CS780: Deep Reinforcement Learning

Assignment #2

Name: Kislay Aditya Oj
Roll NO.: 210524

Solution to Problem 1: Monte-Carlo Control

1. The plot of State-value function of every state is plotted in Figure 1. This is the First Visit Monte Carlo Control (FMVCC) for finding the optimal policy for RME. The signature is same as given in the lecture slides.

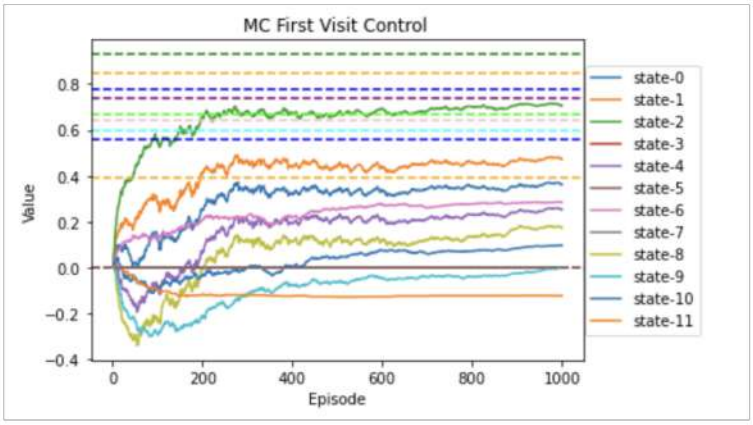

Figure 1: MC-FVC Value estimates vs episodes

2. The plots of State-Action pair Values are plotted in the Figures 2-5. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

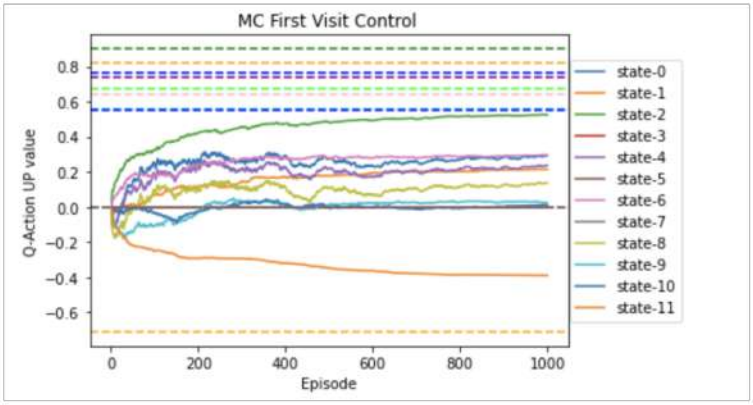

Figure 2: MC-FVC Action UP estimates vs episodes

3. The number of instances taken in this case is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

4. The environment diagram is plotted in figure 6.

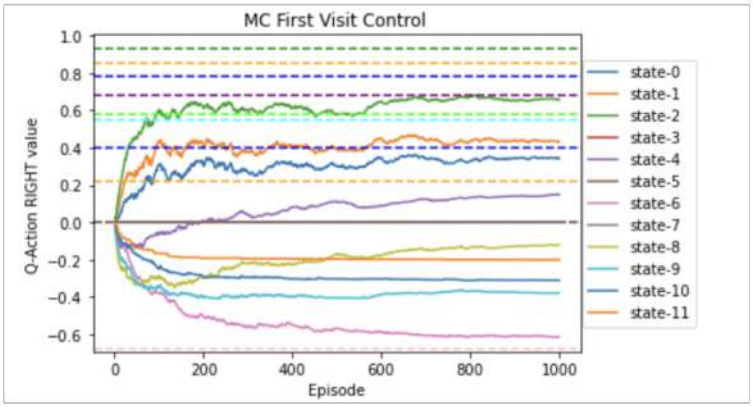

Figure 3: MC-FVC Action RIGHT estimates vs episodes

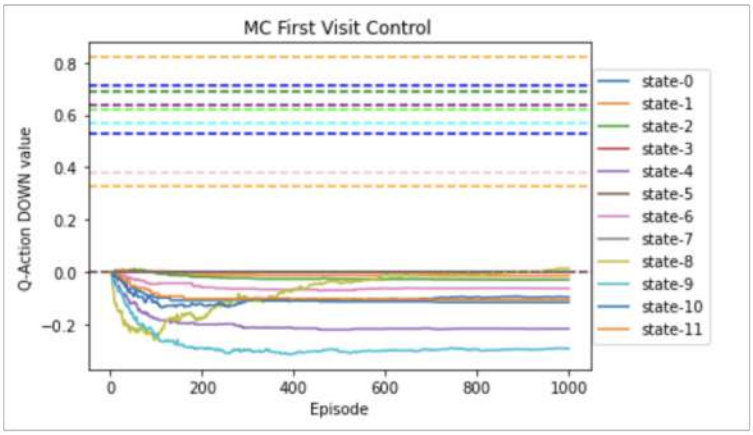

Figure 4: MC-FVC Action DOWN estimates vs episodes

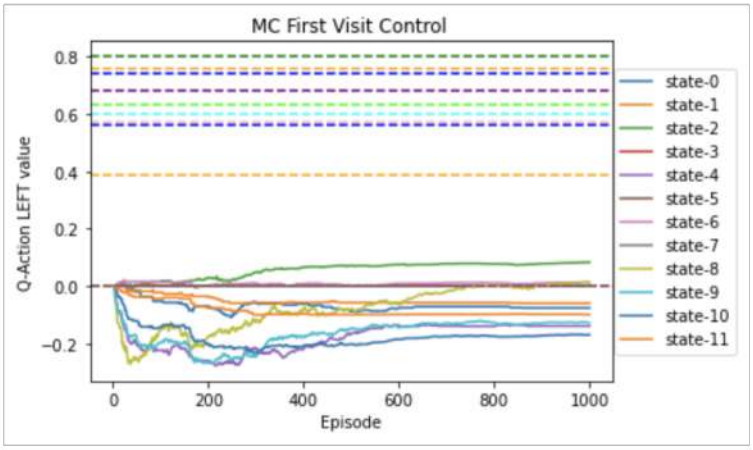

Figure 5: MC-FVC Action LEFT estimates vs episodes

5. The Hyper parameters used in MCFVC are alpha, epsilon , maxSteps , gamma and no of episodes.
   (i). **alpha** - The alpha value taken in the current working code is 0.1 which exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting episodes. Same goes for choice of linear over exponential.
   (ii) **epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.1 which decreases

