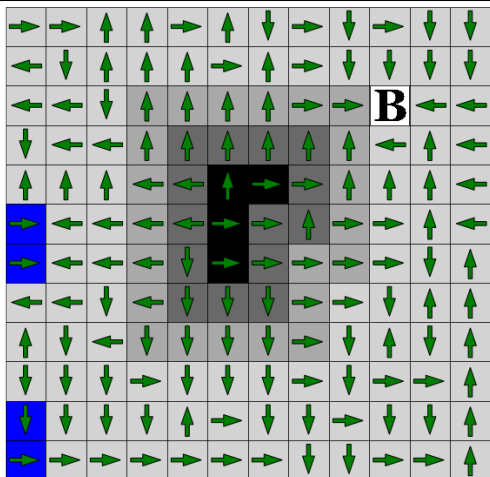
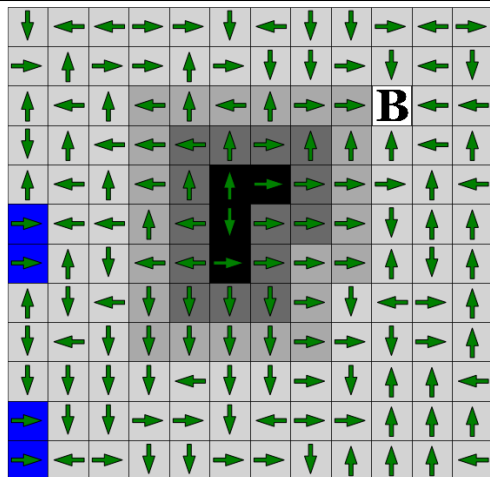
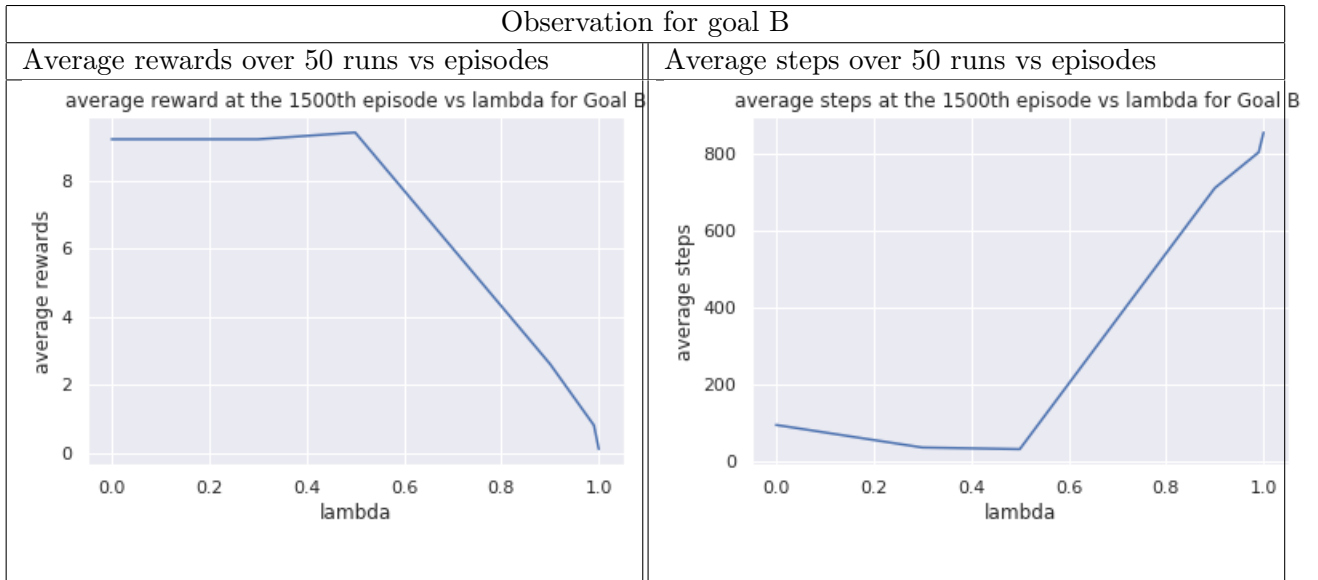


