COL333 A3 document:

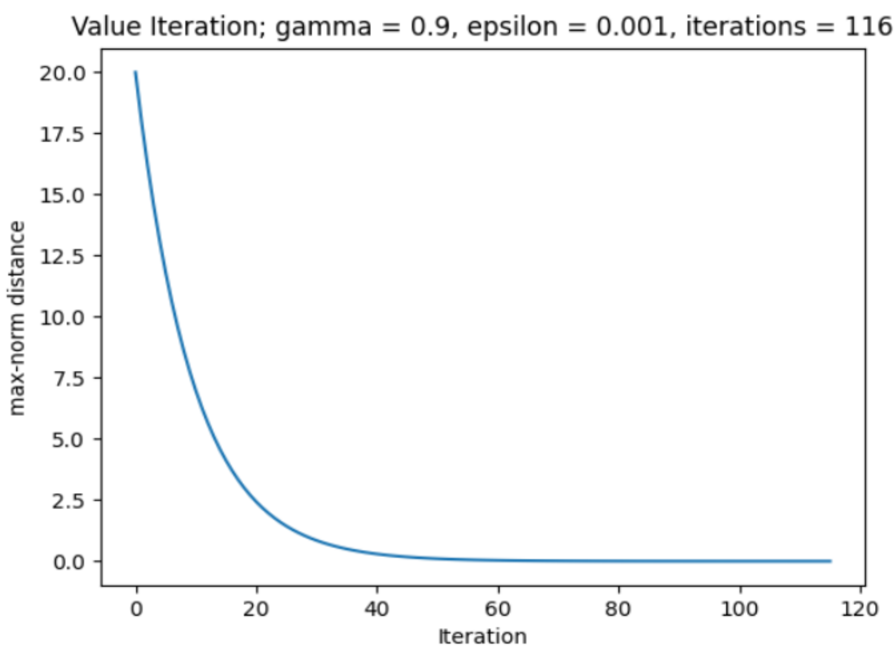
-Dhairya Gupta(2019CS50428), Ramneet Singh(2019CS50445)

Part(A): Computing Policies:

1(a,b,c): The Taxi problem has been implemented as a Markov Decision Process(MDP) with states defined by the TaxiPosition, Passenger position, boolean inTaxi(to check if the passenger is inside the taxi). In a 5x5 grid, there are 650 possible states(25*25 for not inTaxi + 25 for inTaxi) as when inside, both positions are equal.

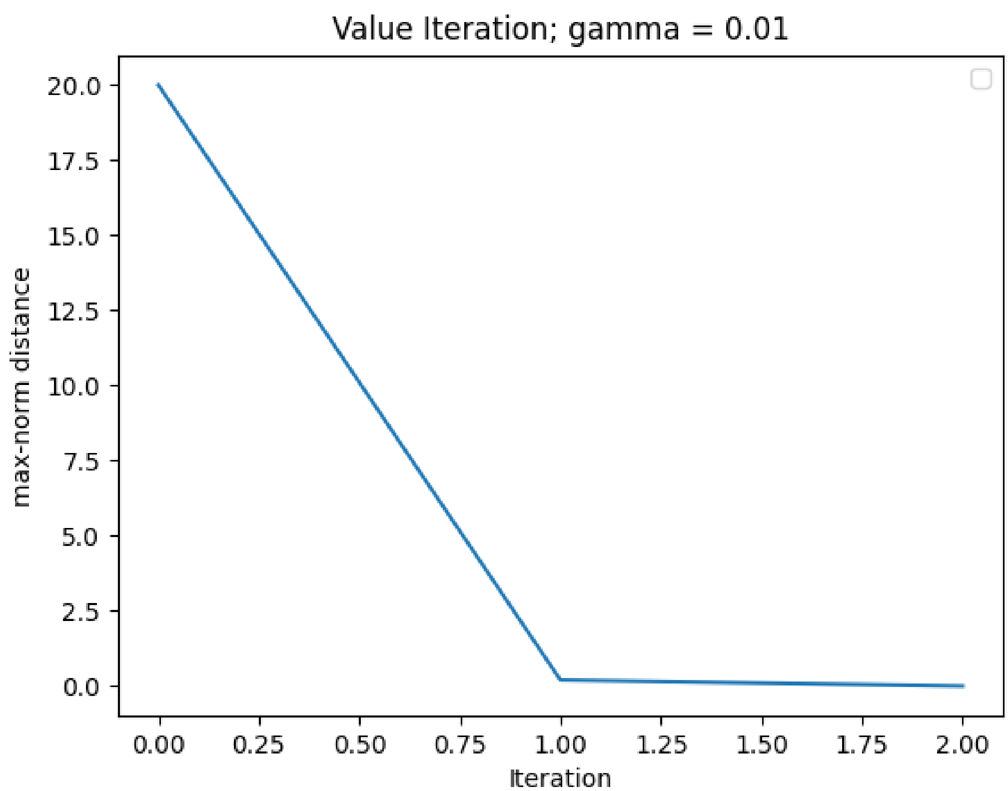
The initialization of the problem requires a starting Taxi depot(or position), targetDepot and starting Passenger Depot. Each (state, action) results in a transition to the next state and a reward.
At each location, the taxi can attempt to PICKUP, PUTDOWN deterministically, or go N,E,W,S. Actual motion action is one of (N,E,W,S) with probability 0.85 of being equal to attempted motion action, or 0.05 for other 3 directions. An actual motion action toward a wall leads to no change in state. Default initial State chosen is {TaxiPos: (0, 4), PasPos: (0, 0), inTaxi: False}

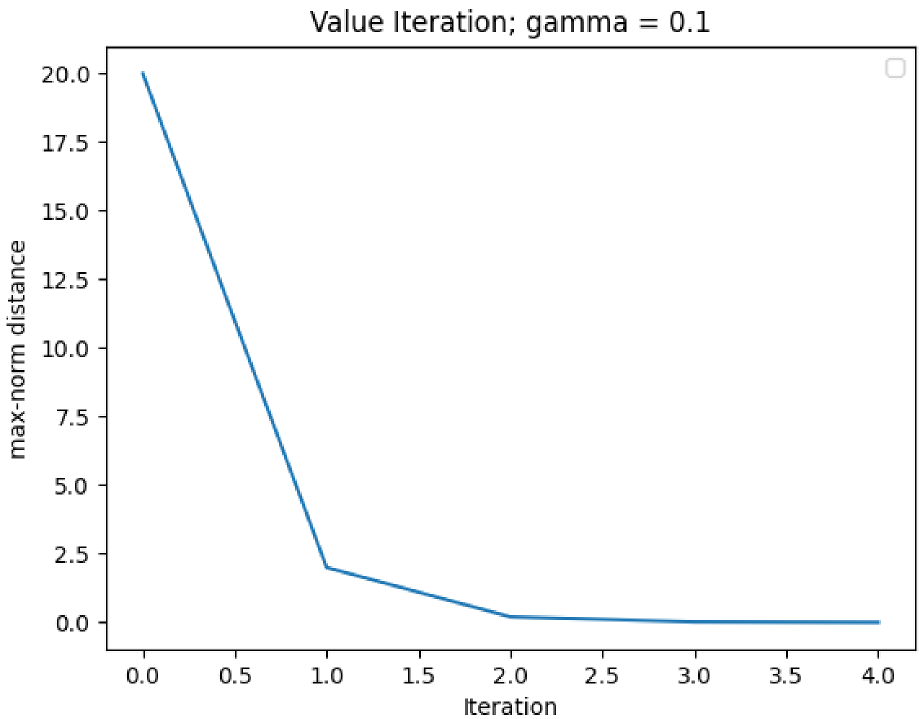
2.(a) putting epsilon = 1e-3 and gamma(discount factor) = 0.9, our algorithm converges in 116 iterations.



