Assignment-3 Submission

Indian Institute of Technology Delhi

COL333: Principles of Artificial Intelligence

Name: Anirudha Entry Number: 2019CS50421 Group: 04
Name: Deepanshu Entry Number: 2019CS50427 Group: 04

1 Computing policies

1.1 Value Iteration for Taxi Domain

In this part, we have chosen epsilon 0.01 as and max number of iterations as 20.

1.2 Value and Policy Iteration Graphs

Varying the discount factor:

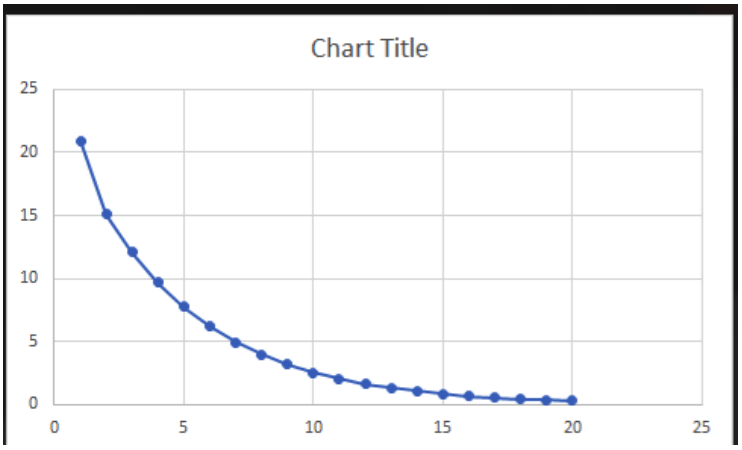

Figure 1: Value iteration

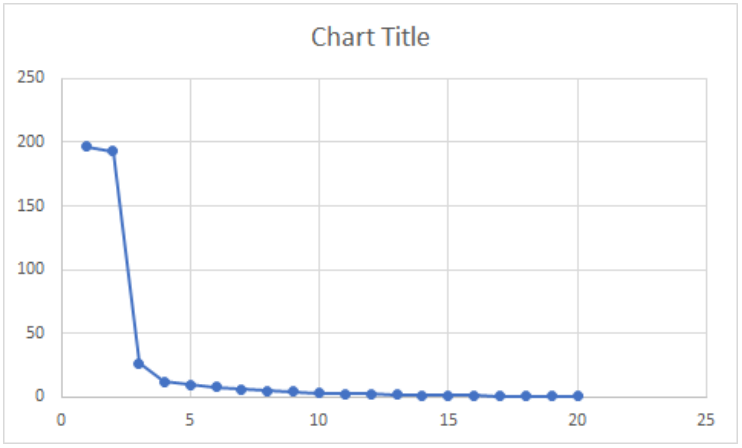

Figure 2: P iteration

1.3 Policy Iteration Implementation

for policy iteration implementation, values corresponding to each state transition is calculated with state, action and state'. max-norm is used to calculate the convergence criteria along with change in policy

2 Incorporating Learning

2.1 Evaluation of Q-Learning and SARSA

In this subsection, we will analyse the performance and convergence of Q-Learning and SARSA with exponential and fixed epsilon rate. We will plot the sum of discounted rewards accumulated during an episode (along y-axis) against the number of training episodes (along x-axis)

