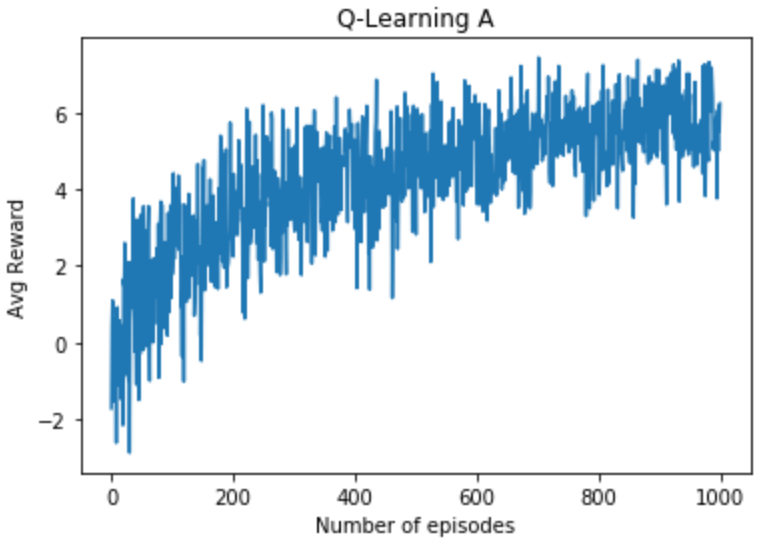
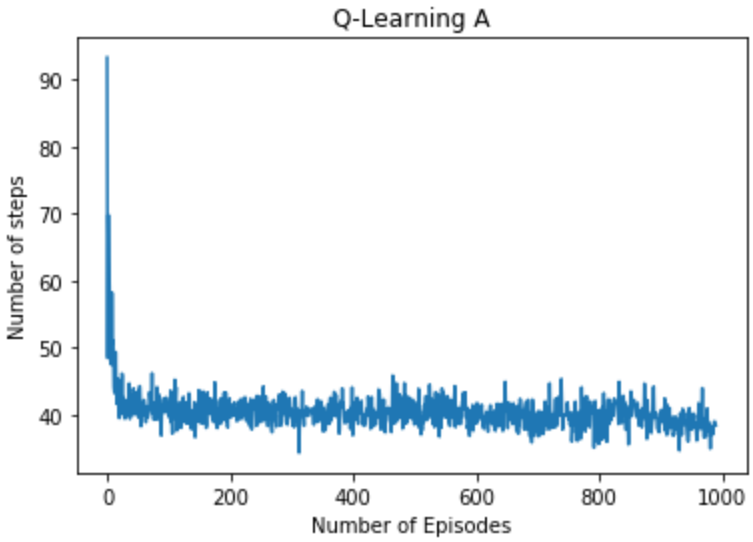
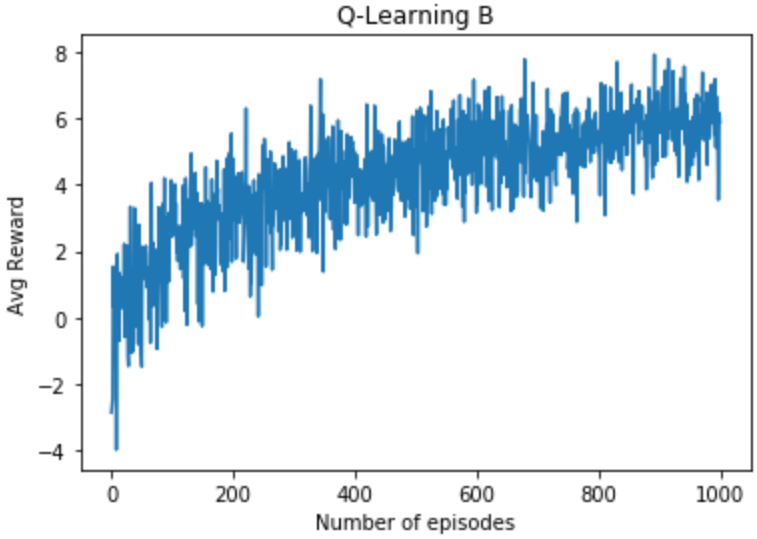
Programming Assignment #2
Control Algorithms

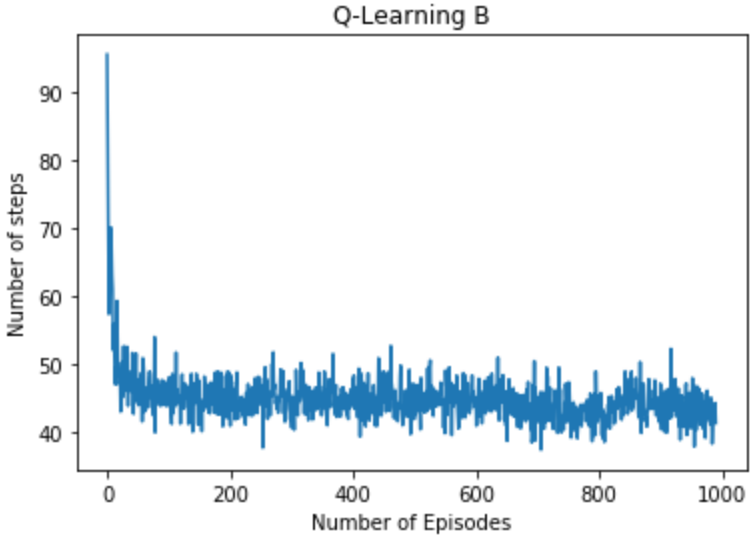
Q-Learning

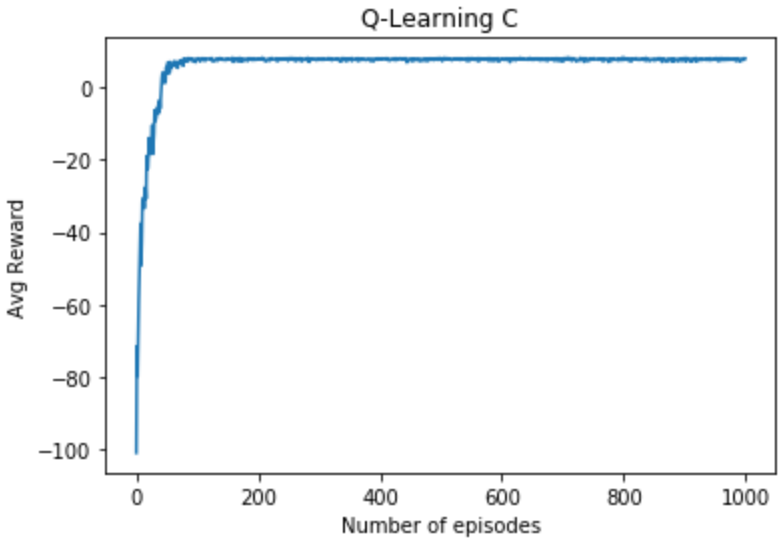
Plots :