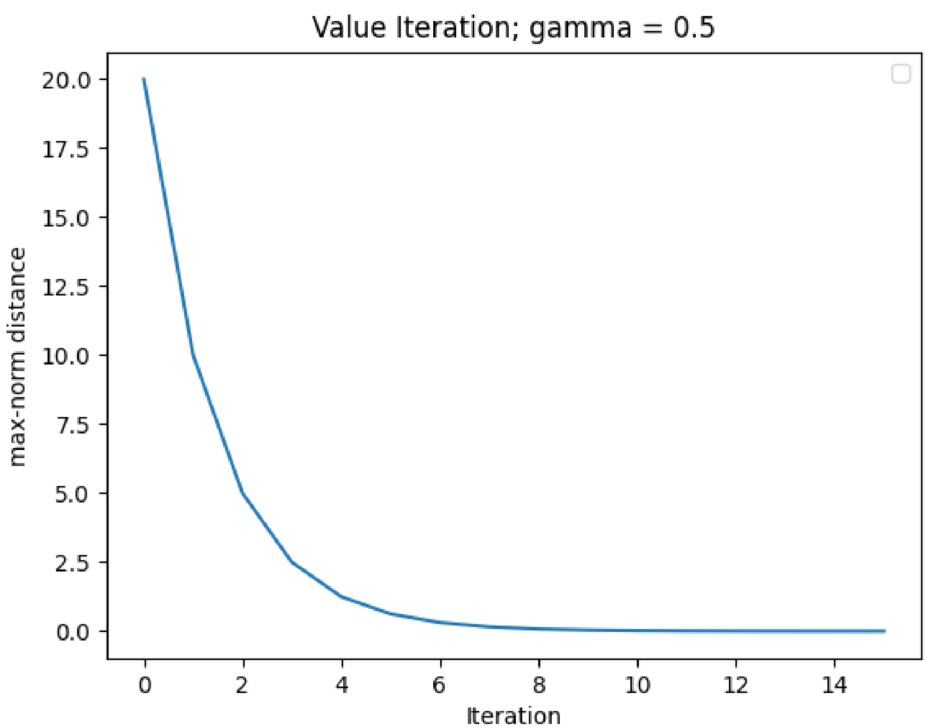
(b) Epsilon is kept constant = 1e-3, iterations taken to converge are given below:
For gamma = 0.01, Iterations = 3
For gamma = 0.1, Iterations = 5
For gamma = 0.5, Iterations = 16
For gamma = 0.8, Iterations = 52
For gamma = 0.99, Iterations = 1444

Above observations can be explained as convergence criterion is delta < epsilon*(1 - gamma)/gamma, where delta is the maximum change in utility in consecutive iterations for all states. So, higher gamma would converge at low values of delta, and therefore have higher number of iterations. We also see max_norm_distance decreases as we reach optimal utility values.

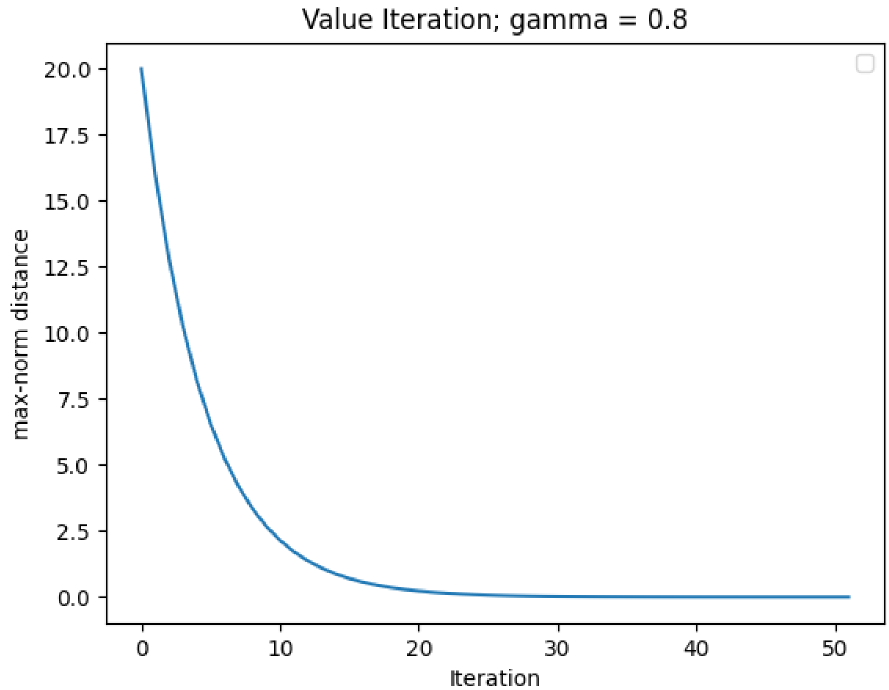


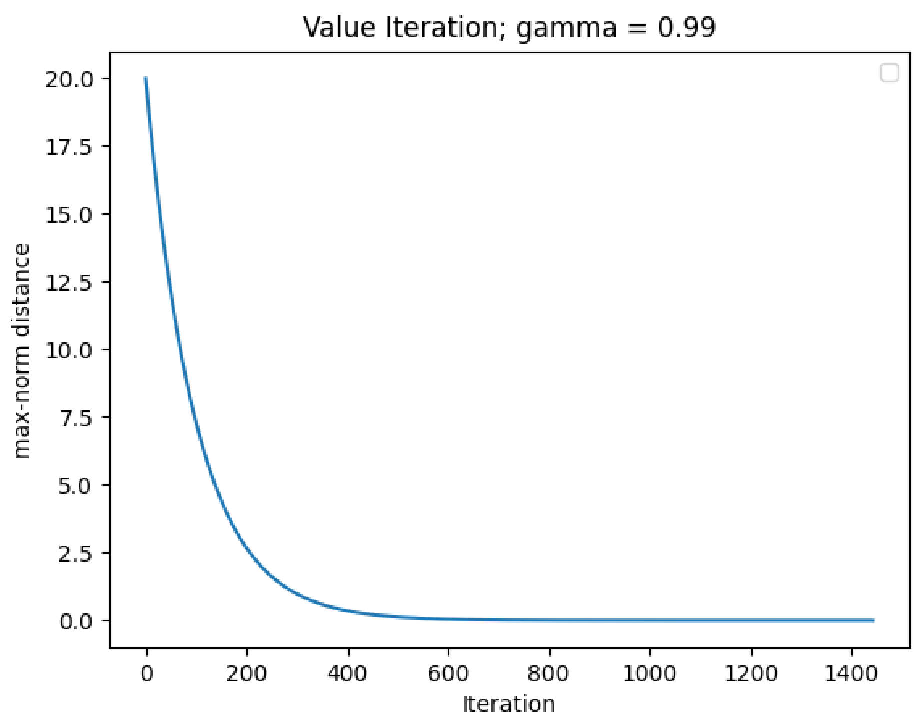


Value Iteration; gamma = 0.1
max-norm distance
Iteration
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0



Value Iteration; gamma = 0.5
max-norm distance
Iteration
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
2
4
6
8
10
12
14


Value Iteration; gamma = 0.8
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
max-norm distance
0
10
20
30
40
50
Iteration