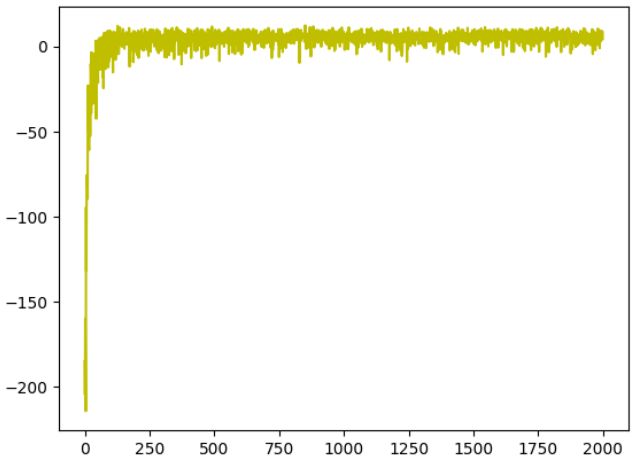

Figure 3: Q-learning with fixed epsilon rate of 0.1

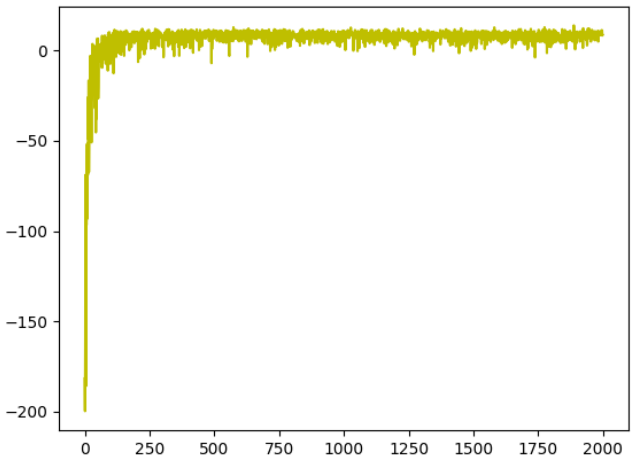

Figure 4: Q-learning with decaying epsilon rate with iteration number

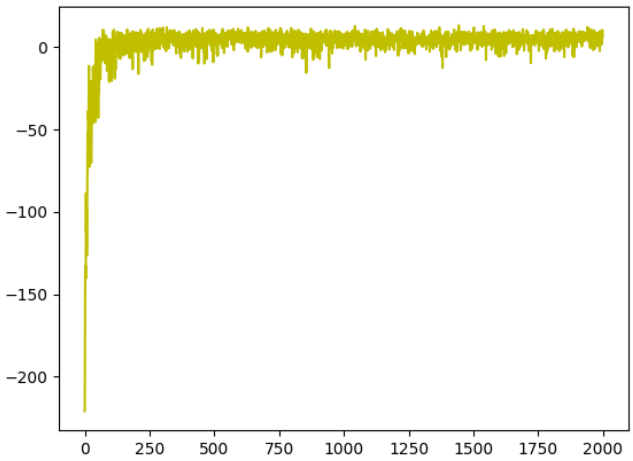

Figure 5: SARSA with fixed epsilon rate of 0.1

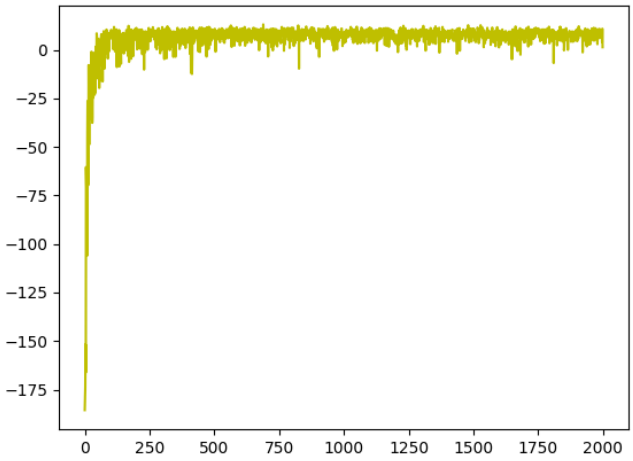

Figure 6: SARSA with decaying epsilon rate with iteration number

Following are the observations from the graphs:

1. A general trend that is observed is the fact that the average reward increases as the number of iterations are increased. This is due to the face that Q[s,a] improves as the learning progresses.

2. Since the reward converges after a certain point, this means that our policy and Q-values converges.

3. Sum of averaged rewards for all the methods are as follows:

   - Q-learning with fixed epsilon rate: 7261.09958917243
   - Q-learning with decaying epsilon rate:13029.58903159633
   - SARSA with fixed epsilon rate:5433.939354633428
   - SARSA with decaying epsilon rate:12301.424245772541

4. From this, we can infer that the best performing method is the Q-learning method with decaying epsilon rate. Similar inference can be made directly from the graph by looking at the reward value of the methods.

2.2 Executing the best policy on 5 problem instances

From the best part, the best performing method is the Q-learning method with decaying epsilon rate. Following was the performance on 5 different instances by varying the initial depot location of the passenger and the initial depot location of the taxi agent.