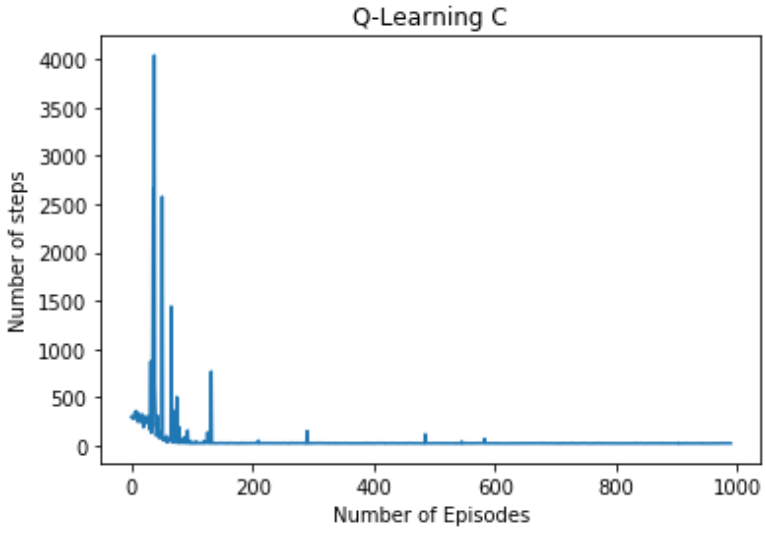






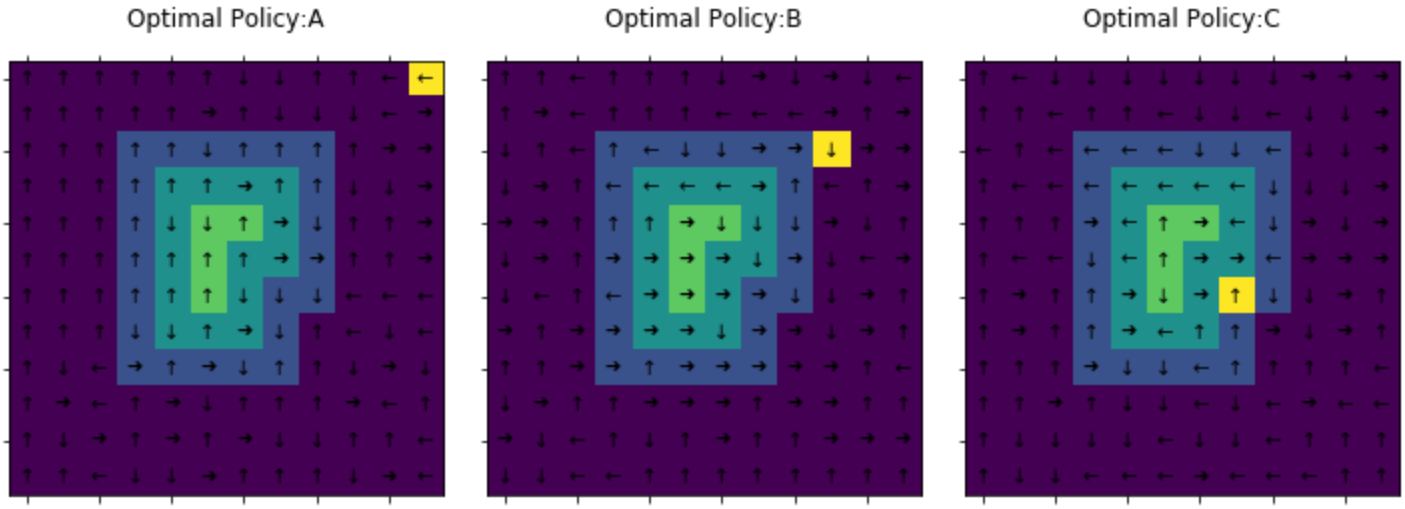






Optimal policies

- kkk



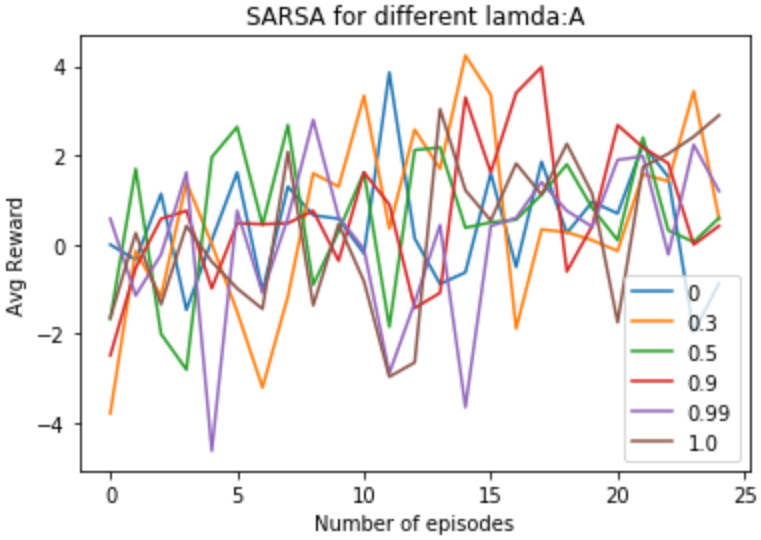
Inference

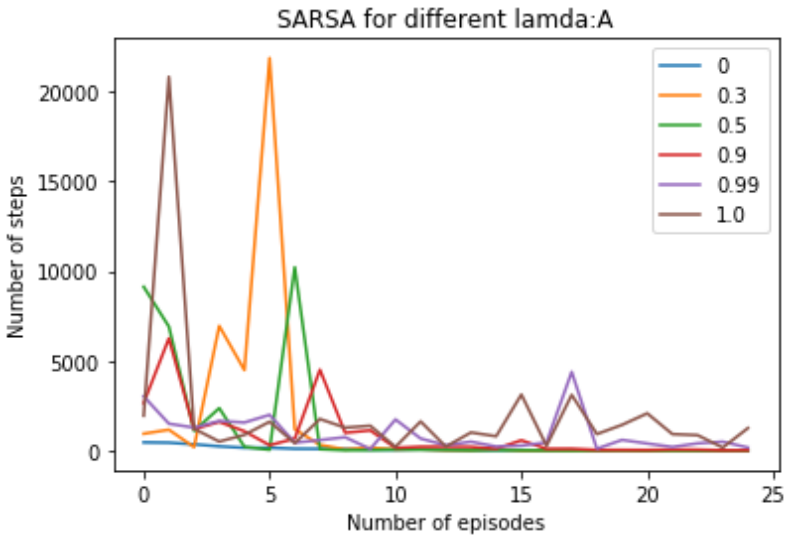
- Better optimal policy and convergence obtained with ALPHA = 0.1 and EPS = 0.1
- Fine Average reward is obtained when this EPS is converged at a rate of 0.99 for every episode.
- The better convergence in C is due to removal of wind - grid becoming more deterministic.

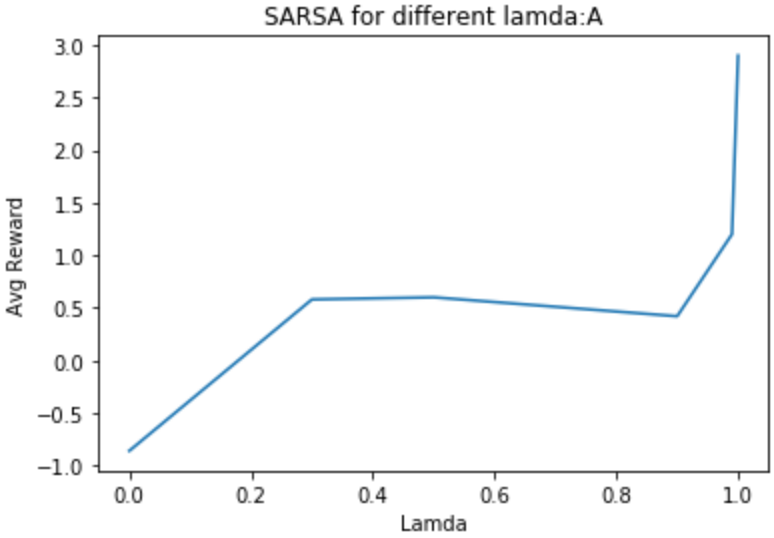
SARSA(λ)