Value Iteration; gamma = 0.99
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
max-norm distance
0
200
400
600
800
1000
1200
1400
Iteration

(c)
Grid diagram for reference:

```
+ --- --- --- --- --- +
|  0   0 | 0   0   0  |
+                     +
|  0   0 | 0   0   0  |
+                     +
|  0   0   0   0   0  |
+                     +
|  0 | 0   0 | 0   0  |
+                     +
|  0 | 0   0 | 0   0  |
+ --- --- --- --- --- +
```

We run value iteration with values of gamma = 0.1, and gamma = 0.99. TargetDepot was fixed as ‘G’. All possible combinations of remaining depots were chosen as Initial Passenger and Initial Taxi Position. For initial TaxiDepot = ‘R’ and initial PassengerDepot = ‘Y’, gamma = 0.1, we get the following sequence:

TaxiStartDepot: ('R', (0, 4)); PassengerStartDepot: ('Y', (0, 0)); DestinationDepot: ('G', (4, 4)); gamma = 0.1

At t = 0, state --> TaxiPos: (0, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 1, state --> TaxiPos: (0, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 2, state --> TaxiPos: (0, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 3, state --> TaxiPos: (0, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 4, state --> TaxiPos: (0, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 5, state --> TaxiPos: (0, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 6, state --> TaxiPos: (0, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 7, state --> TaxiPos: (0, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 8, state --> TaxiPos: (0, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 9, state --> TaxiPos: (1, 3), PasPos: (0, 0), InTaxi: False

Action Taken: N
Reward: -1

At t = 10, state --> TaxiPos: (1, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 11, state --> TaxiPos: (1, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 12, state --> TaxiPos: (1, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 13, state --> TaxiPos: (1, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 14, state --> TaxiPos: (1, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 15, state --> TaxiPos: (1, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 16, state --> TaxiPos: (1, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 17, state --> TaxiPos: (1, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 18, state --> TaxiPos: (1, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 19, state --> TaxiPos: (1, 3), PasPos: (0, 0), InTaxi: False
Action Taken: N
Reward: -1

At t = 20, state --> TaxiPos: (1, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

Clearly, value iteration algorithm has not converged to an optimal utilities for gamma = 0.1, the taxi can be seen oscillating and not moving towards the passenger.

For gamma = 0.99

TaxiStartDepot: ('R', (0, 4)); PassengerStartDepot: ('Y', (0, 0)); DestinationDepot: ('G', (4, 4)); gamma = 0.99

At t = 0, state --> TaxiPos: (0, 4), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 1, state --> TaxiPos: (0, 3), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 2, state --> TaxiPos: (0, 2), PasPos: (0, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 3, state --> TaxiPos: (0, 1), PasPos: (0, 0), InTaxi: False
Action Taken: S

Reward: -1

At t = 4, state --> TaxiPos: (0, 0), PasPos: (0, 0), InTaxi: False
Action Taken: PICKUP
Reward: -1

At t = 5, state --> TaxiPos: (0, 0), PasPos: (0, 0), InTaxi: True
Action Taken: N
Reward: -1

At t = 6, state --> TaxiPos: (0, 1), PasPos: (0, 1), InTaxi: True
Action Taken: N
Reward: -1

At t = 7, state --> TaxiPos: (0, 2), PasPos: (0, 2), InTaxi: True
Action Taken: E
Reward: -1

At t = 8, state --> TaxiPos: (1, 2), PasPos: (1, 2), InTaxi: True
Action Taken: E
Reward: -1

At t = 9, state --> TaxiPos: (2, 2), PasPos: (2, 2), InTaxi: True
Action Taken: N
Reward: -1

At t = 10, state --> TaxiPos: (2, 3), PasPos: (2, 3), InTaxi: True
Action Taken: E
Reward: -1

At t = 11, state --> TaxiPos: (3, 3), PasPos: (3, 3), InTaxi: True
Action Taken: N
Reward: -1

At t = 12, state --> TaxiPos: (3, 4), PasPos: (3, 4), InTaxi: True
Action Taken: E
Reward: -1

At t = 13, state --> TaxiPos: (4, 4), PasPos: (4, 4), InTaxi: True
Action Taken: PUTDOWN
Reward: 20

At t = 14, state --> TaxiPos: (4, 4), PasPos: (4, 4), InTaxi: False
END

We can see at the end of 14 steps, the values have led the taxi to the final state in which the passenger is put down. In this case, coincidentally, random variations are not observed, but another initial state given below shows random noise as well.

TaxiStartDepot: ('R', (0, 4)); PassengerStartDepot: ('B', (3, 0)); DestinationDepot: ('G', (4, 4)); gamma = 0.99

At t = 0, state --> TaxiPos: (0, 4), PasPos: (3, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 1, state --> TaxiPos: (0, 3), PasPos: (3, 0), InTaxi: False
Action Taken: E
Reward: -1

At t = 2, state --> TaxiPos: (1, 3), PasPos: (3, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 3, state --> TaxiPos: (1, 2), PasPos: (3, 0), InTaxi: False

Action Taken: E
Reward: -1

At t = 4, state --> TaxiPos: (2, 2), PasPos: (3, 0), InTaxi: False
Action Taken: E
Reward: -1

At t = 5, state --> TaxiPos: (3, 2), PasPos: (3, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 6, state --> TaxiPos: (3, 1), PasPos: (3, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 7, state --> TaxiPos: (3, 2), PasPos: (3, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 8, state --> TaxiPos: (3, 1), PasPos: (3, 0), InTaxi: False
Action Taken: S
Reward: -1

At t = 9, state --> TaxiPos: (3, 0), PasPos: (3, 0), InTaxi: False
Action Taken: PICKUP
Reward: -1

At t = 10, state --> TaxiPos: (3, 0), PasPos: (3, 0), InTaxi: True
Action Taken: N
Reward: -1

At t = 11, state --> TaxiPos: (3, 1), PasPos: (3, 1), InTaxi: True
Action Taken: N
Reward: -1

At t = 12, state --> TaxiPos: (3, 2), PasPos: (3, 2), InTaxi: True
Action Taken: N
Reward: -1

At t = 13, state --> TaxiPos: (3, 3), PasPos: (3, 3), InTaxi: True
Action Taken: N
Reward: -1

At t = 14, state --> TaxiPos: (3, 4), PasPos: (3, 4), InTaxi: True
Action Taken: E
Reward: -1

At t = 15, state --> TaxiPos: (4, 4), PasPos: (4, 4), InTaxi: True
Action Taken: PUTDOWN
Reward: 20

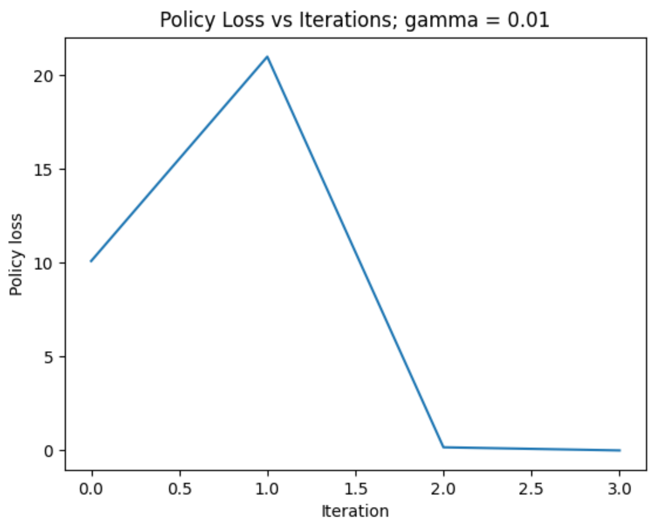
At t = 16, state --> TaxiPos: (4, 4), PasPos: (4, 4), InTaxi: False
END