Figure 13: Average steps over 50 runs vs episodes

For Goal A Observation

lambda = 0	lambda = .3
	
lambda = .5	lambda = .9
	
lambda = .99	lambda = 1.0
	



3.6 Inference from Observation for Sarsa Lambda

Referring to the above plots:

- Average steps for goal C is 949
- Average steps for goal A is 39
- Average steps for goal B is 32
- As we know that reaching the goal agent gets 10 reward.
- As we know that
- **Average reward** : Before reaching the goal agent also has to face negative rewards . But because the environment non-deterministic with some probability and agent is also exploring so it reaches the goal. But average reward is less because goal itself is in puddle.
- **Average steps** : As we can see form the plot and the result we got, average step increases as the goal keeps coming near to puddle word. Because using q learning algo, agent does not hesitate to move inside the puddle.
- **Policy** : As it is clear form the average steps taken to reach the goal. As the goal moves towards puddle average steps decrease, it means learnt policy is not optimal. As we can see from the plot all the states does not learn the optimal policy. On of the reason might me possible that agent might not have visited that state that many number of times.
- For goal C sarsa lambda explodes

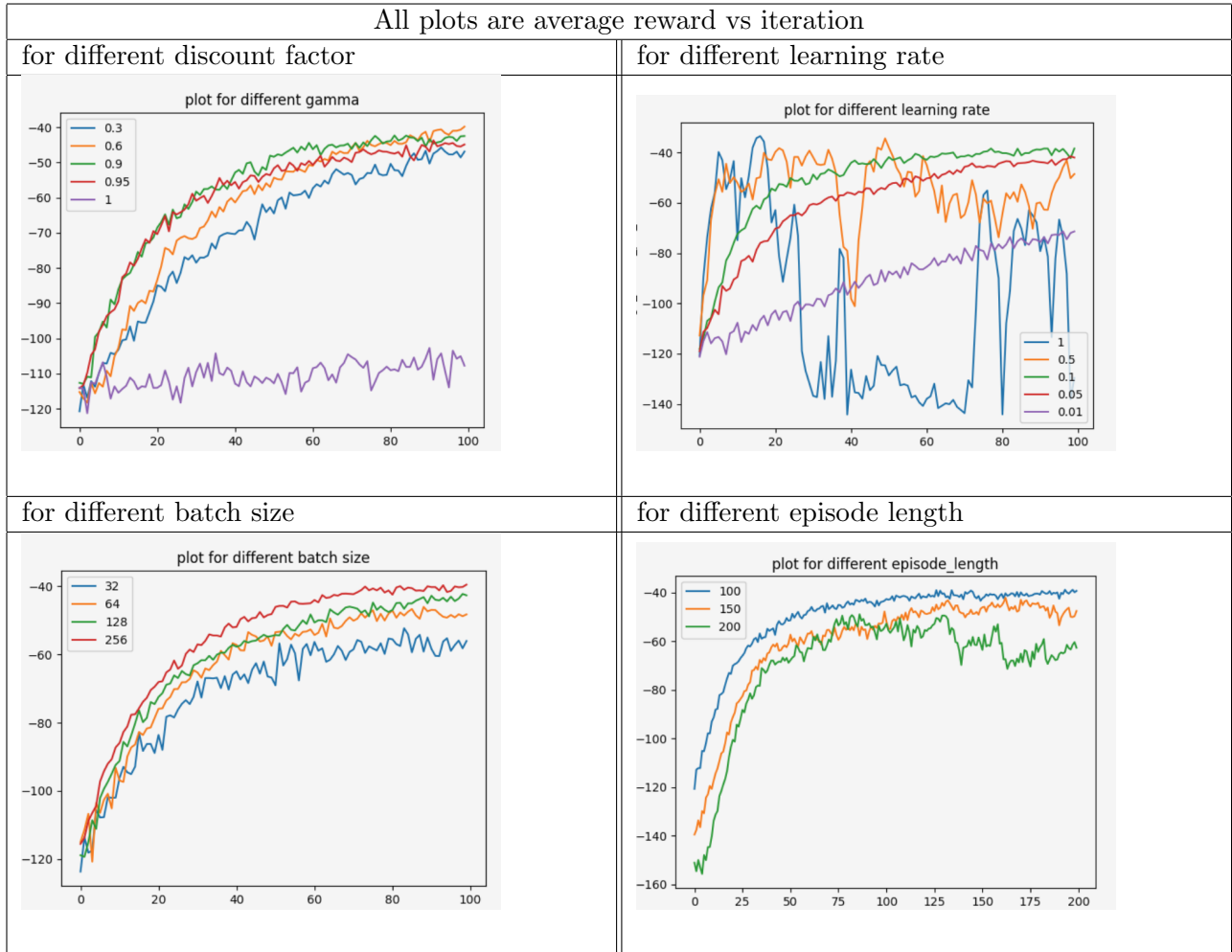
- for the figure 12 and 13 we can see that lambda increases average reward decreases and average steps increases.it is happening because as the lambda increase variance increase because we start considering whole trajectory and also my learning rate is high so it is updating so fast on current gradient values which leads to disaster.
- From the above figure it can be seen that as lambda increases average rewards and average steps suddenly change the behaviour because of high variance in the trajectories that is being sampled.
-