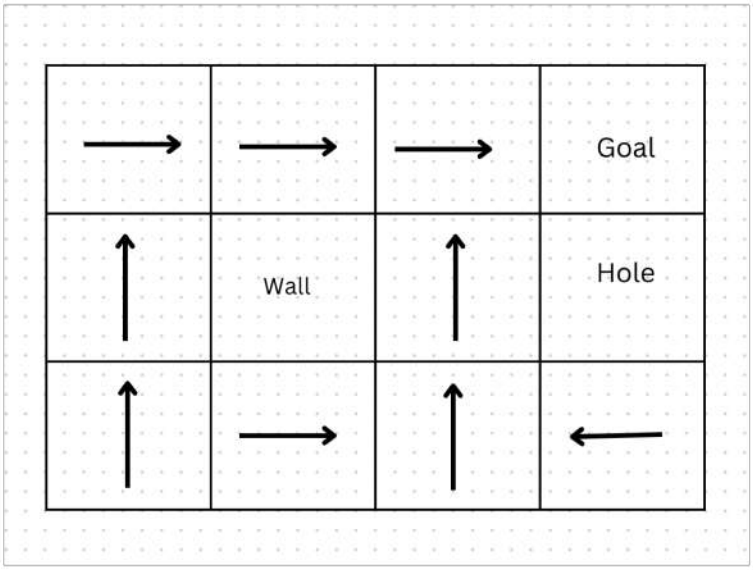

Figure 6: Optimal policy using MCFVV

exponentially to 0.01. Technically , we start with exploration and go towards more greedy approach but we give more chance to greed here as it gives better result and kind of lead us towards Q learning approach. (iii) **maxSteps , gamma and No. of episodes** - The values of all three parameters are taken to be the standard values ie maxsteps to be 100 as there will be rarely a case where the no. of steps in a episode reaches 100. The gamma is specified as 0.99 and no. of episodes taken here is 500 as the value more or less converges in this range.

6. One thing we can observe quickly is that the values fluctuate a lot in initial episodes and takes quite long to reach it's true values . Also in all of the Q functions two actions in Q function has a positive value , which makes sense as it points directly towards the goal state with +1 reward. All the other state which is near to hole has value around -1 as we can end up in hole following the environment dynamics. Most of the values are negative because of the living penalty of -0.04 and only high values are near goal state and those actions are leading directly or indirectly towards holes.

Solution to Problem 2: SARSA (TD Control)

1. The plot of State-value function of every state is plotted in Figure 7. This is SARSA (TD-Control) for finding the optimal policy for RME. The signature is same as given in the lecture slides.

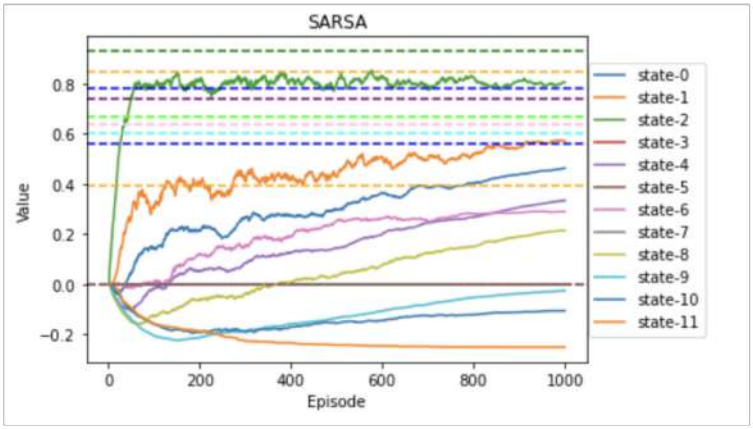

Figure 7: SARSA Value estimates vs episodes

2. The plots of State-Action pair Values for SARSA are plotted in the Figures 8-11. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

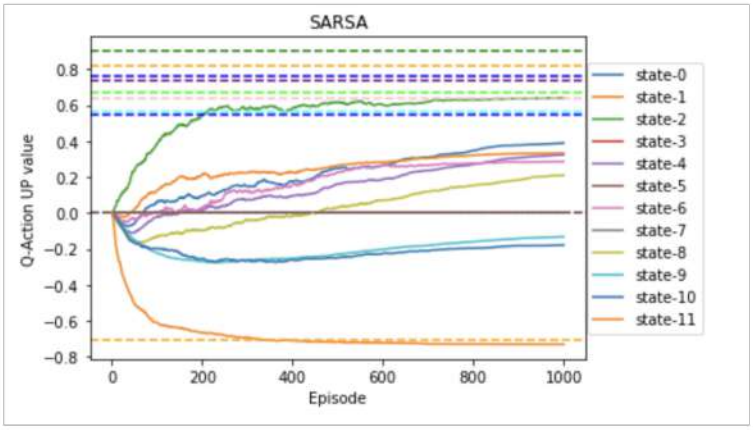

Figure 8: SARSA Action UP estimates vs episodes

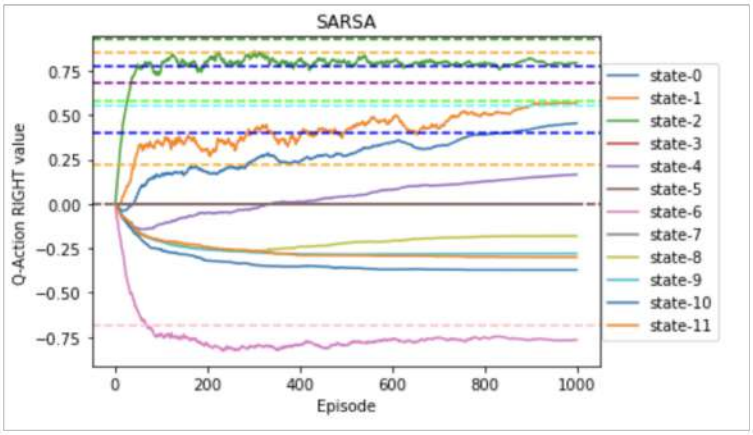

Figure 9: SARSA Action RIGHT estimates vs episodes

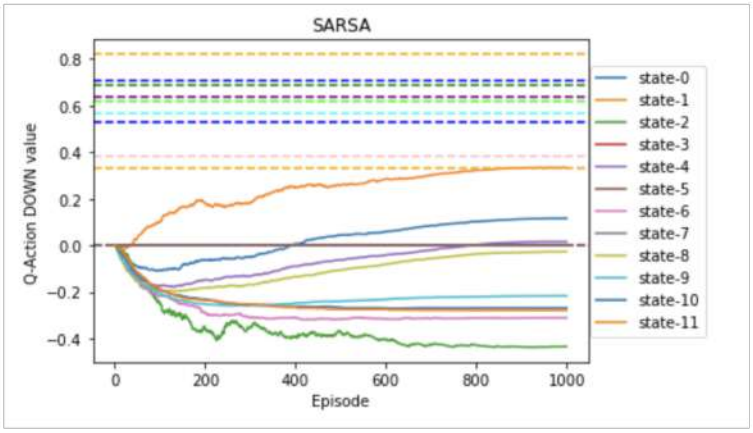

Figure 10: SARSA Action DOWN estimates vs episodes

3. The number of instances taken in this case is same as in last case which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