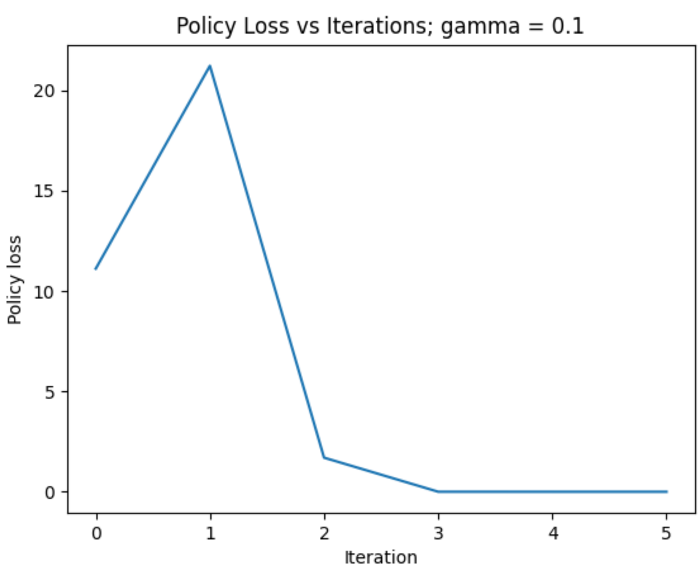
From t=6 to t=7, non-intended action occurs showing stochasticity in the environment model. Again, we see the taxi successfully picks up and drops passenger despite random variations being experienced.

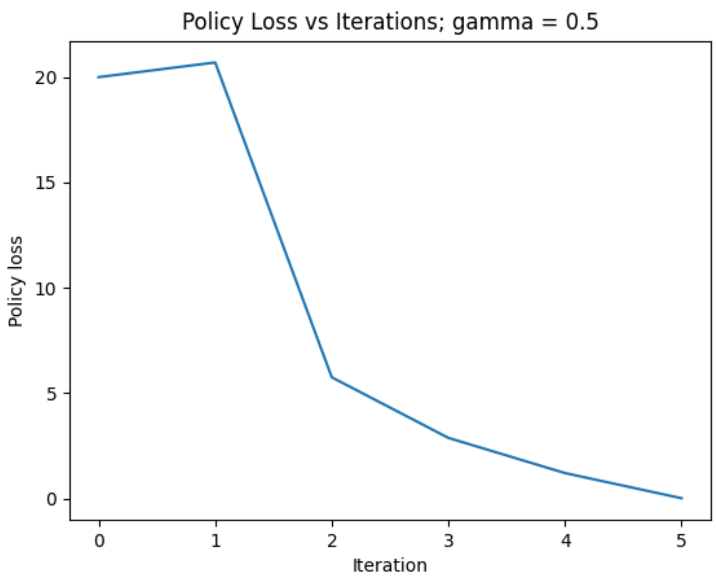
3. (a) For policy iteration, we apply linear algebra/iterative policy evaluation followed by policy improvement. During experimentation, we found linear algebra solvers to be faster for higher values of gamma as calculation of utilities in policy evaluation is done in a single step while it takes multiple iterations during iterative. For higher values of gamma, iterations are more due to convergence being later(as explained in 2).

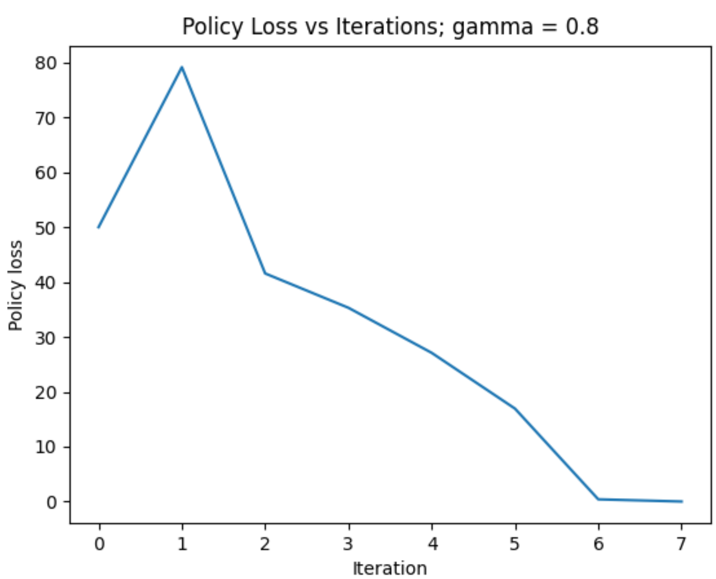
(b) We plotted policy iteration vs loss for multiple values of gamma = [0.01, 0.1, 0.5, 0.8, 0.9]. For all values of gamma, policy iteration converged in 5-6 iterations, much lower than value iterations. However, we observe that policy loss values for gamma = 0.99 were much higher indicating a steep change in randomized policy. This can be explained by the higher contribution of previous values due to large discount factor leading to large change in policy values.

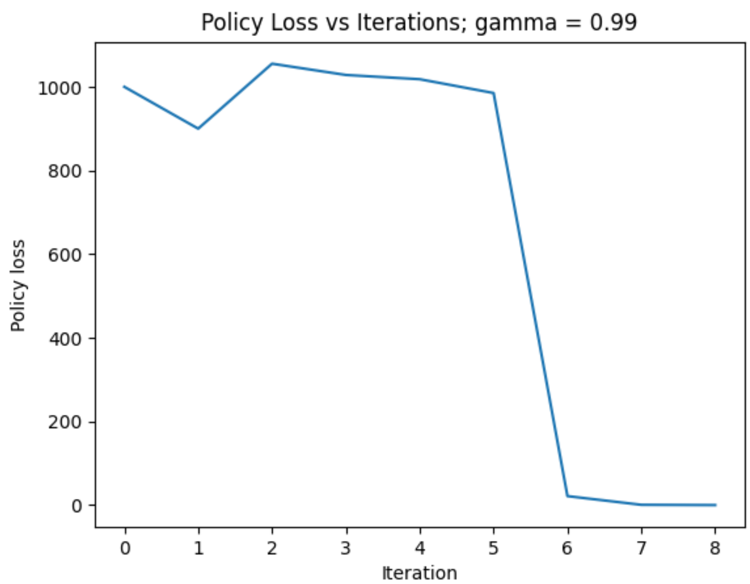
Plots for policy_loss vs iterations are:






Policy Loss vs Iterations; gamma = 0.5
Policy loss
Iteration
0
5
10
15
20
0
1
2
3
4
5


Policy Loss vs Iterations; gamma = 0.8
Policy loss
Iteration
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6
7