Observation is taken from the average of last episode over all runs for goal A	
Lambda value	average steps taken
0	91
0.3	37
0.5	88
0.9	34
0.99	528
1.0	515

4 Policy Gradient

4.1 Chakra

4.1.1 Hyper parameter tuning



4.1.2 Value function visualisation

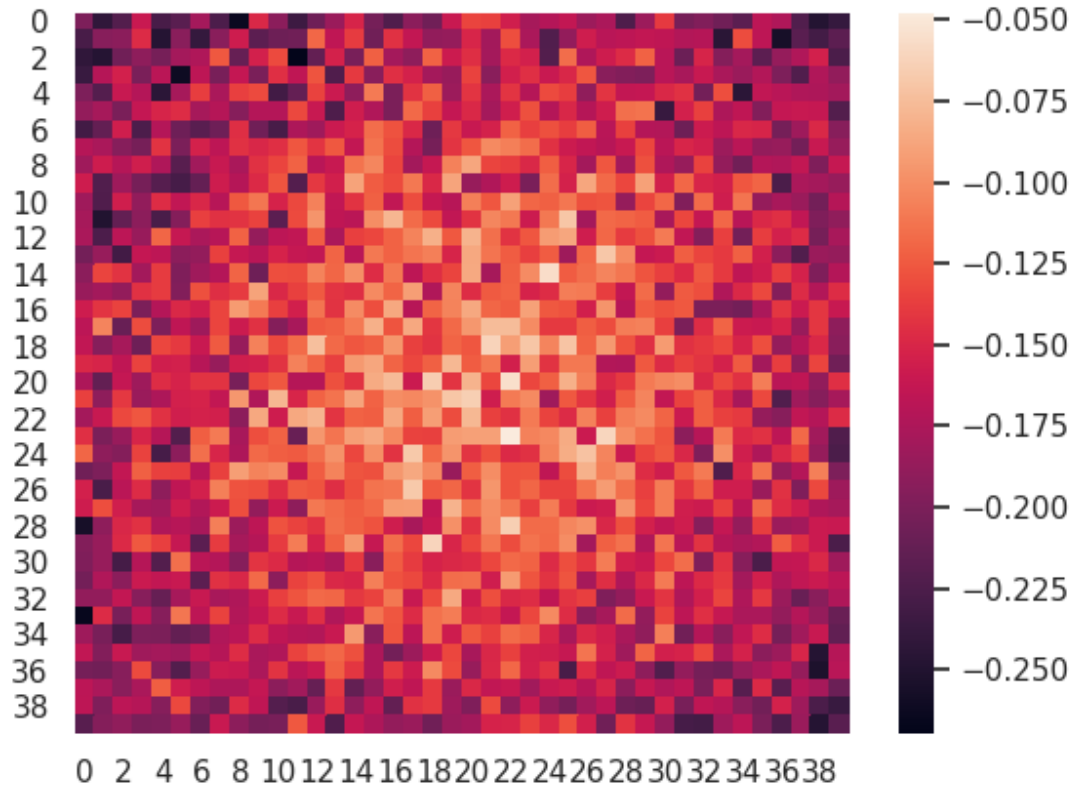


Figure 14: Value Function for CHAKRA

4.1.3 Trajectory and policy

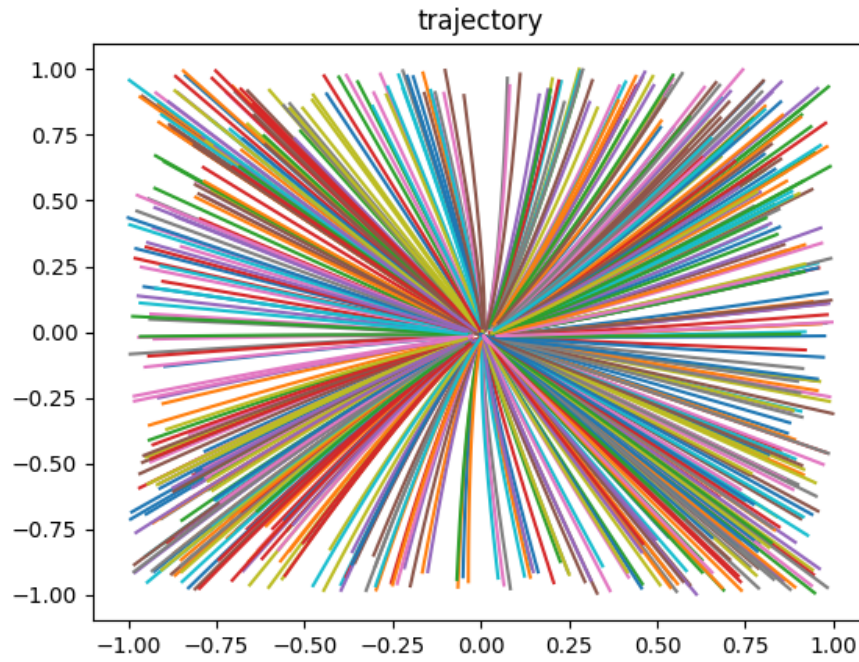


Figure 15: Trajectory for CHAKRA

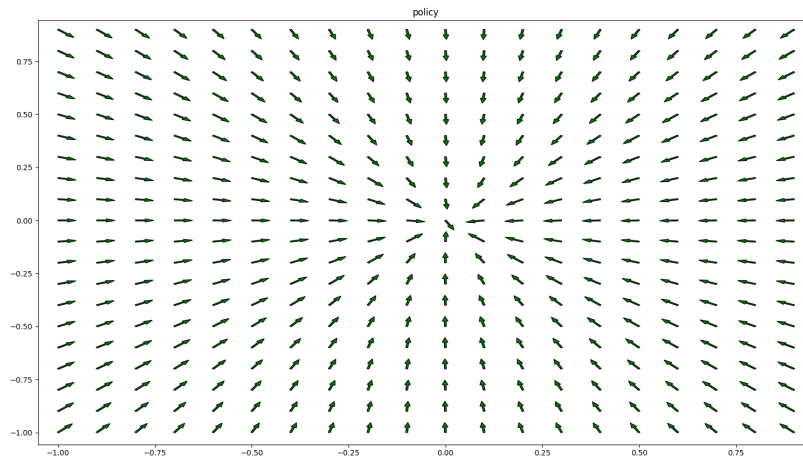


Figure 16: Policy for CHAKRA

4.1.4 Inference from observation

Best hyper parameter values: Batch size = 700, Iterations = 500, Episode length = 100, Discount factor = .9, Learning rate for policy parameter = .05, learning rate for value function = .5

Hyper-parameter tuning : Default value set to perform hyper parameter tuning experiment are:

Batch size = 128, Iterations = 200, Episode length = 100, Discount factor = .9, Learning rate for policy parameter = .05, learning rate for value function = .5

- **Discount Factor** As we can see from the above plot as we keep on increasing gamma average reward increases more fast. But from the figure it is clear that although gamma = .3 is slow but after much number of iterations it gives good average rewards. As we learning from trajectory so there is higher variance associated with it too much dependence on future rewards is not good, which can be seen from gamma = 1. For gamma = 1 it is possible that agent is trying to go out side more often which leads to ending the episode
- **Learning Rate :** As we can see from the above plot as we decrease learning rate it performs good although it is taking more iterations to converge. Very high learning rate results in high variance in average rewards. taking the Small step opposite to gradient helps a lot. Because we are learning using trajectory samples so there is high variance so keep learning rate low is good. Because there exist local optima also so values lower then .05 performs decent.
- **Batch size :** As we increase batch size performance keep on improving. Large batch size means large interactions with environment , so in result we will have good gradient.
- **Episode Length :** As we keep increasing the episode length average reward dose not show any improvement because if in a given episode agent does not reach the goal then it keep exploring the environment which leads to increase the number of steps but if the episode length is small then if the agent does not reach the goal it stops early as the episode length is small.