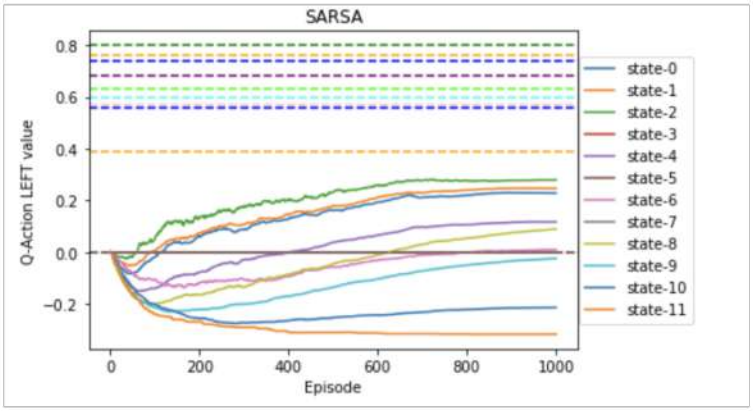

Figure 11: SARSA Action LEFT estimates vs episodes

4. The environment diagram is plotted in figure 6.

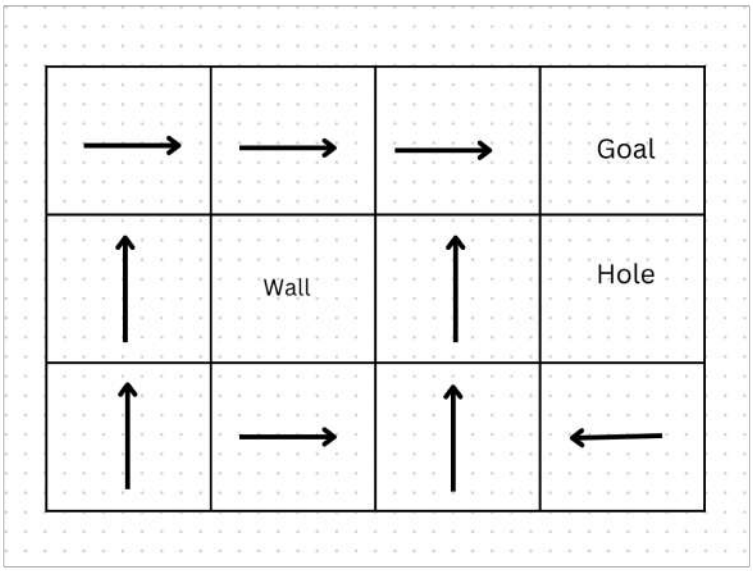

Figure 12: Optimal policy using SARSA

5. The Hyper parameters used in SARSA are alpha, epsilon , gamma and no of episodes.
   (i). **alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting episodes. However the fluctuations here are less than what was in MCFVC.The value is taken same as above to maintain uniformity. Same goes for choice of linear over exponential.
   (ii) **epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.1 which decreases exponentially to 0.01. Technically , we start with exploration and go towards more greedy approach.
   (iii) **gamma and No. of episodes** - The values of both parameters are taken to be the standard values ie maxsteps to be 100 as there will be rarely a case where the no. of steps in a episode reaches 100. The gamma is specified as 0.99 and no. of episodes taken here is 1000 as the values near the hole took almost these many steps to converge. Max Steps is not used as TD algorithm takes only one step at a time.

6. One thing we can observe quickly is the values near the hole state converges to it's true value faster than in MCFVC.But the state near the hole took many iterations. I think the possible reason is due to the ambiguity of environment of going in orthogonal directions which create randomness around the hole state as the agent can lead to the hole more than one way. Also in all of the Q functions only two actions in

Q function has a positive value , which makes sense as it points directly towards the goal state with +1 reward. All the other state which is near to hole has value around -1 as we can end up in hole following the environment dynamics. Most of the values are negative because of the living penalty of -0.04 and only high values are near goal state.

Solution to Problem 3: Q Learning

1. The plot of State-value function of every state is plotted in Figure 13. This is Q-Learning for finding the optimal policy for RME. The signature is same as given in the lecture slides.

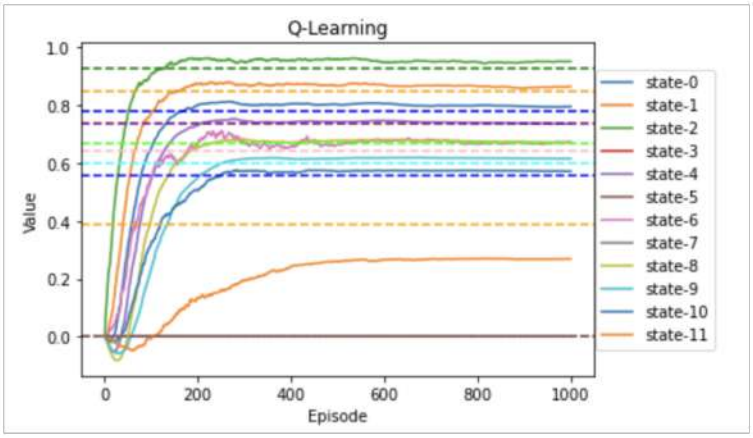

Figure 13: Q Value estimates vs episodes

2. The plots of State-Action pair Values for Q-Learning are plotted in the Figures 14-17. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

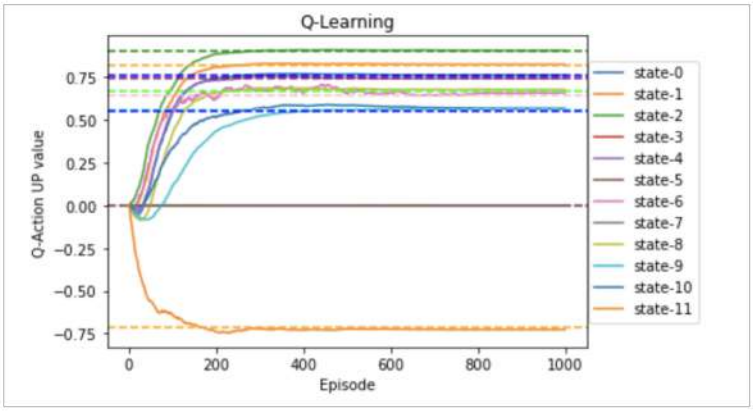

Figure 14: Q Action UP estimates vs episodes

3. The number of instances taken in this case is same as in last case which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