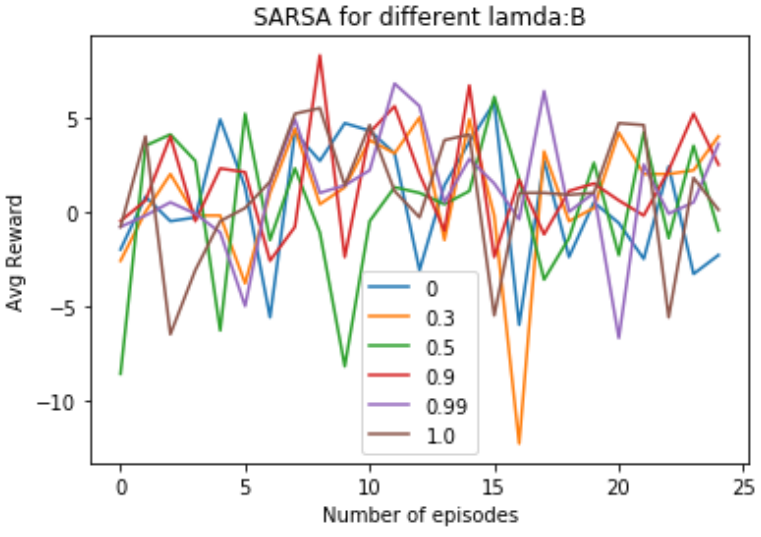
Plots :

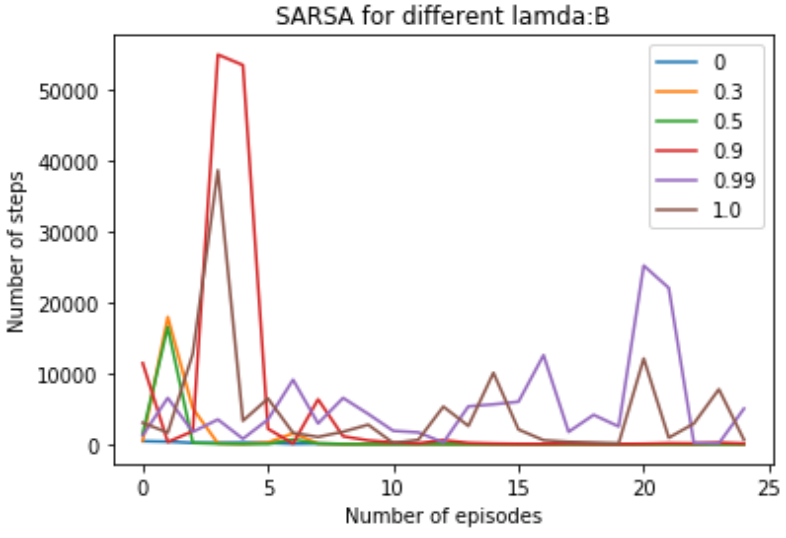
- For 25 learning trails

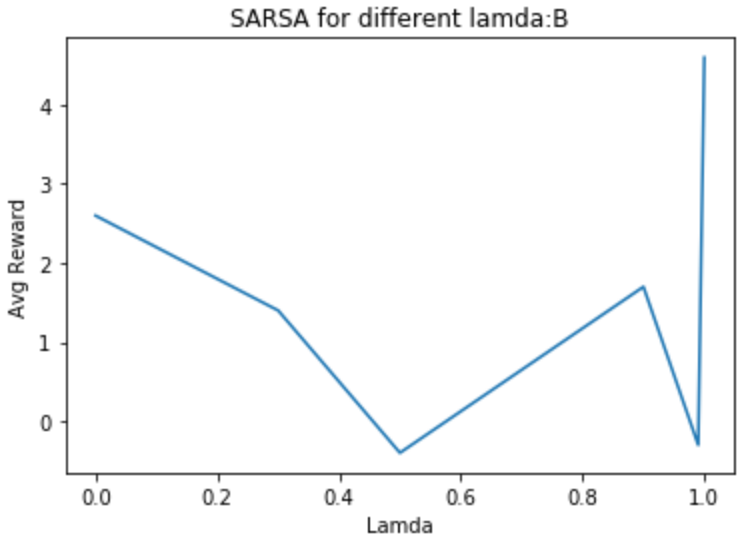

SARSA for different lamda:A
Avg Reward
Number of episodes
0
0.3
0.5
0.9
0.99
1.0

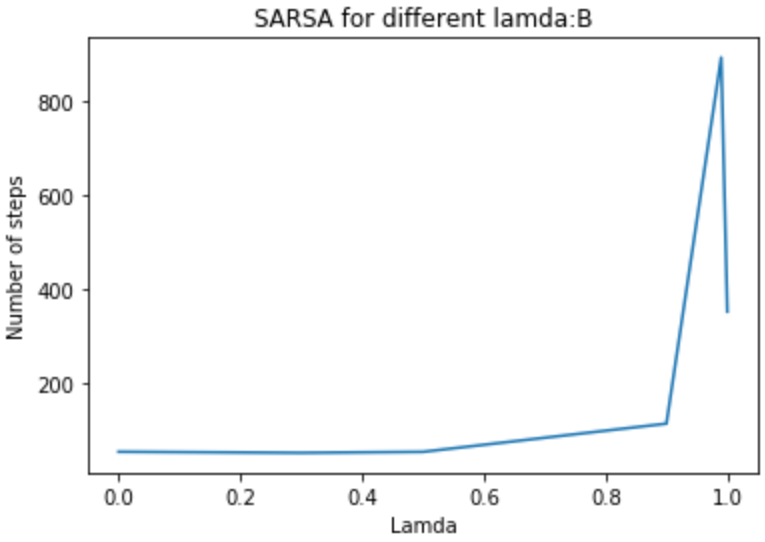

SARSA for different lamda:A
Number of steps
Number of episodes
0
0.3
0.5
0.9
0.99
1.0