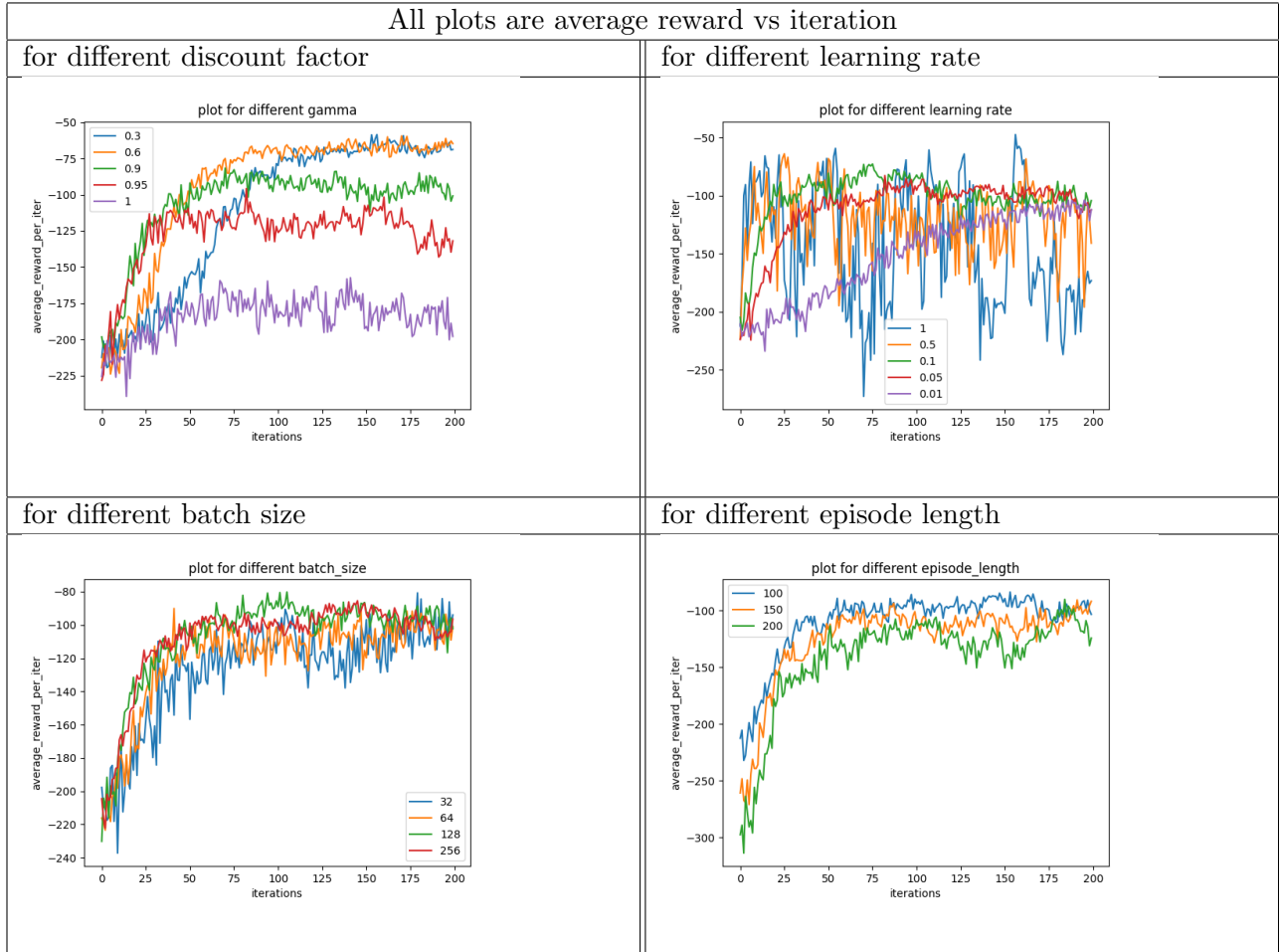
Value Function : As we can see from the above figure it looks like same as environment. Because negative reward keep on increasing as we go far from center. That is why state value function also become small as we go far from center.