4. The environment diagram is plotted in figure 6.

5. The Hyper parameters used in Q are alpha, epsilon , gamma and no of episodes.

(i). **alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the

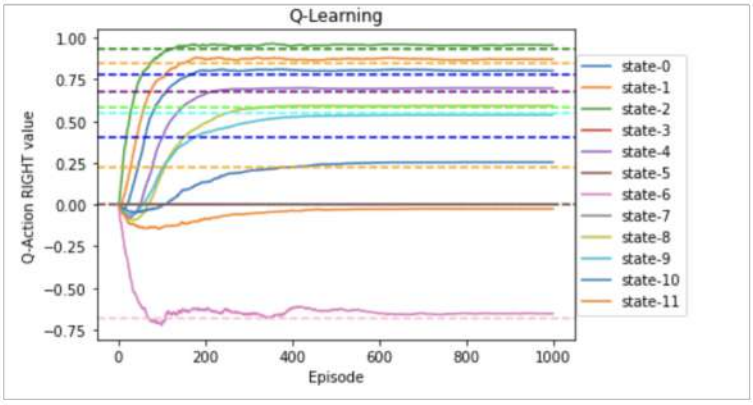

Figure 15: Q Action RIGHT estimates vs episodes

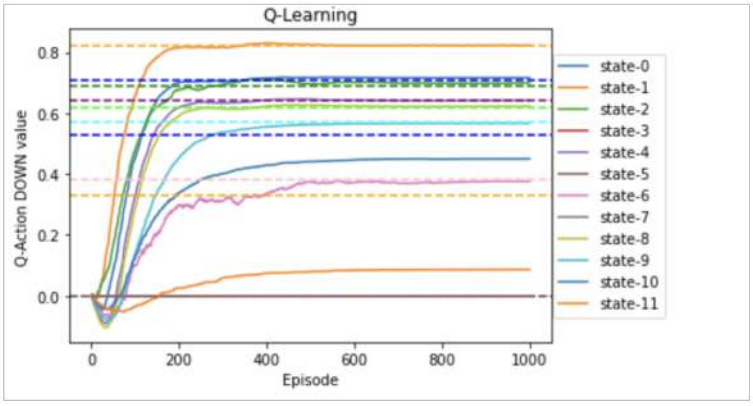

Figure 16: Q Action DOWN estimates vs episodes

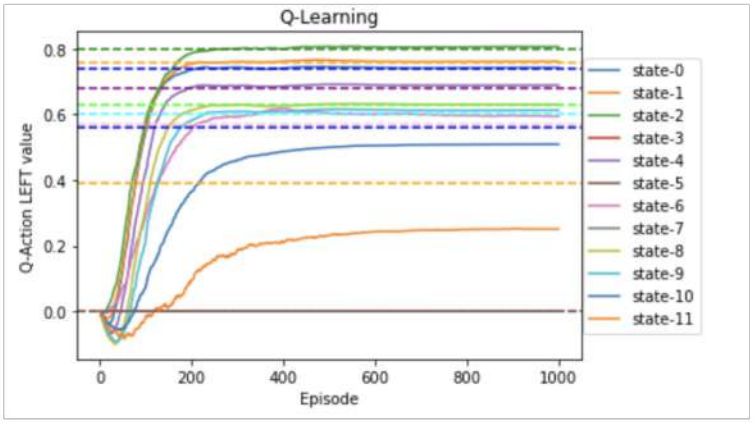

Figure 17: Q Action LEFT estimates vs episodes

value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting episodes. However the fluctuations here are less than what was in MCFVC.The value is taken same as above to maintain uniformity. Same goes for choice of linear over exponential.

(ii) **epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.9 which decreases exponentially to 0.1. Technically , we start with exploration and go towards more greedy approach.

(iii) **Gamma and No. of episodes** - The values of both parameters are taken to be the standard values . The gamma is specified as 0.99 and no. of episodes taken here is 1000 as the values near the hole took almost these many steps to converge. Max Steps is not used as Q algorithm takes only one step at a time

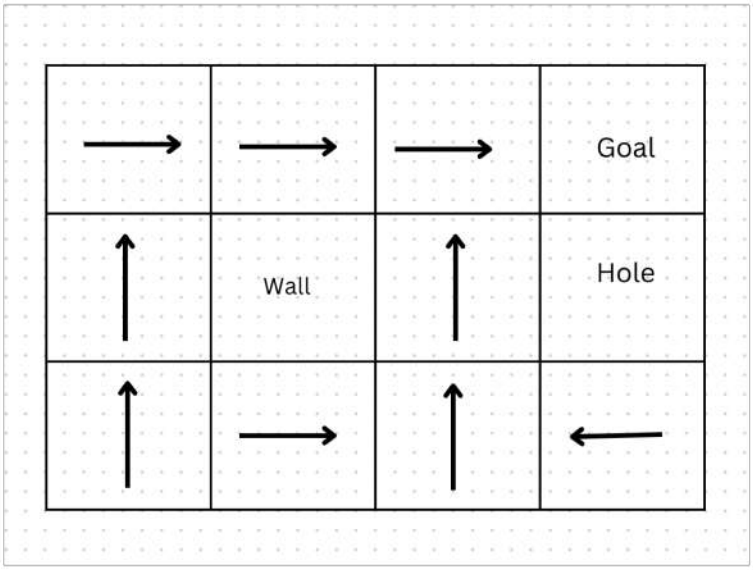

Figure 18: Optimal policy using Q-Learning

and selects the maximum value of the action.

6. We can clearly observe is the values converges to it's true value faster than in any other algorithm see before . We can also see the maximization bias as in the plot that in the initial episodes it quite overshoots the values and then return to it's true values over the episodes. One of the advantage of using this algorithm is there's not much variance and it reaches it's true value faster. Not just that Q functions are almost perfect and achieves its true action values. We can clearly see the distinction between the actions to choose in the given state.

Solution to Problem 4: Double Q Learning

1. The plot of State-value function of every state is plotted in Figure 19. This is Double Q-Learning for finding the optimal policy for RME. The signature is same as given in the lecture slides.

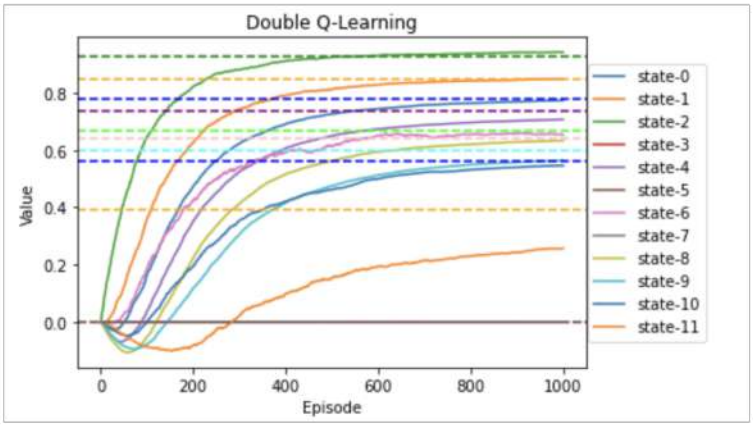

Figure 19: Double Q Value estimates vs episodes

2. The plots of State-Action pair Values for Q-Learning are plotted in the Figures 20-23. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