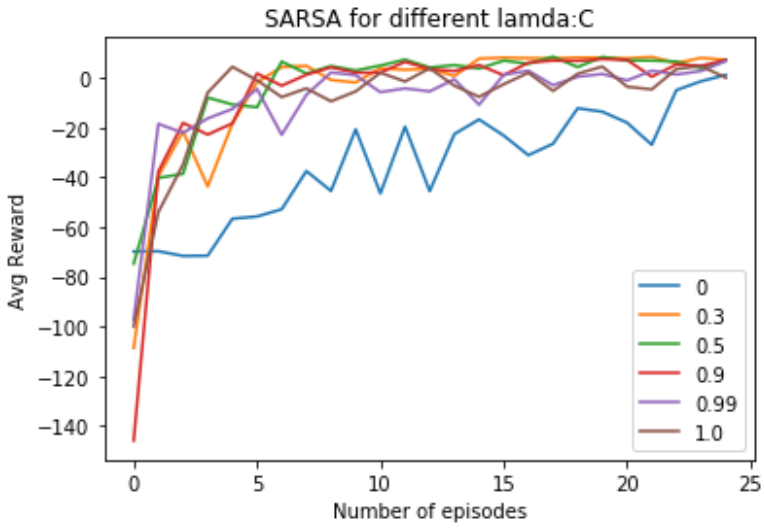

SARSA for different lamda:A
Avg Reward
Lamda

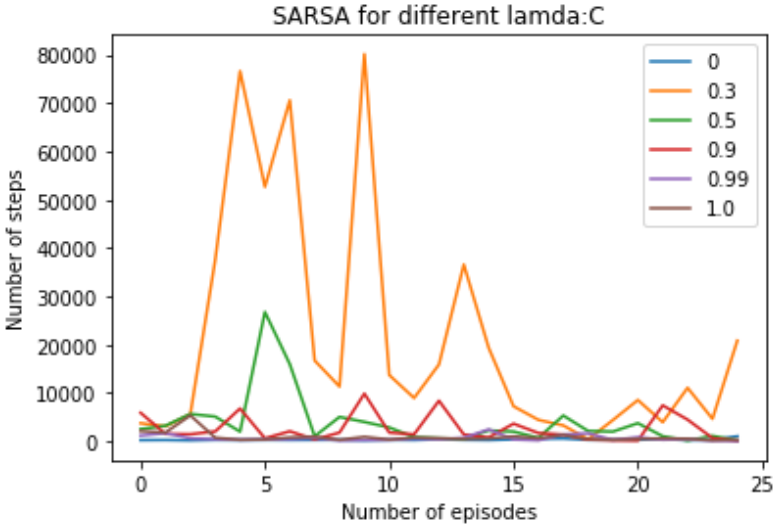

SARSA for different lamda:B
Avg Reward
Number of episodes
0
0.3
0.5
0.9
0.99
1.0

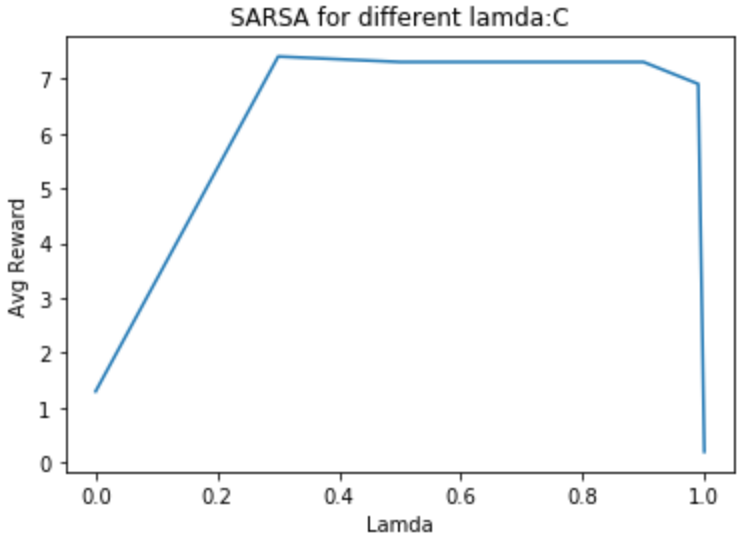

SARSA for different lamda:B
Number of steps
Number of episodes
0
0.3
0.5
0.9
0.99
1.0

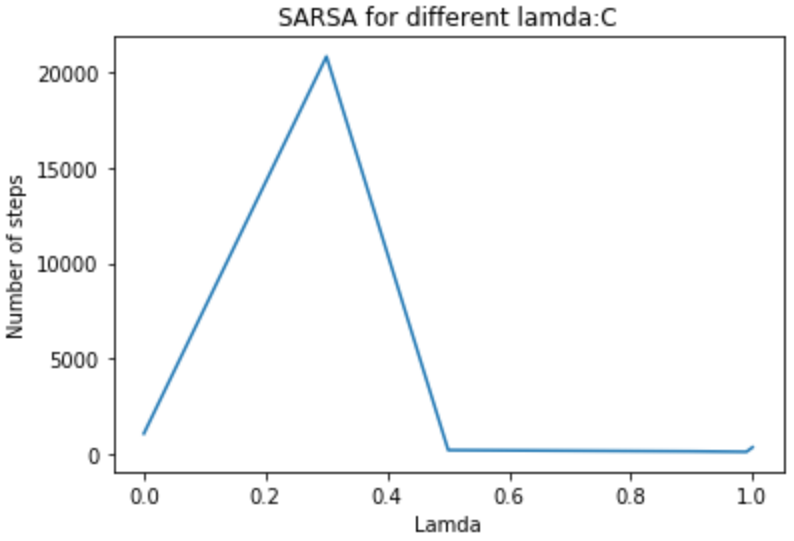

SARSA for different lamda:B
Avg Reward
Lamda


SARSA for different lamda:B
Number of steps
Lamda


SARSA for different lamda:C
Avg Reward
Number of episodes
0
0.3
0.5
0.9
0.99
1.0


SARSA for different lamda:C
Number of steps
Number of episodes
0
0.3
0.5
0.9
0.99
1.0