1.
   - Taxi: (3,1)
   - Person: (3,0)
   - Steps taken: 6

2.
   - Taxi: (1,4)
   - Person: (4,4)
   - Steps taken: 16

3.
   - Taxi: (0,3)
   - Person: (0,0)
   - Steps taken: 13

4.
   - Taxi: (3,1)
   - Person: (0,4)
   - Steps taken: 17

5.
   - Taxi: (4,2)
   - Person: (3,0)
   - Steps taken: 10

2.3 Impact of using a high or a low exploration rate

Varying the exploration rate (epsilon):

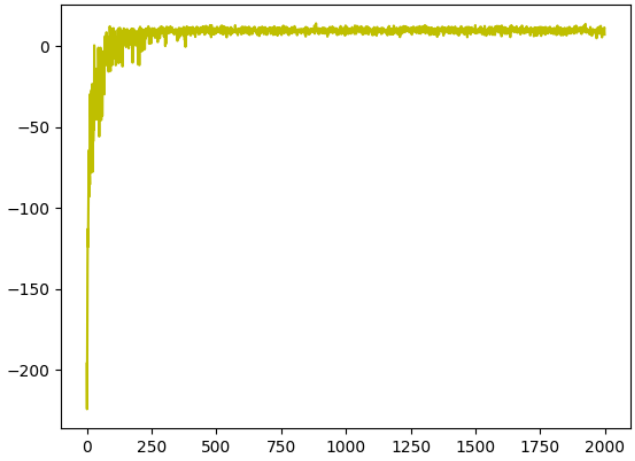

Figure 7: Q-learning with fixed epsilon rate of 0

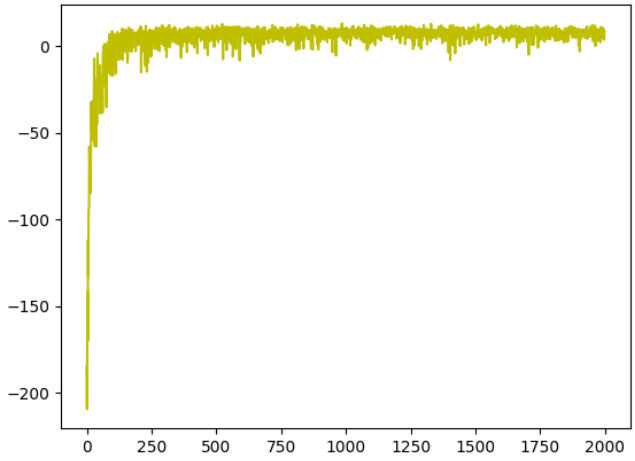

Figure 8: Q-learning with fixed epsilon rate of 0.05

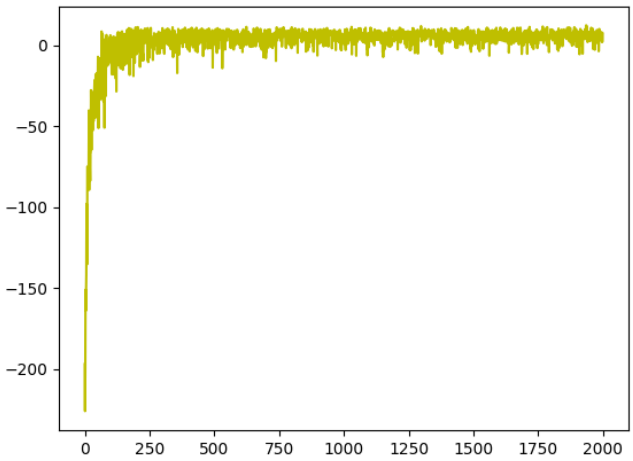

Figure 9: Q-learning with fixed epsilon rate of 0.1

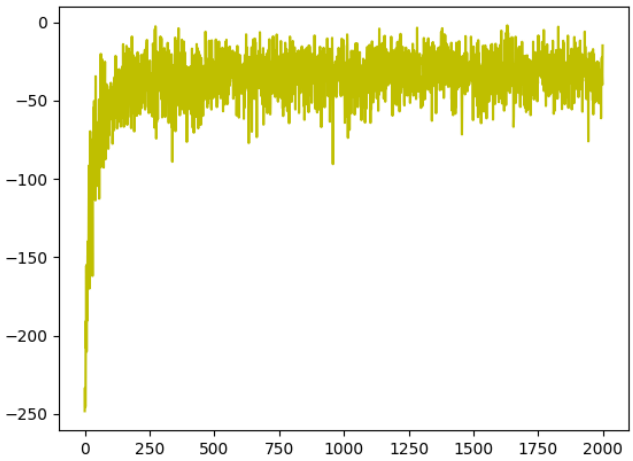

Figure 10: Q-learning with fixed epsilon rate of 0.5

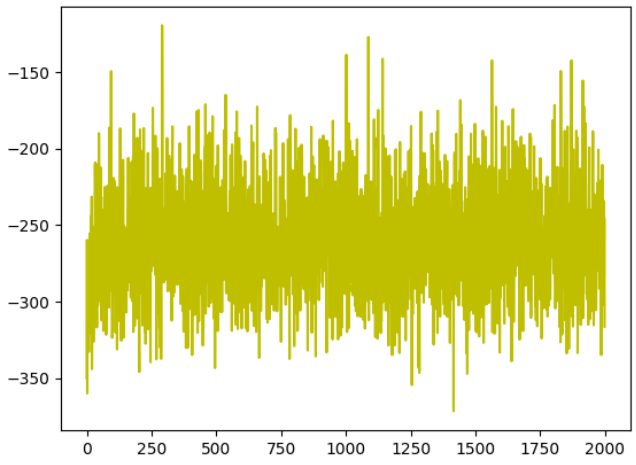

Figure 11: Q-learning with fixed epsilon rate of 0.9

Following are the observations from the graphs:

1. There is little to no benefit for fixed high epsilon as that is yielding a large number of random decisions and thus random walks. Thus, the graph shows no convergence.

2. Sum of averaged rewards for all the methods are as follows:

   - Epsilon rate of 0: 14035.766201033643
   - Epsilon rate of 0.05: 9798.307707008289
   - Epsilon rate of 0.1: 4879.4032841702165
   - Epsilon rate of 0.5: -73898.58946384738
   - Epsilon rate of 0.9: -520080.3003826425

3. Maximum reward sum is obtained at epsilon rate of 0. This is because after convergence, 0 epsilon would mean strictly sticking to the policy that will eventually yield a better reward.

4. As we increase the epsilon rate to 0.5 and 0.9, it leads to too much random walks rather than fixed path walk that results in large negative rewards.

Varying the learning rate (alpha):