3. The number of instances taken in this case is same as in last case which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

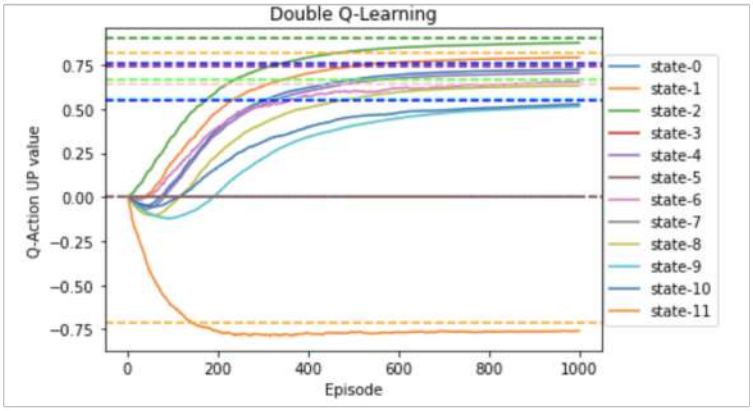

Figure 20: Double Q Action UP estimates vs episodes

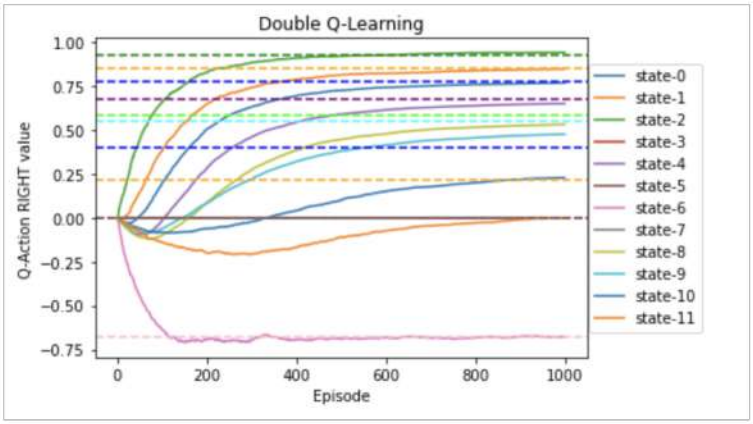

Figure 21: Double Q Action RIGHT estimates vs episodes

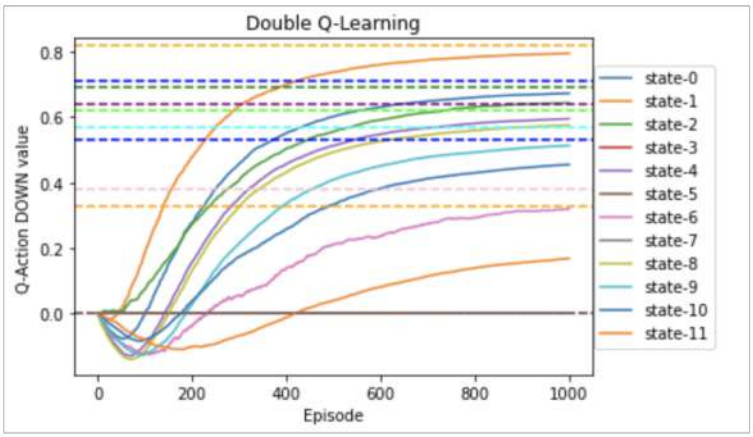

Figure 22: Double Q Action DOWN estimates vs episodes

4. The environment diagram is plotted in figure 24.

5. The Hyper parameters used in Double Q are Alpha, epsilon , gamma and no of episodes.

(i). **Alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting episodes. However the fluctuations here are almost zero.The value is taken same as above to maintain

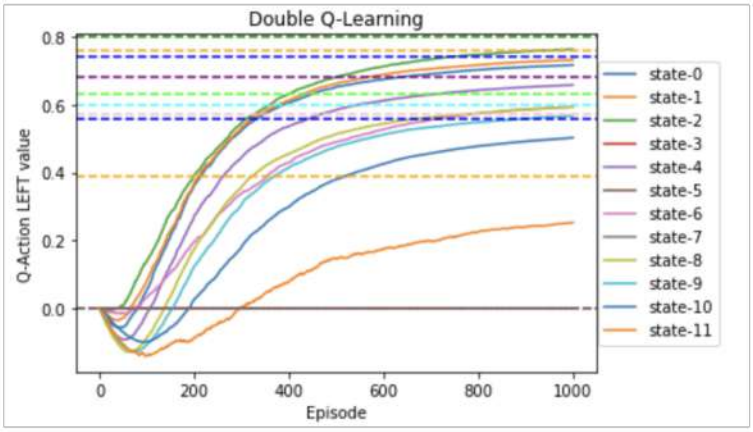

Figure 23: Double Q Action LEFT estimates vs episodes

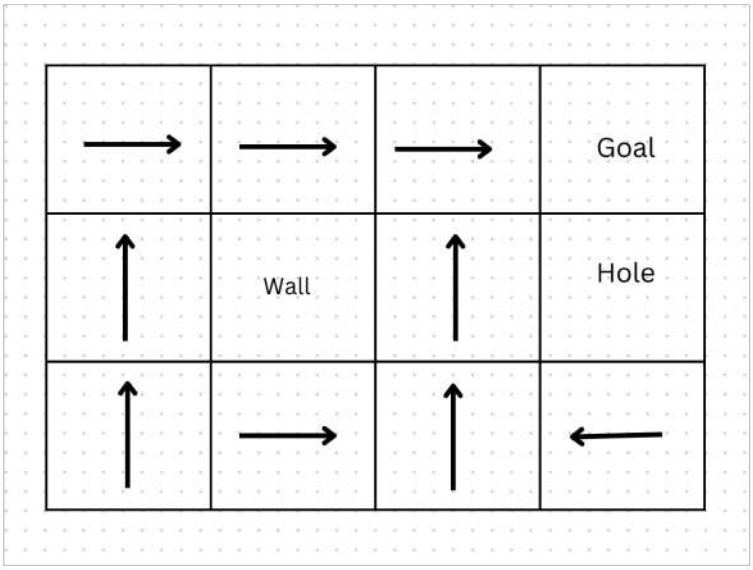

Figure 24: Optimal policy using Double Q-Learning

uniformity. Same goes for choice of linear over exponential.
(ii) **Epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.9 which decreases exponentially to 0.1. Technically , we start with exploration and go towards more greedy approach.
(iii) **Gamma and No. of episodes** - The values of both parameters are taken to be the standard values . The gamma is specified as 0.99 and no. of episodes taken here is 1000 as the values near the hole took almost these many steps to converge. Max Steps is not used as Double Q algorithm takes only one step at a time and selects the maximum value of the action.