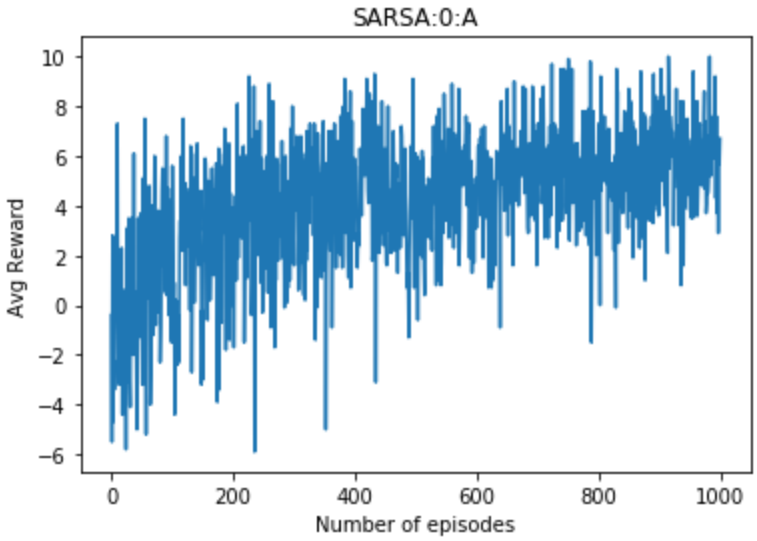

SARSA for different lamda:C
Avg Reward
Lamda

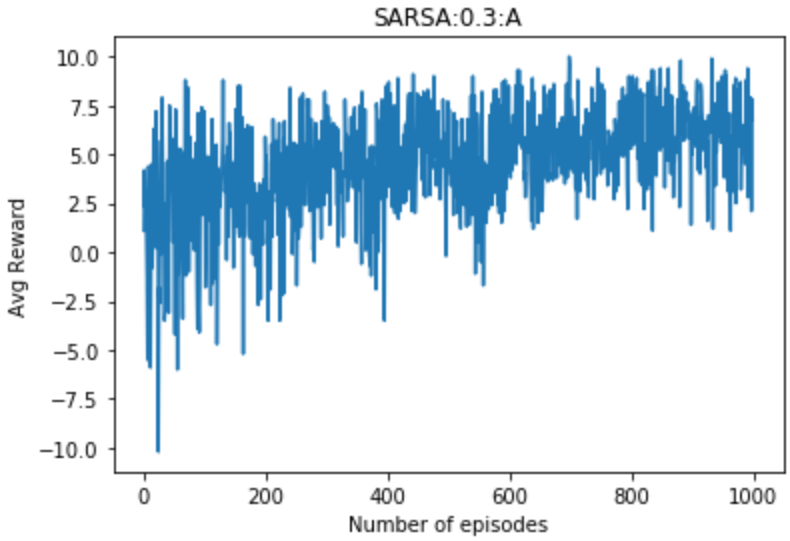

SARSA for different lamda:C
Number of steps
Lamda

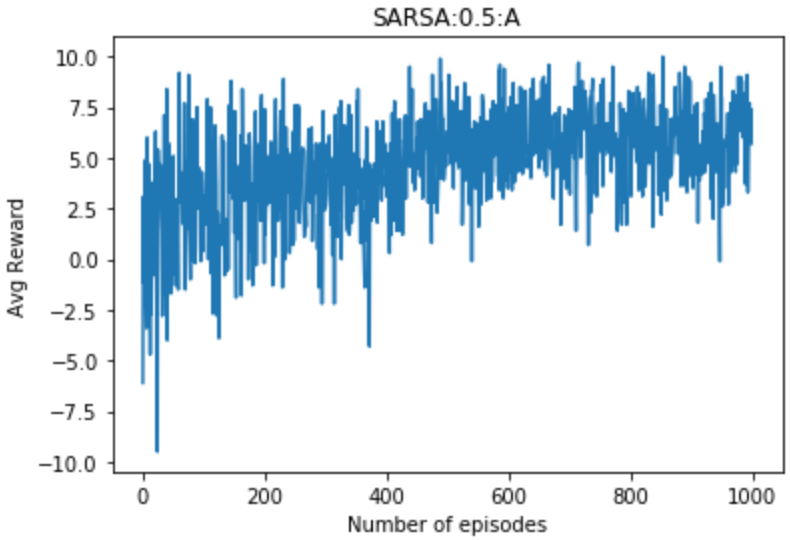
Inference

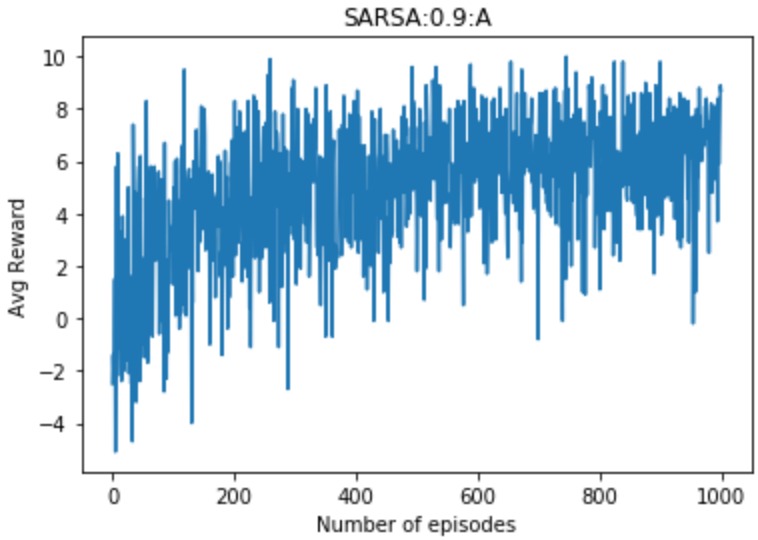
- λ = 0.5 and 0.3 are showing decent average outputs for different variants.
- Parameters after tuning ALPHA = 0.15 and EPS = 0.1
- SARSA is computationally more expensive than Q-learning due to updating every state for each step part in the algorithm.
- For the same reason specified above variant C gives good results for all λ. The heavy variance owes to wind in A and B.