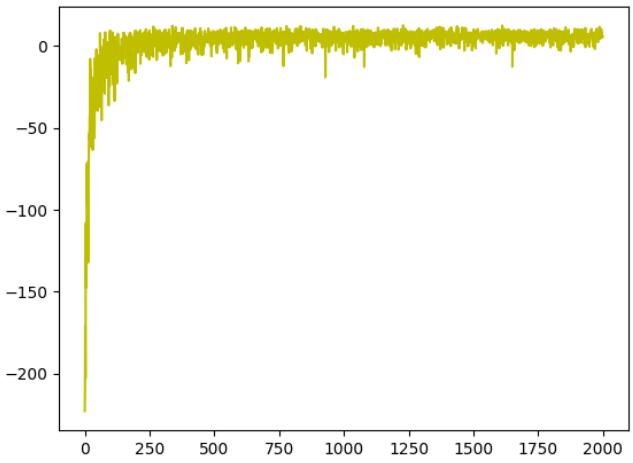

Figure 12: Q-learning with fixed epsilon and alpha of 0.1

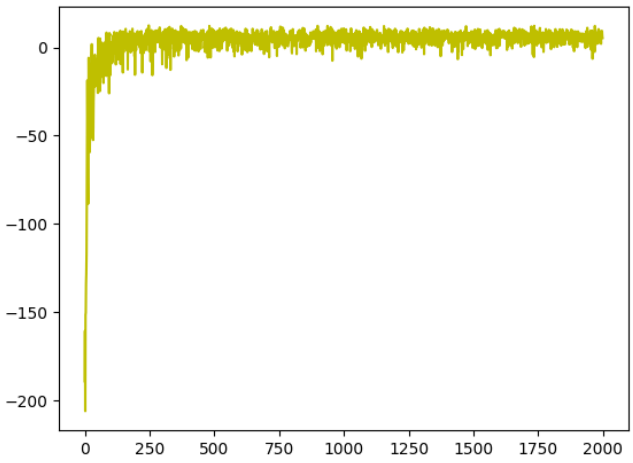

Figure 13: Q-learning with fixed epsilon and alpha of 0.2

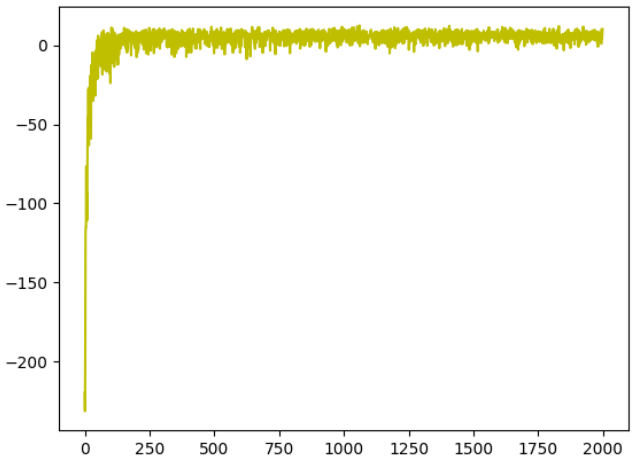

Figure 14: Q-learning with fixed epsilon and alpha of 0.3

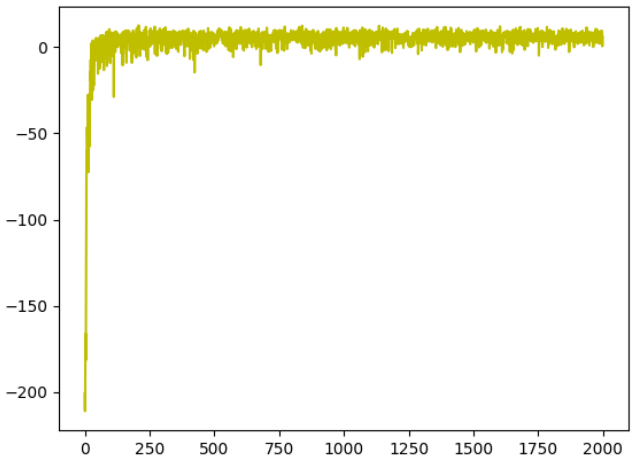

Figure 15: Q-learning with fixed epsilon and alpha of 0.4

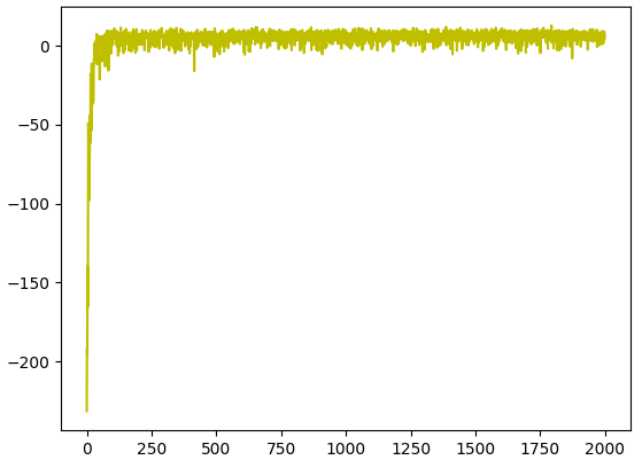

Figure 16: Q-learning with fixed epsilon and alpha of 0.5

Following are the observations from the graphs:

1. The graph remains more or less similar in all the cases.
2. Convergence does not get affected from changing the alpha.
3. Sum of averaged rewards for all the methods are as follows:
   - Alpha = 0.1: 4813.306131904512
   - Alpha = 0.2: 7269.061812133985
   - Alpha = 0.3: 7699.056626136997
   - Alpha = 0.4: 7694.694002411031
   - Alpha = 0.5: 8219.300872677153
4. Alpha accounts for how much change we incorporate from the learning and how much do we retain the old value.
5. High alpha would mean higher priority to learning and lesser to the old values. Increasing alpha makes the initial rewards as volatile but it also leads to faster convergence. Thus, alpha = 0.5 is yielding a better reward. Note that increasing alpha further to 0.8 or 0.9 would result in even better result but initial disturbances could be more.

3 Work by different functions

In this section, we will define what each of the functions in the code is doing:

3.1 simulate()

This is the function that is created to evaluate the learning and the policy. It generates random locations for the taxi and passenger and runs the traversal according to different policies and generates the total reward value and also gives the total number of actions taken to reach the goal.

3.2 pActions()

This is a helper function that does the probabilistic picking of destination state. According to problem specified, we need to consider that each navigation action succeeds with a probability of 0.85 and moves in a random other direction with probability 0.15.

3.3 sarsa() and q_learning()

These are the main function where updation of q-values and learning takes place. Here, the loop runs 500 times and tries to update the q-values and bring it closer to the optimal value. Apart from it, the equations of Q-Learning and SARSA are implemented in these functions.

3.4 isSafe()

This function looks at whether the move that we are trying to do is feasible or not. By feasibility, we check the border condition and the wall condition. If there is no obstacle (wall or border), then it returns true and false otherwise.

3.5 reward()

This is one of the most important functions. This is the function that governs the preference for a state and actions. Via this function, the information of destination propagates and is reflected in the q-values.a