6. We can clearly observe that the plot converges as fast as Q learning but it gets rid of the maximization bias that we was facing in Q learning. There's no overshooting of the values and it gradually converges to it's true values. This method is almost as accurate as Q learning in the choice of optimal policy and the values of Action-value functions.

Solution to Problem 5: Comparing Control Algorithms

1. The plot is given as in Figure 25 :

2. In spite of getting the same optimal policy , the policy convergence graph shows us the quickness of the algorithms in achieving the policy. The Q and Double Q algorithms are quite fast in getting to the optimal

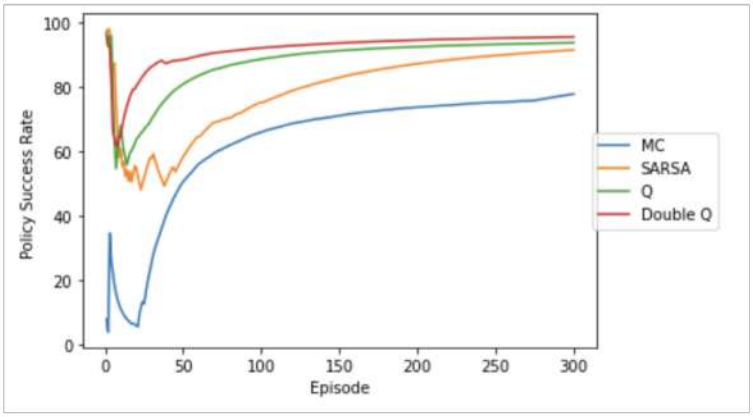

Figure 25: Policy Success rate (%) vs Episodes

policy while SARSA is slow followed by MCFCV. The no. of episodes taken in this plot is 300 because upto 500 or 1000 all the plots converges to a certain maximum value.

3. The plot is given as in Figure 26 :

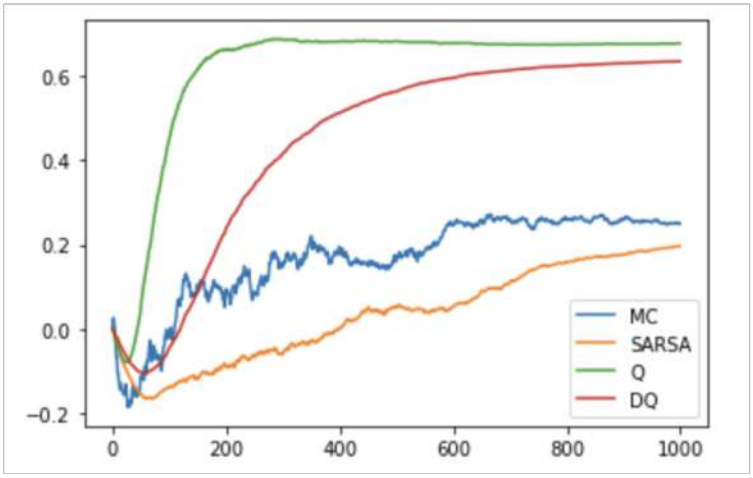

Figure 26: Estimated Expected Return (from start state) vs Episodes

4. We can see how quickly Q and Double Q learning estimates the true return of the starting value while MC and SARSA still requires more episodes to reach their value. Also we can see that SARSA is still rising while MC kinda bounce around a single value showing how fast and good SARSA is against MCFVC.

5. The plot is given as in Figure 27 :

6. We can see how quickly Q and Double Q errors got to 0 while the other algorithms still has an error after 1000 episodes. The steepness of Q algorithm is to be noticed here showing how reliable and good of an algorithm it is compared to others.

Solution to Problem 6: SARSA Lambda - Replacing Trace

1. The plot of State-value function of every state is plotted in Figure 28. This is SARSA Lambda Replacing for finding the optimal policy for RME. The signature is same as given in the lecture slides.

2. The plots of State-Action pair Values for SARSA Lambda Replacing are plotted in the Figures 8-11. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

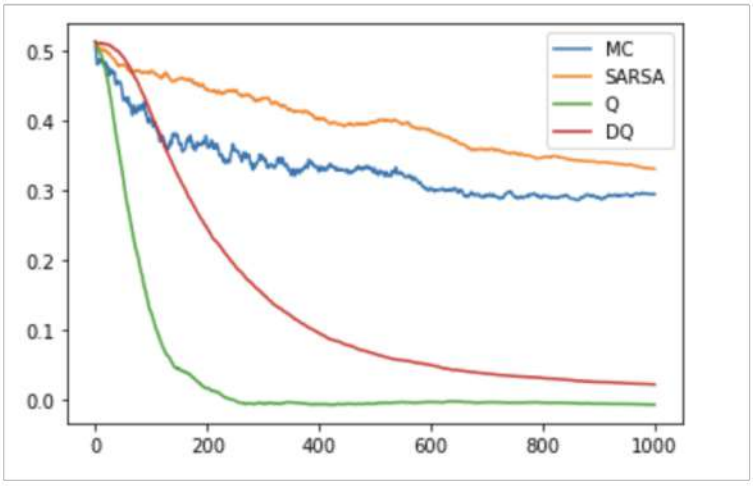

Figure 27: State Value Function Estimation Error vs Episodes

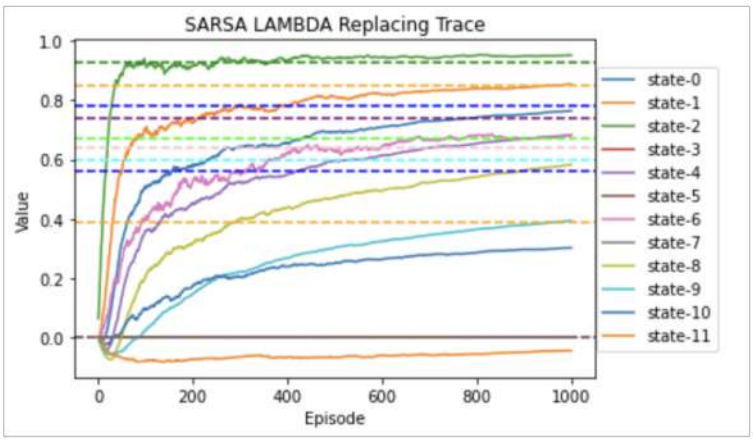

Figure 28: SARSA Lambda Replacing Value estimates vs episodes

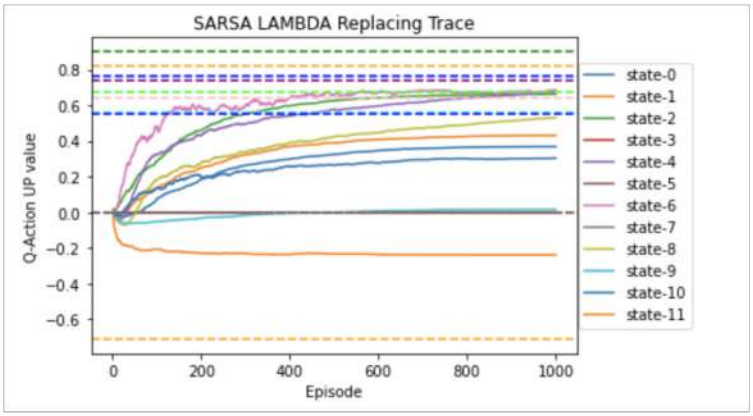

Figure 29: SARSA Lambda Replacing Action UP estimates vs episodes

3. The number of instances taken in this case is same as in above cases which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