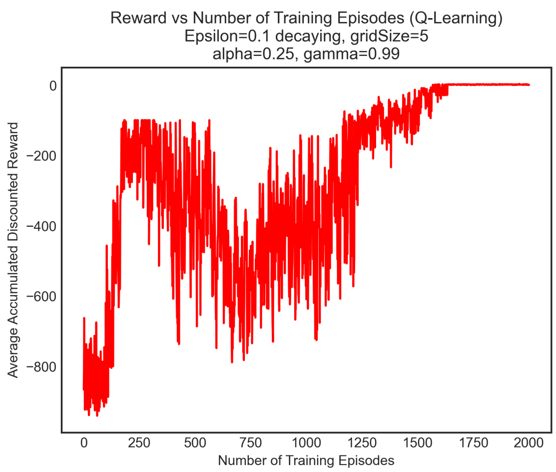


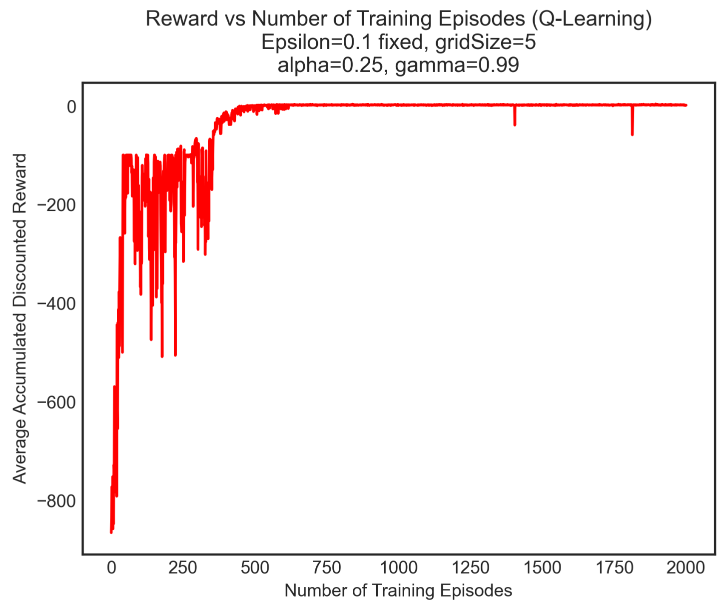
Plots observed are non-steady at different steps, movement of policy is different, hence graph has irregular descent, however policy still finally converges.

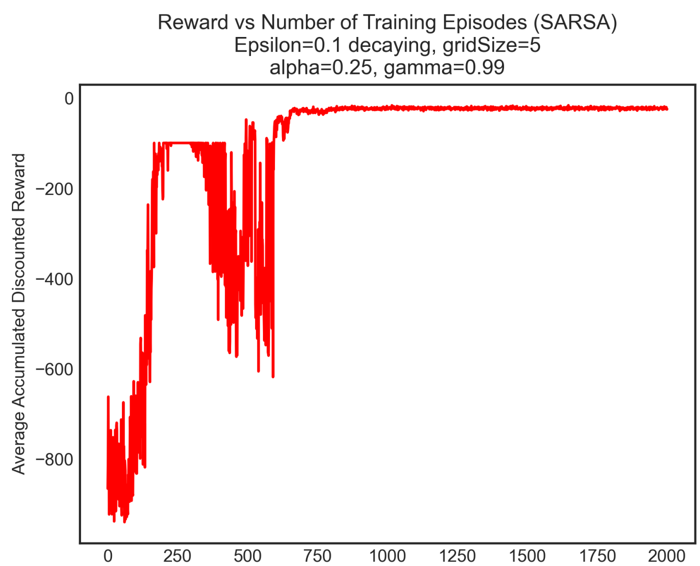
Reinforcement Learning:

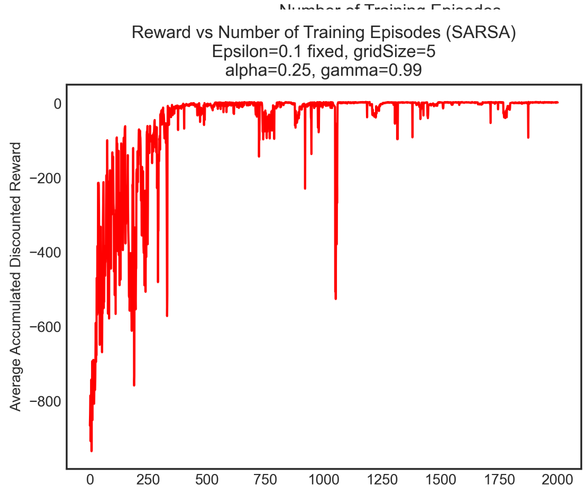
2. We set the destination as (4,4). As the following plots show, introducing a decaying exploration rate slows down convergence. We see that Q-Learning with a fixed rate converges the fastest, and Q-Learning with a decaying rate converges the slowest. All of these roughly converge to the same value of average reward over multiple runs, except SARSA with decaying rate, that converges to a negative value. Their final reward values are:

A. Q-Learning, Fixed : 1.637
B. Q-Learning, Decaying: 1.04
C. SARSA, Fixed: 2.06
D. SARSA, Decaying: -24.29

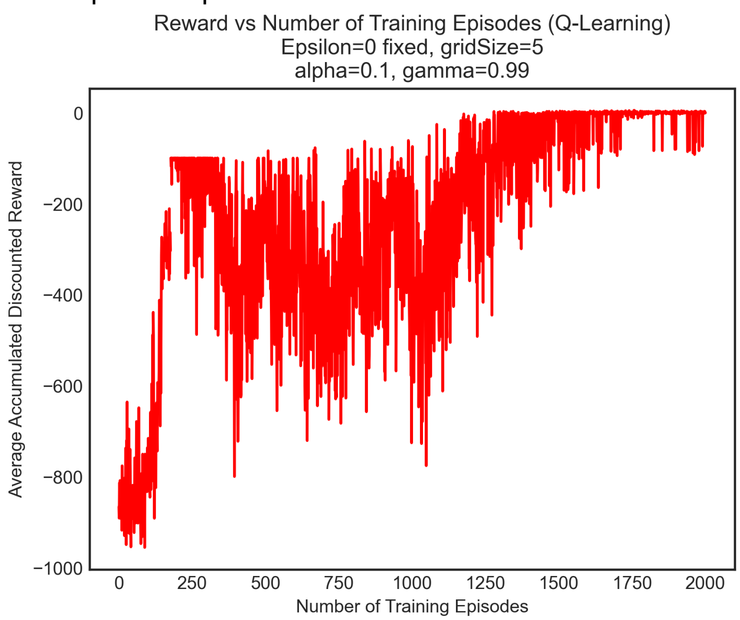


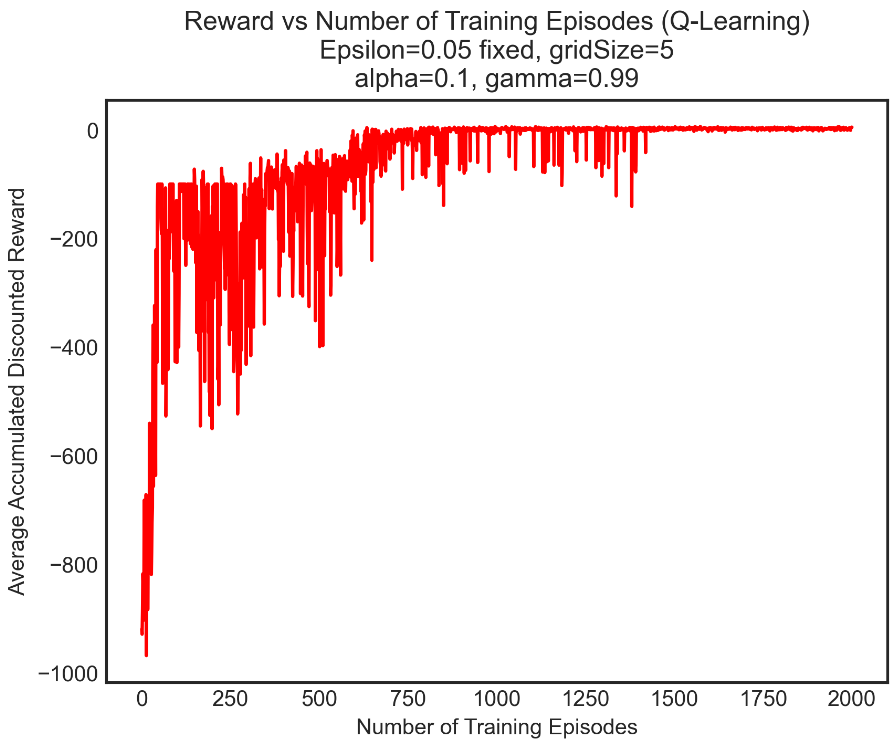

Reward vs Number of Training Episodes (Q-Learning)
Epsilon=0.1 fixed, gridSize=5
alpha=0.25, gamma=0.99
Average Accumulated Discounted Reward
0
−200
−400
−600
−800
0
250
500
750
1000
1250
1500
1750
2000
Number of Training Episodes