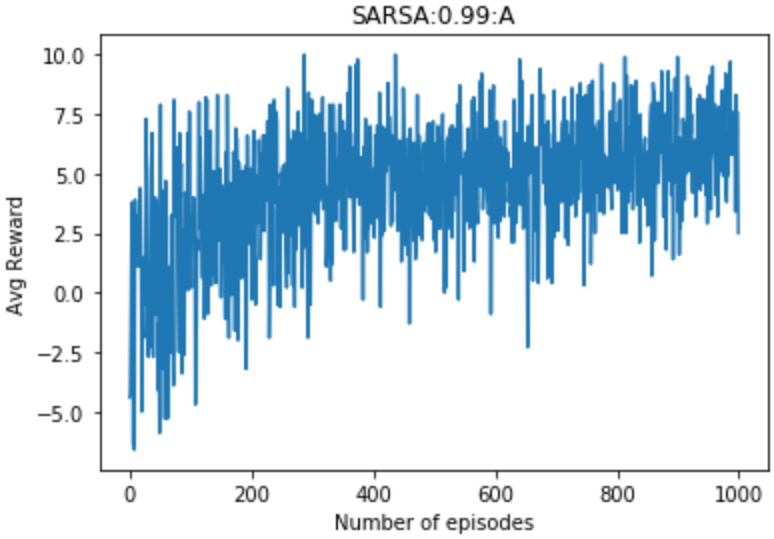
Other plots

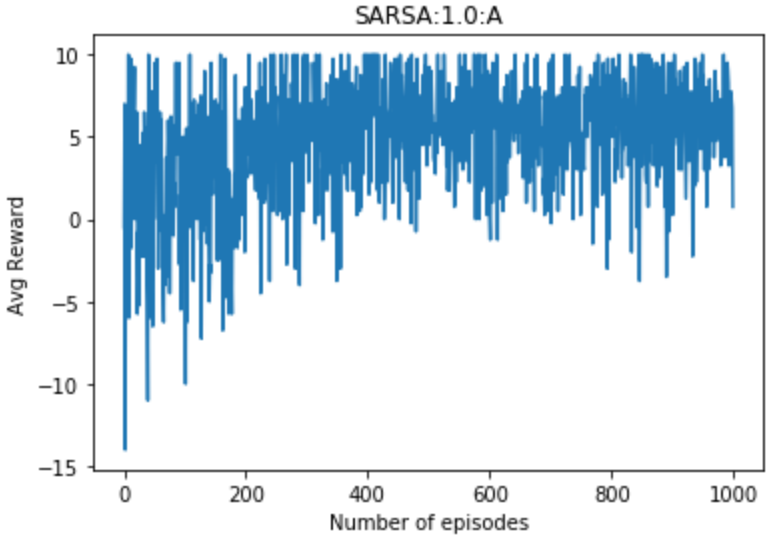


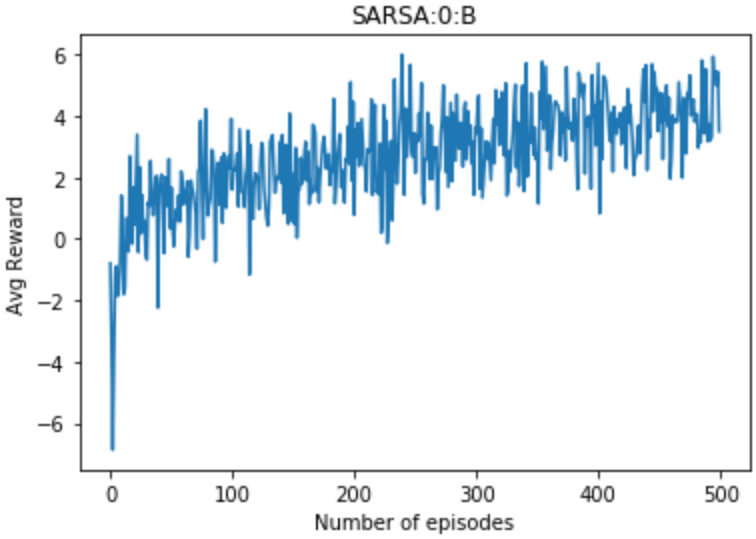


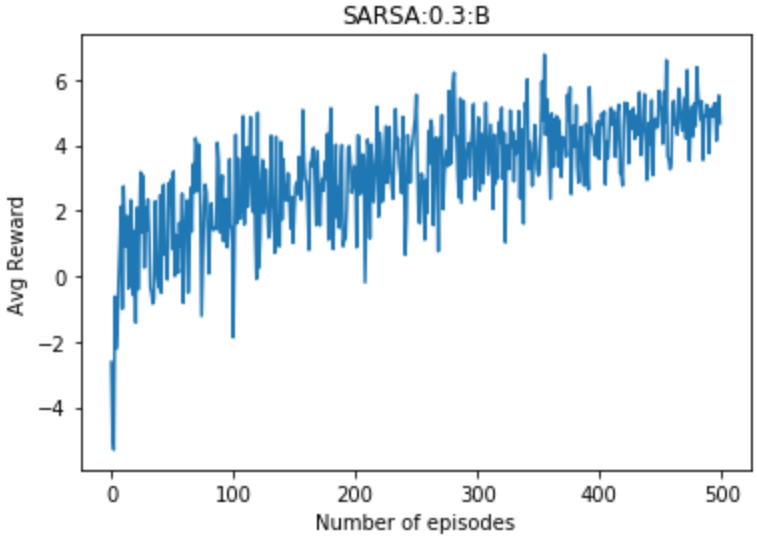


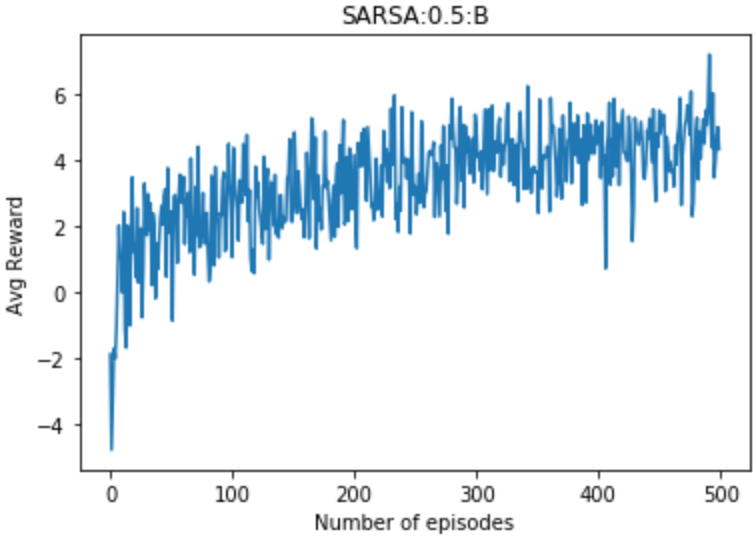


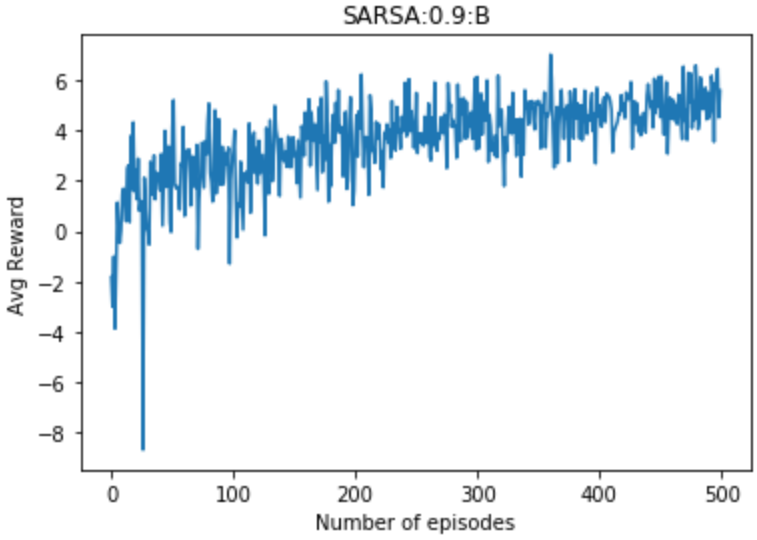

SARSA:0.99:A
Avg Reward
Number of episodes

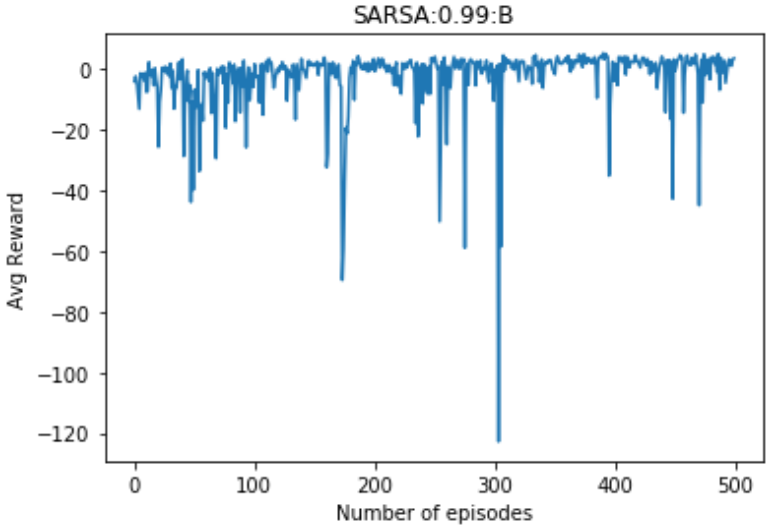

SARSA:1.0:A
Avg Reward
Number of episodes

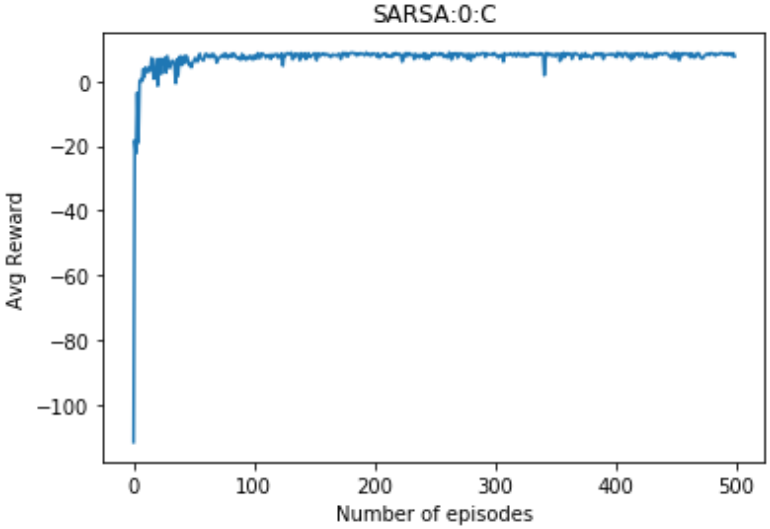

SARSA:0:B
Avg Reward
Number of episodes

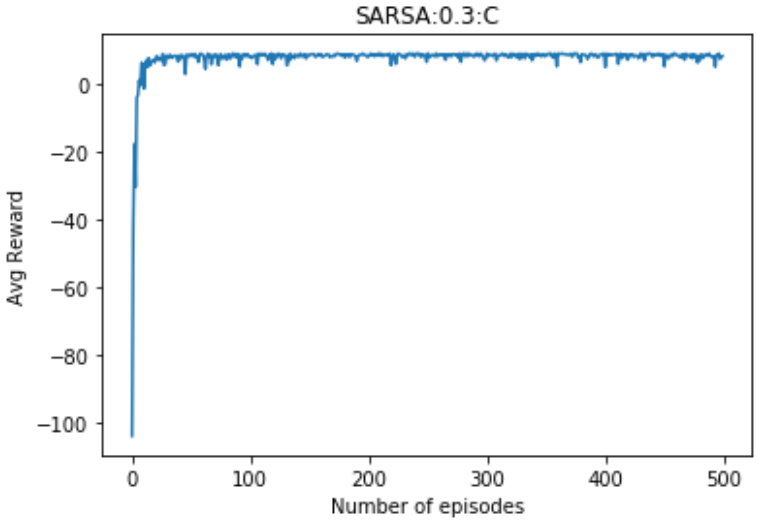

SARSA:0.3:B
Avg Reward
Number of episodes

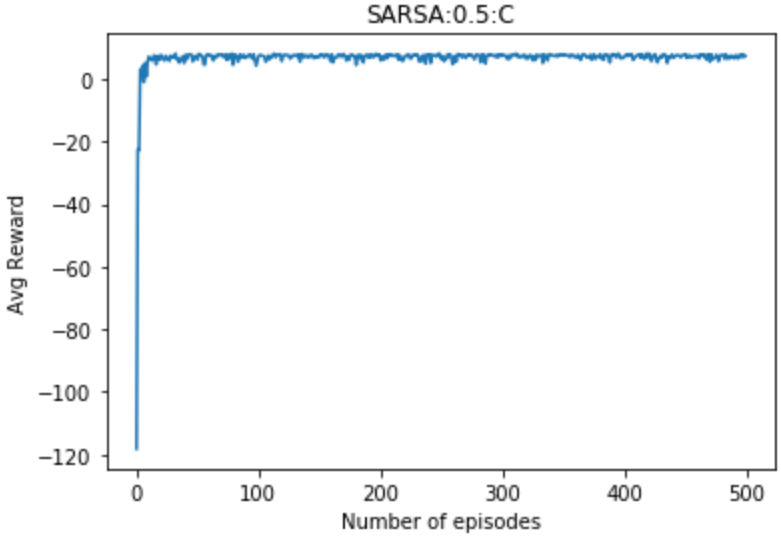

SARSA:0.5:B
Avg Reward
Number of episodes

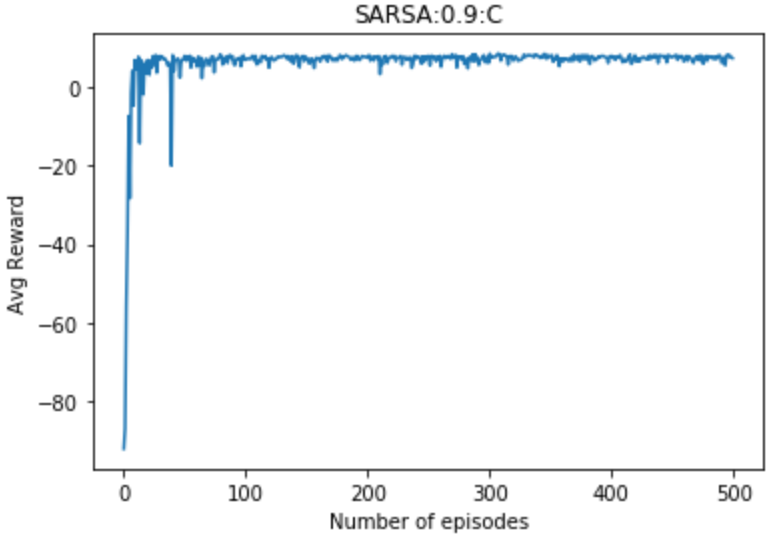

SARSA:0.9:B
Avg Reward
Number of episodes

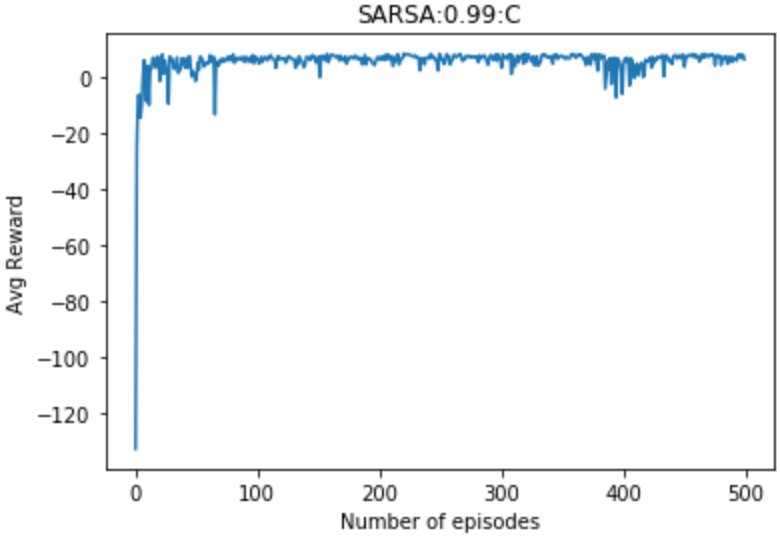

SARSA:0.99:B
Avg Reward
Number of episodes

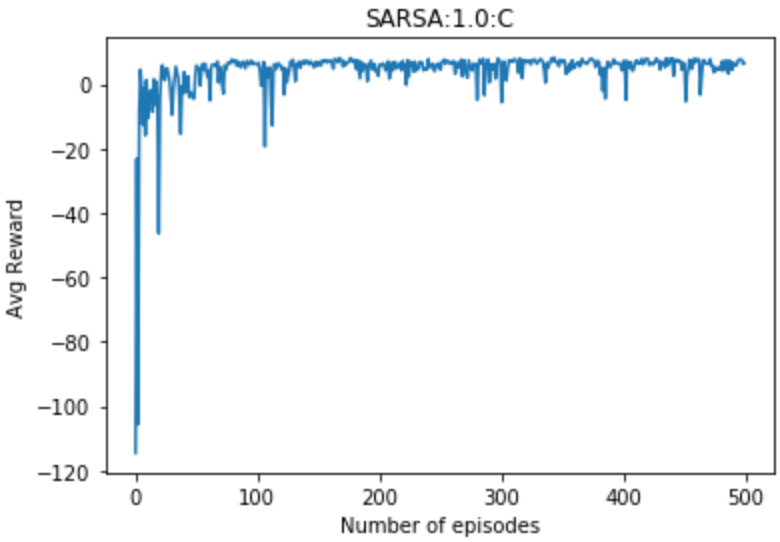

SARSA:0:C
Avg Reward
Number of episodes


SARSA:0.3:C
Avg Reward
Number of episodes


SARSA:0.5:C
Avg Reward
Number of episodes

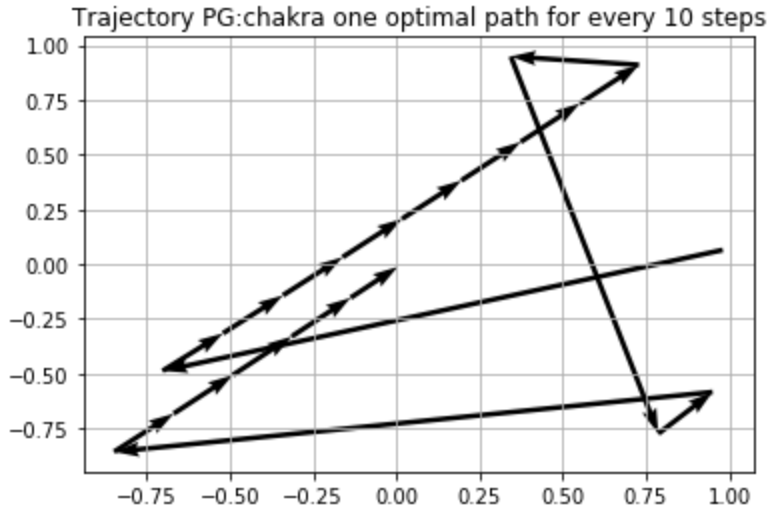

SARSA:0.9:C
Avg Reward
Number of episodes

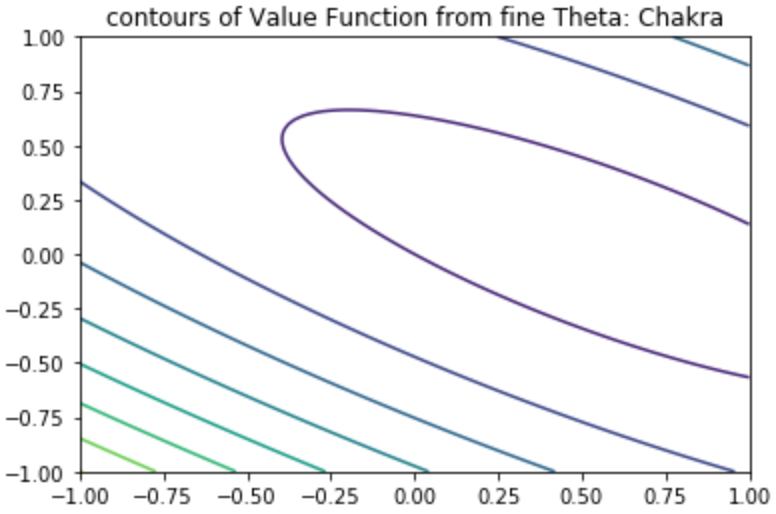

SARSA:0.99:C
Avg Reward
Number of episodes

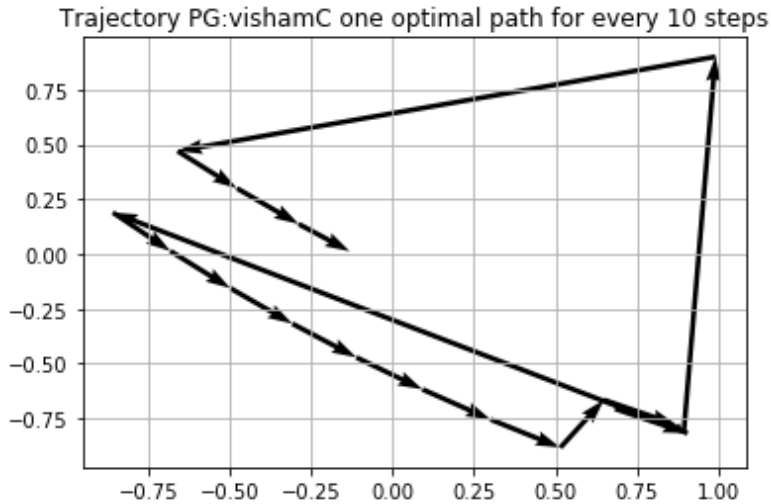