Trajectory : As we can see from the above plot trajectories are going straight to the center by following the optimal policy. As the environment is deterministic so value of the states at the same distant from center has approx equal value.

Policy : As we can see from the above policy plot all the arrow goes towards center. Pattern is , it is symmetric around the goal.

4.2 Visahmc

4.2.1 Hyper parameter tuning



4.2.2 Value function visualisation

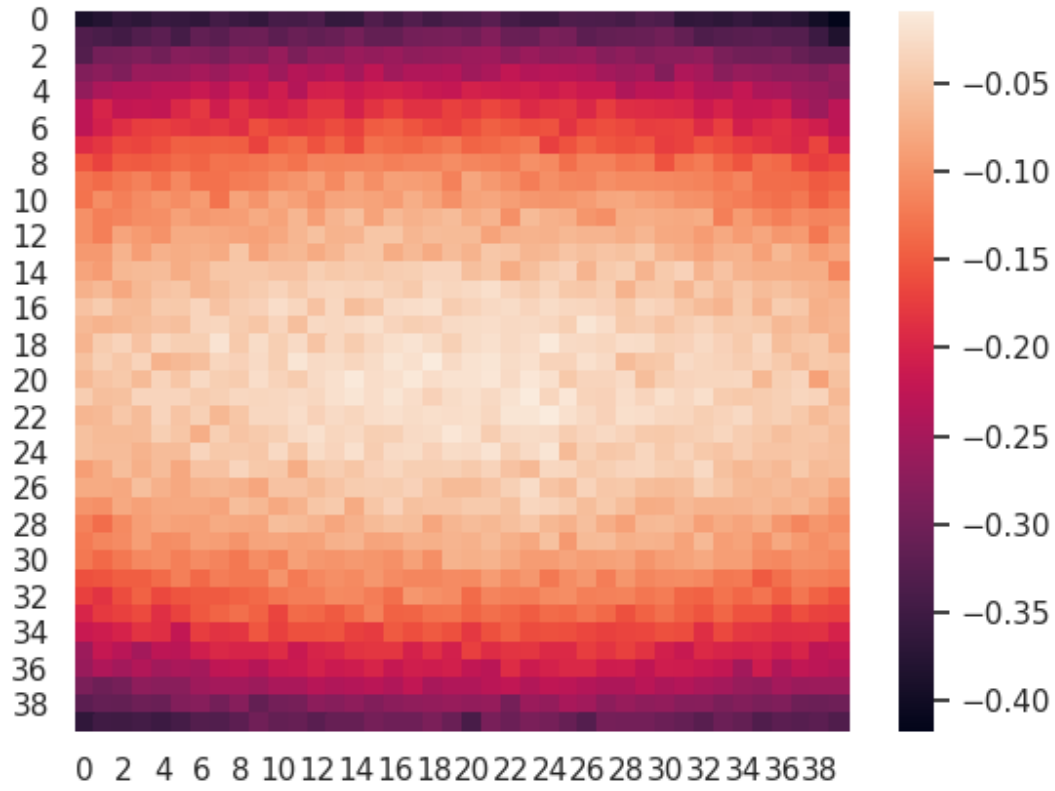


Figure 17: Value Function for VISHAMC

4.2.3 Trajectory and policy

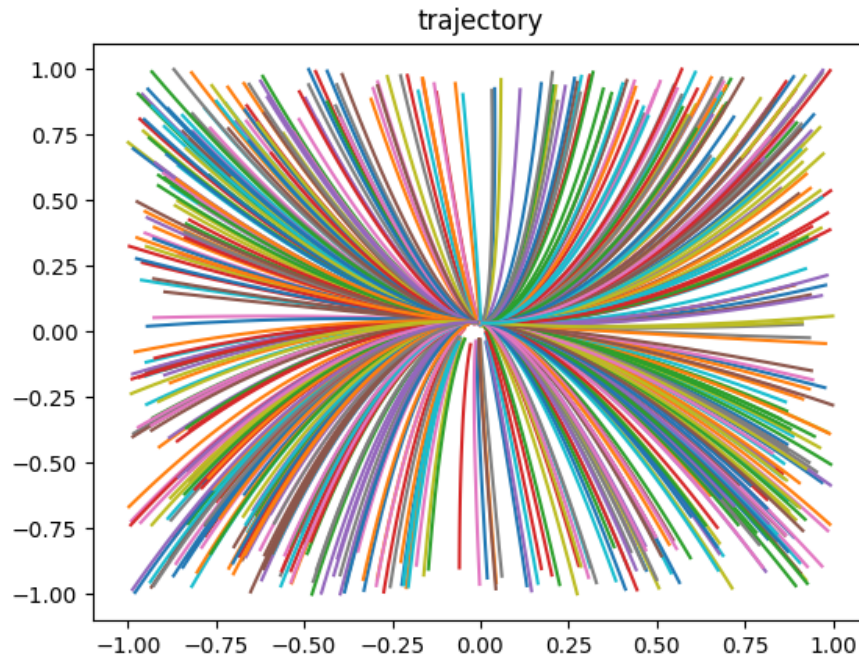


Figure 18: Trajectory for VISHAMC

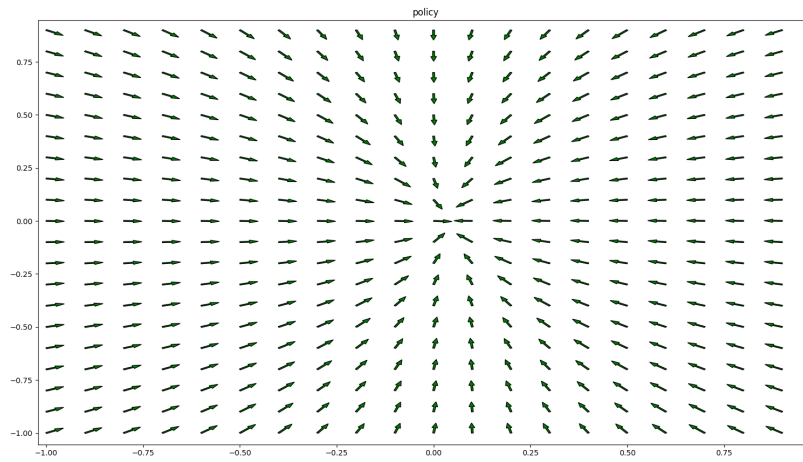


Figure 19: Policy for VISHAMC

4.2.4 Inference from observation

Best hyper parameter values: Batch size = 700, Iterations = 500, Episode length = 100, Discount factor = .9, Learning rate for policy parameter = .05, learning rate for value function = .5

Hyper-parameter tuning : Default value set to perform hyper parameter tuning experiment are:

Batch size = 128, Iterations = 200, Episode length = 100, Discount factor = .9, Learning rate for policy parameter = .05, learning rate for value function = .5

- **Discount Factor** As we can see from the above plot as we keep on increasing gamma average reward increases more fast. But from the figure it is clear that although gamma = .3 is slow but after much number of iterations it gives good average rewards. As we learning from trajectory so there is higher variance associated with it too much dependence on future rewards is not good, which can be seen from gamma = 1. For gamma = 1 it is possible that agent is trying to go out side more often which leads to ending the episode