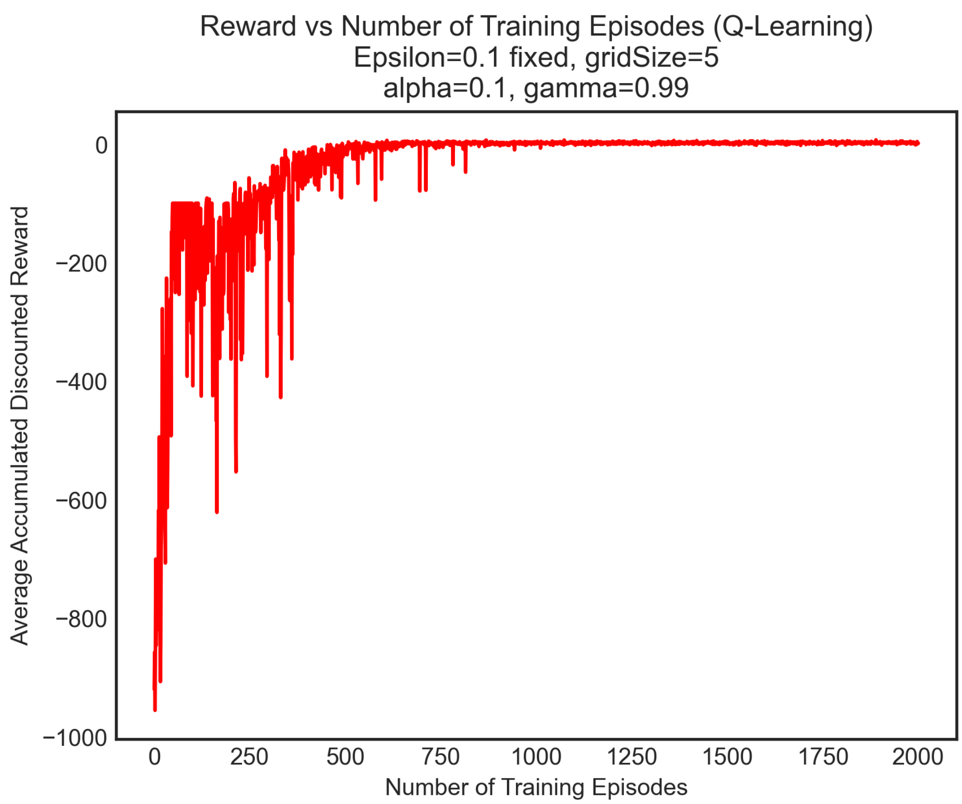

Reward vs Number of Training Episodes (SARSA)
Epsilon=0.1 decaying, gridSize=5
alpha=0.25, gamma=0.99
Average Accumulated Discounted Reward
0
−200
−400
−600
−800
0
250
500
750
1000
1250
1500
1750
2000

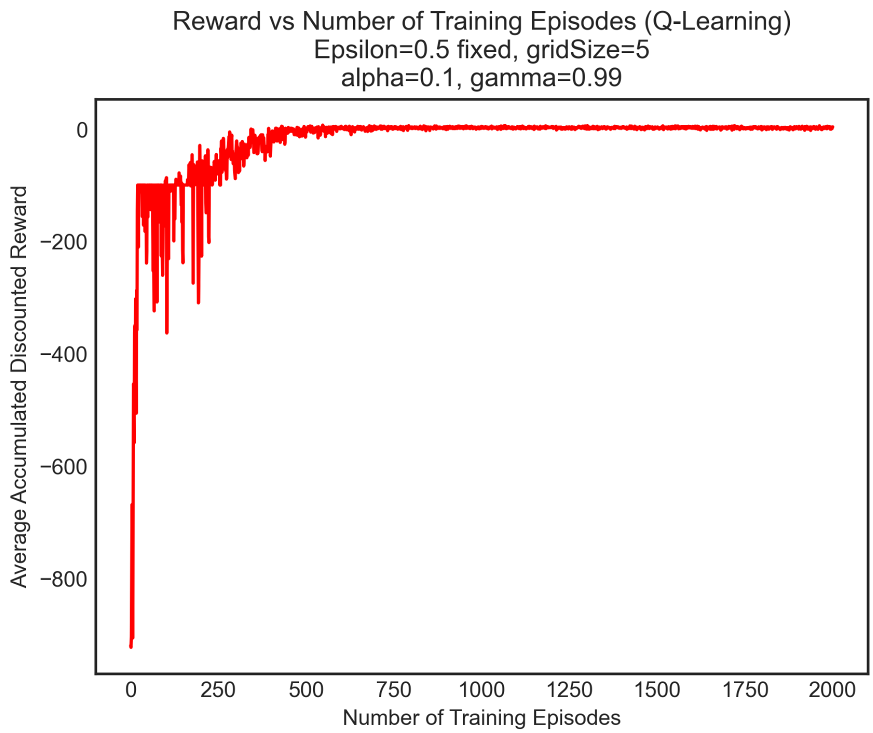

Reward vs Number of Training Episodes (SARSA)
Epsilon=0.1 fixed, gridSize=5
alpha=0.25, gamma=0.99
Average Accumulated Discounted Reward
0
−200
−400
−600
−800
0
250
500
750
1000
1250
1500
1750
2000
Number of Training Episodes

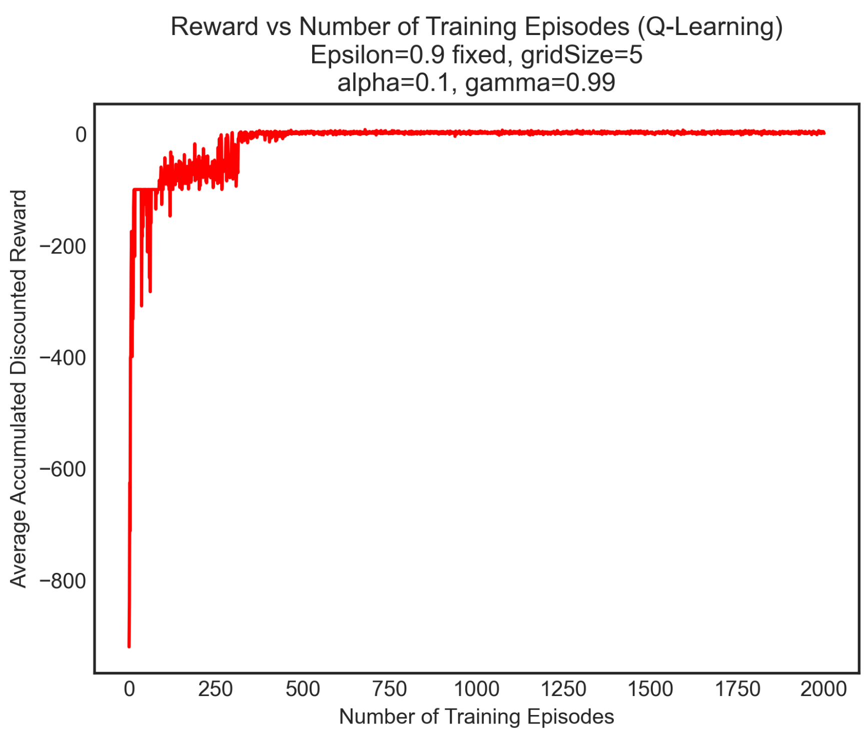
4.Keeping a very low exploration rate means that we are stuck with the initial Q Value estimates, and the resulting policy is nearly random. This results in high stochasticity and poor rewards. On the other hand, keeping a very high exploration rate also means that most of our actions are randomly selected and thus results in high stochasticity of the observed rewards. When we vary the learning rate, we see that a high learning rate places too much importance on the just recently acquired sample over the previous estimates we have learnt. On the other hand, a low learning rate keeps us stuck with our initial estimates too much, and doesn't allow us to use the new samples to update our estimates and learn.