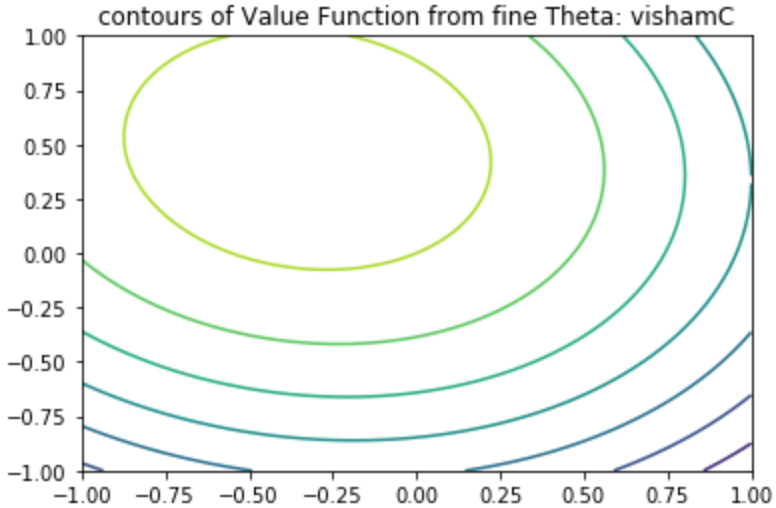
Policy Gradient

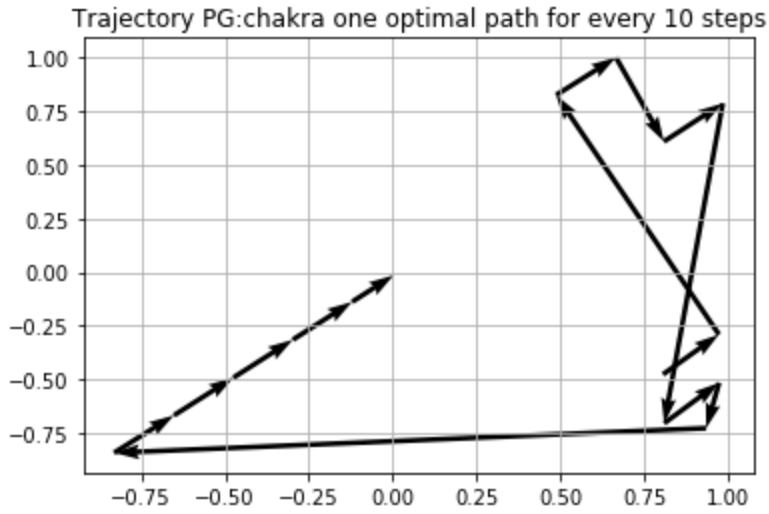
Plots:

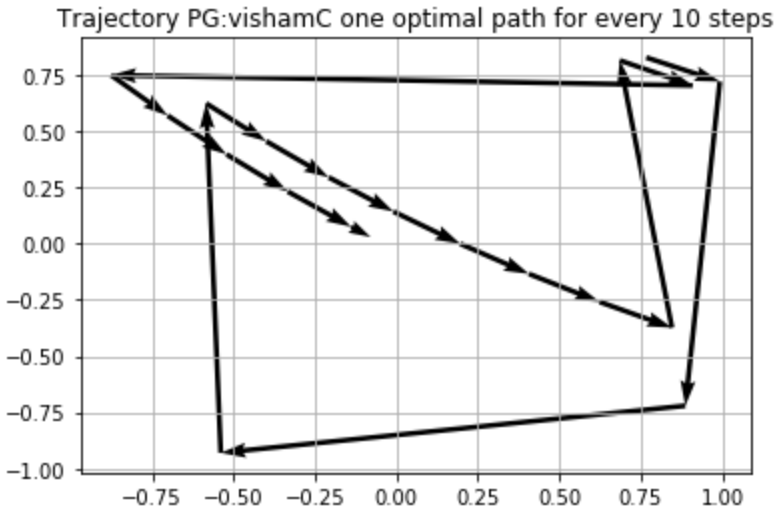












Working:

- Chakra - My reward function is the difference of distance between old position and present one. This one is giving better convergence faster than the given one. Although both make sense. That also explains different plots for the value function above.
- vishamC : Given Reward func is converging.

Tuning Parameters:

- max_iter = 30
- batch_size = 20
- step_limit = 1000
- ALPHA = 0.05
- GAMMA = 0.99

Answers

1. 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?

On increasing the batch size the aberration caused due to different random initialization and other stochastic aspects of the environment will be minimized. Keeping a big batch number might take longer time for the first iteration but the obtained theta after every batch run will be more precise.

On increasing learning rate the theta might oscillate around minima on the other hand less alpha will take a long time or might struck up at a local minima. It should be tuned to appropriate value.

Discount factor is a property of the environment. The step of 0.025 is small compared to the size of grid for a continuous problem like this so for good results lamda close to one is suggested.

2. 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.

   The plots for value functions are presented above with all plots. For chakra the function should be circles concentric but my reward function is different hence i got something like a distorted circles but the centre remained. For vishamC they are concentric ellipses as obtained in the figure. The centre is misplaced because in our theta the coefficient for bias term isn't the best.

3. 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?

   The best policy for chakra is going radially towards origin. The policy I arrived at is similar one which directs the agent almost towards origin but misses a few times. The trajectories of agent for optimal theta for 2 examples is plotted above.

   The best policy for vishamC is the optimal policy will be a spiral biased towards y coz of lambda = 10 factor. Suppose an agent starting at 1,1 takes long steps along Y and thereby crosses x = 0 and turns around. This process goes on until it hits origin.