4. The environment diagram is plotted in figure 33.

5. The Hyper parameters used in SARSA Lambda Replacing are Lambda , alpha, epsilon , gamma and no

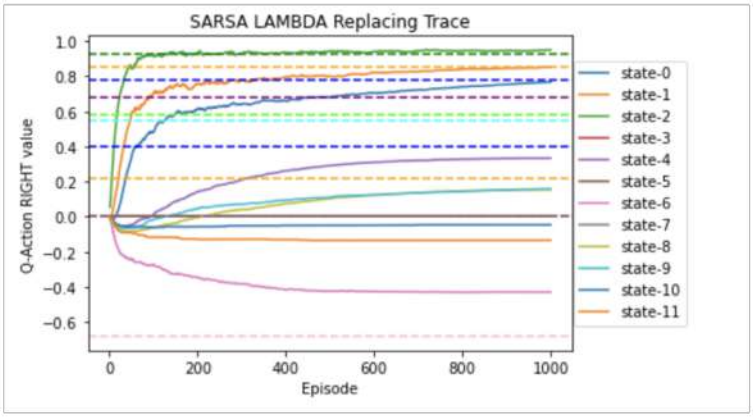

Figure 30: SARSA Lambda Replacing Action RIGHT estimates vs episodes

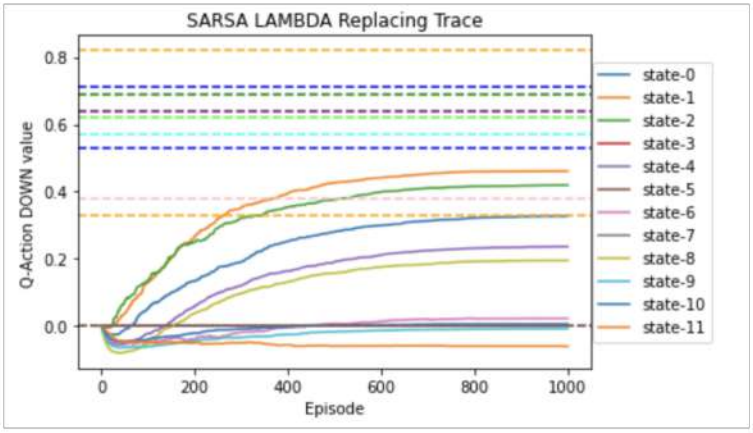

Figure 31: SARSA Lambda Replacing Action DOWN estimates vs episodes

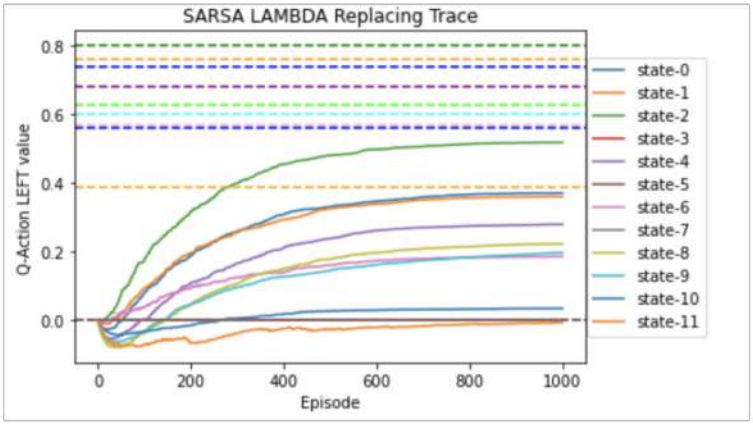

Figure 32: SARSA Lambda Replacing Action LEFT estimates vs episodes

of episodes.
(i). **Alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting episodes. .The value is taken same as above to maintain uniformity. Same goes for choice of linear over exponential.
(ii) **Epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.9 which decreases

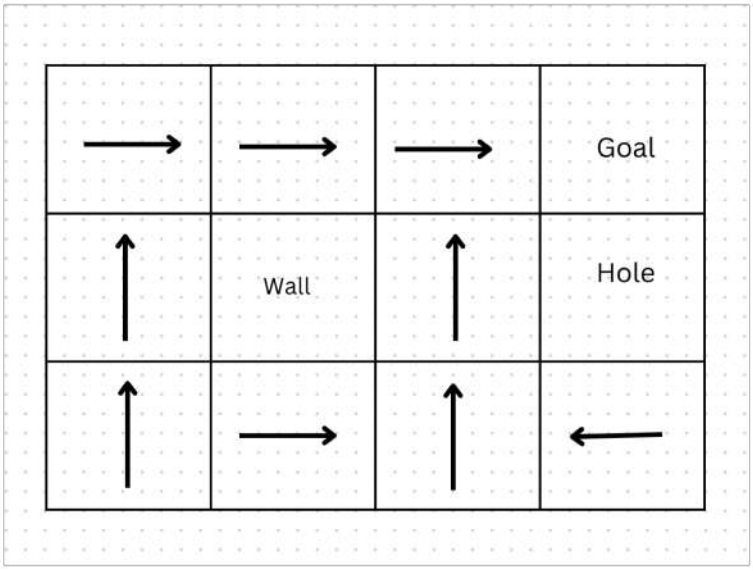

Figure 33: Optimal policy using SARSA Lambda Replacing

exponentially to 0.1. Technically , we start with exploration and go towards more greedy approach.
(iii) **Gamma and No. of episodes** - The values of both parameters are taken to be the standard values. The gamma is specified as 0.99 and no. of episodes taken here is 1000 as the values near the hole took almost these many steps to converge. Max Steps is not used as TD algorithms takes only one step at a time.
(iv) **Lambda** - The hyper-parameter Lambda is used here to specify the weights of eligibility values used in next iterations which is taken here to be 0.1. Higher values of lambda results in slower convergence and is kinda biased to previously obtained values. While lower values almost neglects the eligibility trace matrix.

6. The plot shows us how big of an improvement is this algorithm over the normal SARSA . the values converges to it's true values which the SARSA was unable to do in given 1000 episodes. Instead of bootstrapping it uses Lambda returns to give more accurate results as shown in the plot . Also the value of state 11 is quite low because of the ambiguity of its action. It has two major actions to select which is kinda similar , so that drags it's values quite down .