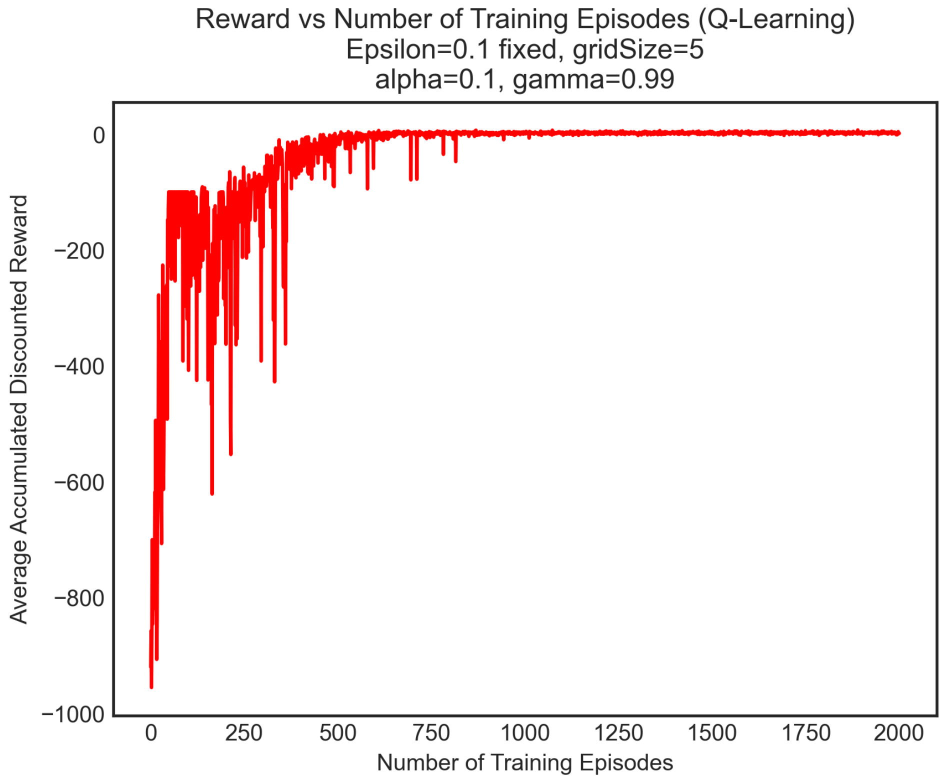

Reward vs Number of Training Episodes (Q-Learning)
Epsilon=0.05 fixed, gridSize=5
alpha=0.1, gamma=0.99
Average Accumulated Discounted Reward
Number of Training Episodes
0
−200
−400
−600
−800
−1000
0
250
500
750
1000
1250
1500
1750
2000

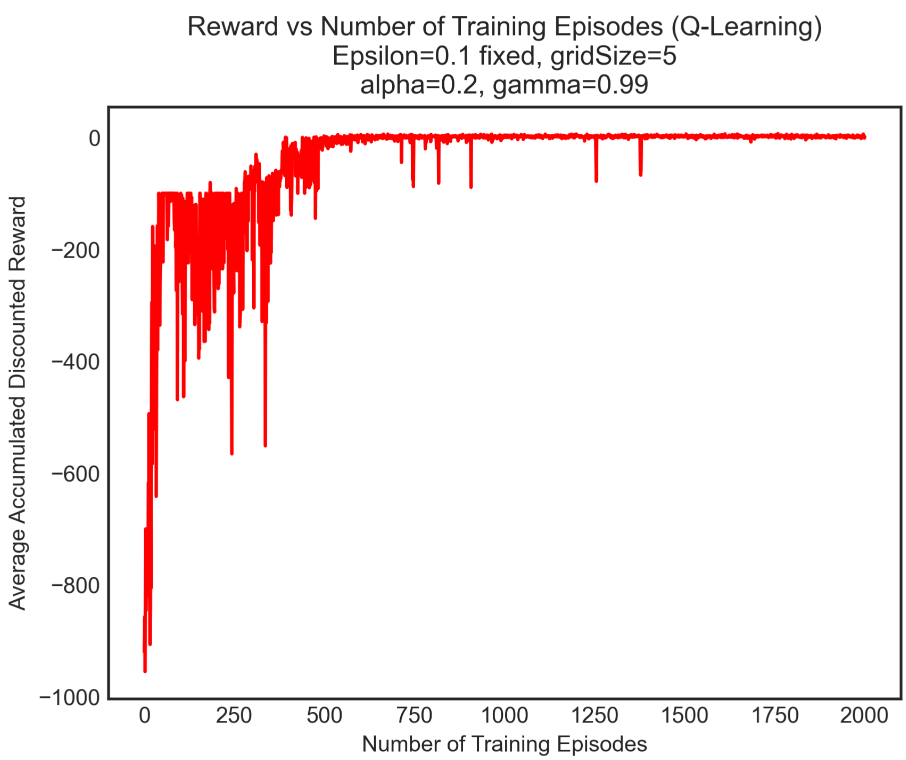

Reward vs Number of Training Episodes (Q-Learning)
Epsilon=0.1 fixed, gridSize=5
alpha=0.1, gamma=0.99
Average Accumulated Discounted Reward
Number of Training Episodes
0
−200
−400
−600
−800
−1000
0
250
500
750
1000
1250
1500
1750
2000