- **Learning Rate :** As we can see from the above plot as we decrease learning rate it performs good although it is taking more iterations to converge. Very high learning rate results in high variance in average rewards. taking the Small step opposite to gradient helps a lot. Because we are learning using trajectory samples so there is high variance so keep learning rate low is good. Because there exist local optima also so values lower then .05 performs decent.
- **Batch size :** As we increase batch size performance keep on improving. Large batch size means large interactions with environment , so in result we will have good gradient.
- **Episode Length :** As we keep increasing the episode length average reward dose not show any improvement because if in a given episode agent does not reach the goal then it keep exploring the environment which leads to increase the number of steps but if the episode length is small then if the agent does not reach the goal it stops early as the episode length is small.

Value Function : As we can see from the above figure it looks like same as environment. Value function also looks like ellipse. Because negative reward increase more fast in the direction of small radius of the ellipse and that is reflected in the state value function.

Trajectory : As we can see from the figure, trajectories are bent. All trajectories tries to pass through those states which has high values. Because it is a ellipse so bigger radius side will have higher states values. As we can see from the above plot trajectories are going bent to the center by following the optimal policy. As the environment is deterministic so value of the states follow the ellipse structure.

Policy : As we can see from the above policy plot all the arrow goes towards center. Pattern is , it is following ellipse structure around the goal.