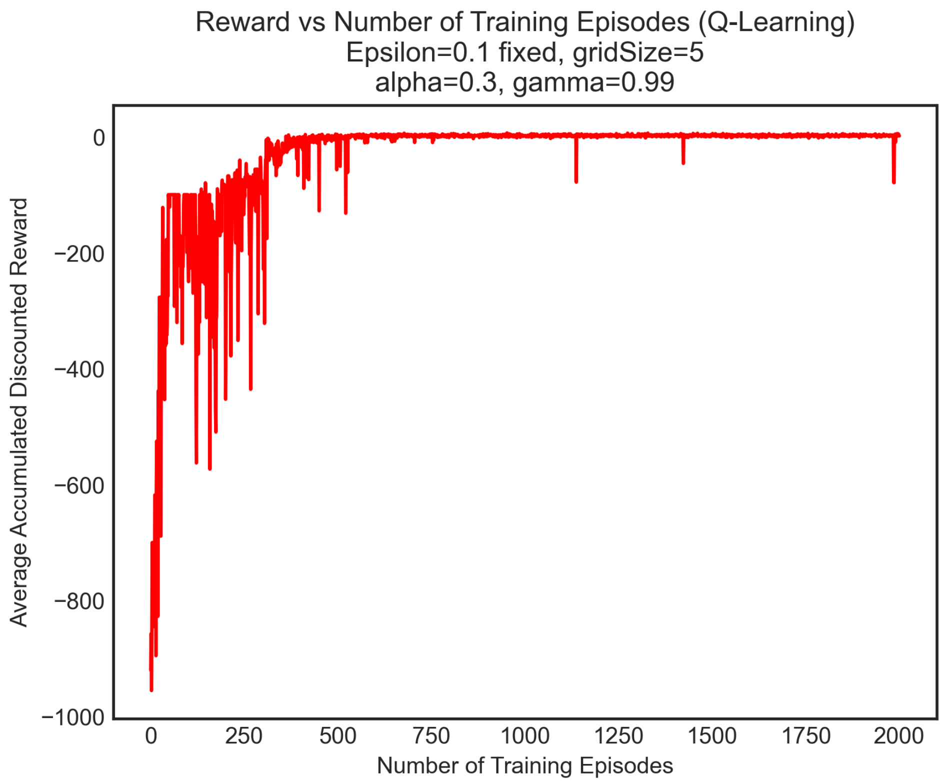

Reward vs Number of Training Episodes (Q-Learning)
Epsilon=0.5 fixed, gridSize=5
alpha=0.1, gamma=0.99
Average Accumulated Discounted Reward
0
−200
−400
−600
−800
0
250
500
750
1000
1250
1500
1750
2000
Number of Training Episodes

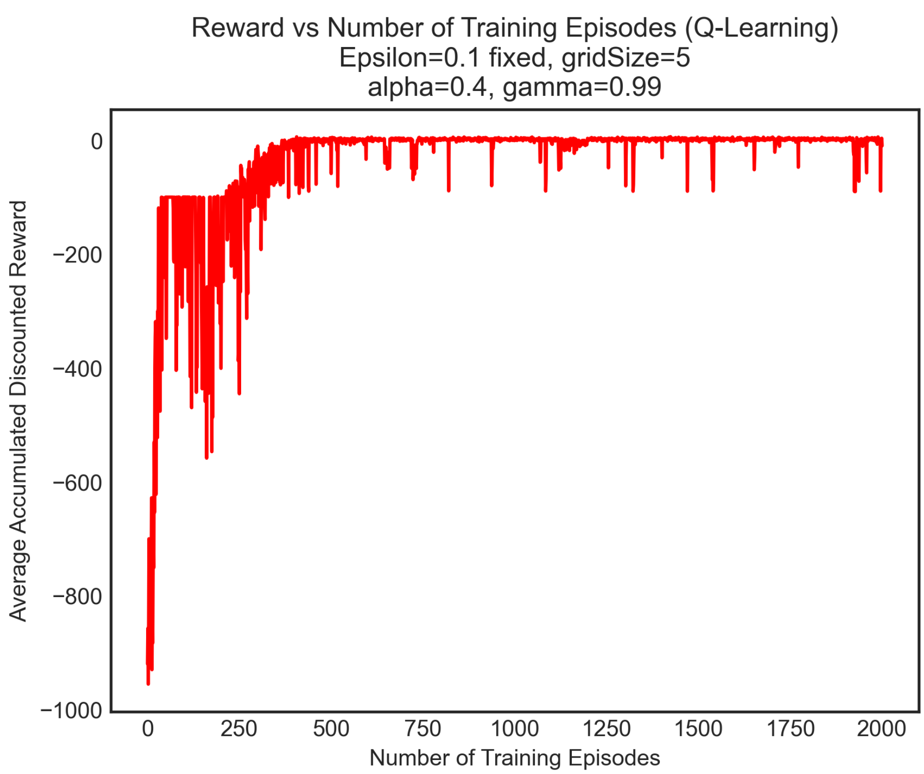

Reward vs Number of Training Episodes (Q-Learning)
Epsilon=0.9 fixed, gridSize=5
alpha=0.1, gamma=0.99
Average Accumulated Discounted Reward
0
−200
−400
−600
−800
0
250
500
750
1000
1250
1500
1750
2000
Number of Training Episodes

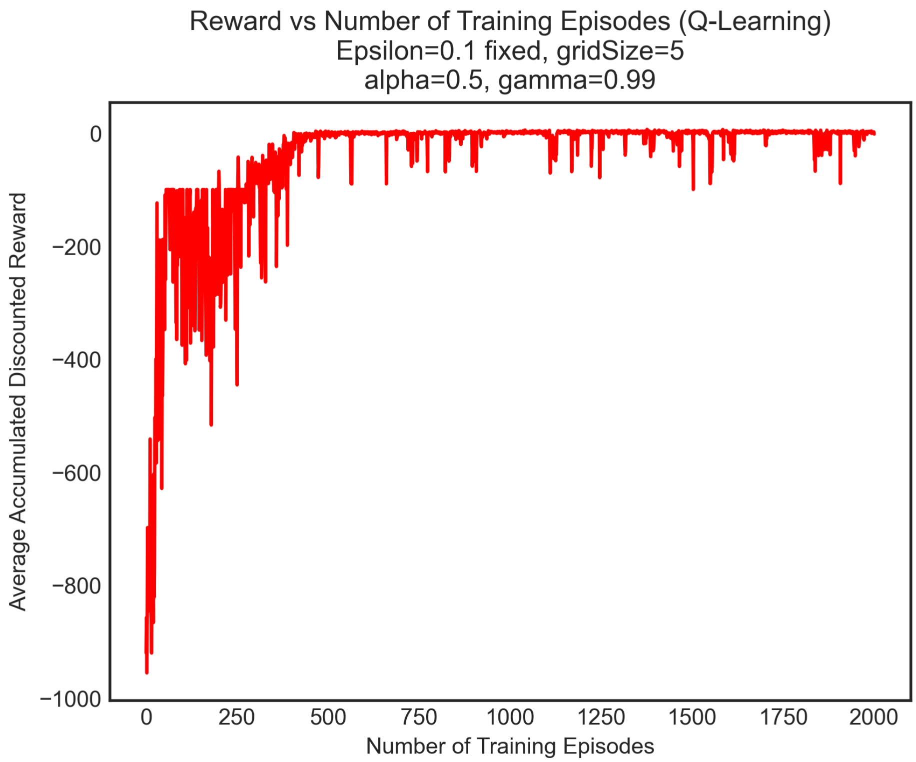
Changing alpha:






Reward vs Number of Training Episodes (Q-Learning)
Epsilon=0.1 fixed, gridSize=5
alpha=0.3, gamma=0.99
Average Accumulated Discounted Reward
0
−200
−400
−600
−800
−1000
0
250
500
750
1000
1250
1500
1750
2000
Number of Training Episodes


Reward vs Number of Training Episodes (Q-Learning)
Epsilon=0.1 fixed, gridSize=5
alpha=0.4, gamma=0.99
Average Accumulated Discounted Reward
0
−200
−400
−600
−800
−1000
0
250
500
750
1000
1250
1500
1750
2000
Number of Training Episodes


Reward vs Number of Training Episodes (Q-Learning)
Epsilon=0.1 fixed, gridSize=5
alpha=0.5, gamma=0.99
Average Accumulated Discounted Reward
0
−200
−400
−600
−800
−1000
0
250
500
750
1000
1250
1500
1750
2000
Number of Training Episodes