Solution to Problem 7: SARSA Lambda - Accumulating Trace

1. The plot of State-value function of every state is plotted in Figure 34. This is SARSA Lambda Accumulating for finding the optimal policy for RME. The signature is same as given in the lecture slides.

2. The plots of State-Action pair Values for SARSA Lambda Accumulating are plotted in the Figures 35-38. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

3. The number of instances taken in this case is same as in above cases which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

4. The environment diagram is plotted in figure 39.

5. The Hyper parameters used in SARSA Lambda Accumulating are Lambda , alpha, epsilon , gamma and no of episodes.
(i). **Alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting

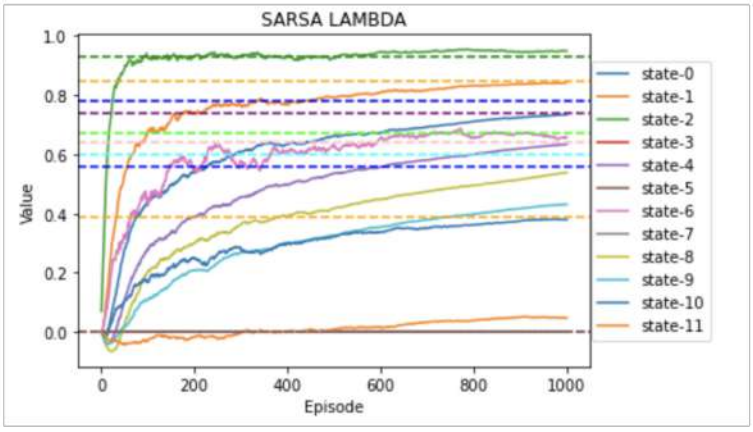

Figure 34: SARSA Lambda Accumulating Value estimates vs episodes

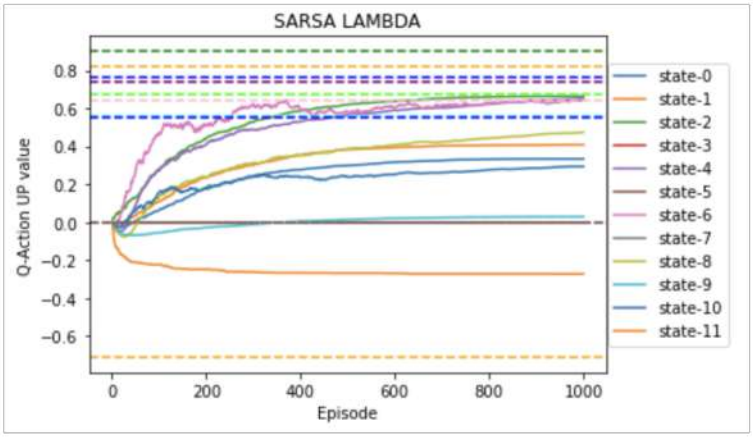

Figure 35: SARSA Lambda Accumulating Action UP estimates vs episodes

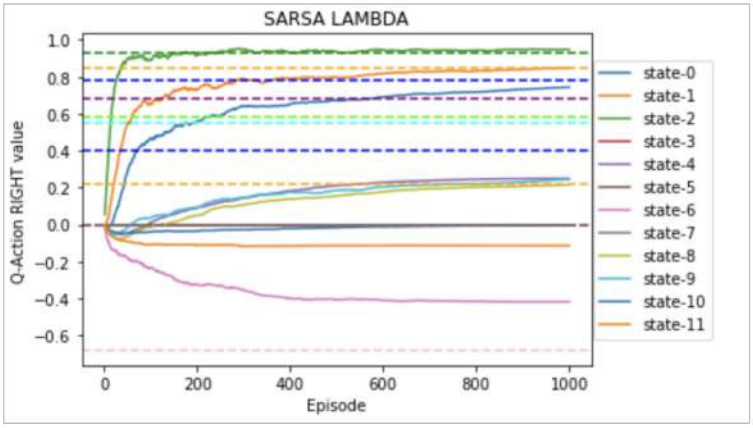

Figure 36: SARSA Lambda Accumulating Action RIGHT estimates vs episodes

episodes. .The value is taken same as above to maintain uniformity. Same goes for choice of linear over exponential.
(ii) **Epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.9 which decreases exponentially to 0.1. Technically , we start with exploration and go towards more greedy approach.
(iii) **Gamma and No. of episodes** - The values of both parameters are taken to be the standard values. The gamma is specified as 0.99 and no. of episodes taken here is 1000 as the values near the hole took almost these many steps to converge. Max Steps is not used as TD algorithms takes only one step at a

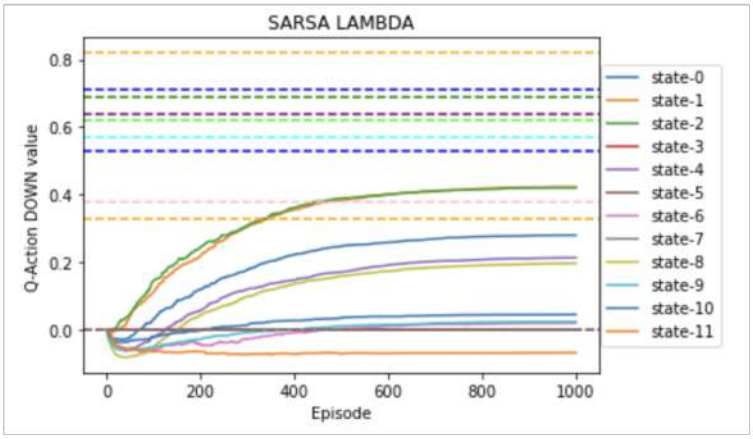

Figure 37: SARSA Lambda Accumulating Action DOWN estimates vs episodes

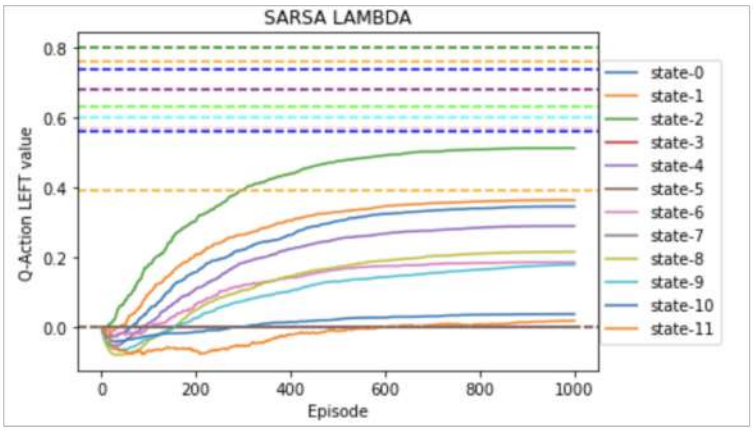

Figure 38: SARSA Lambda Accumulating Action LEFT estimates vs episodes

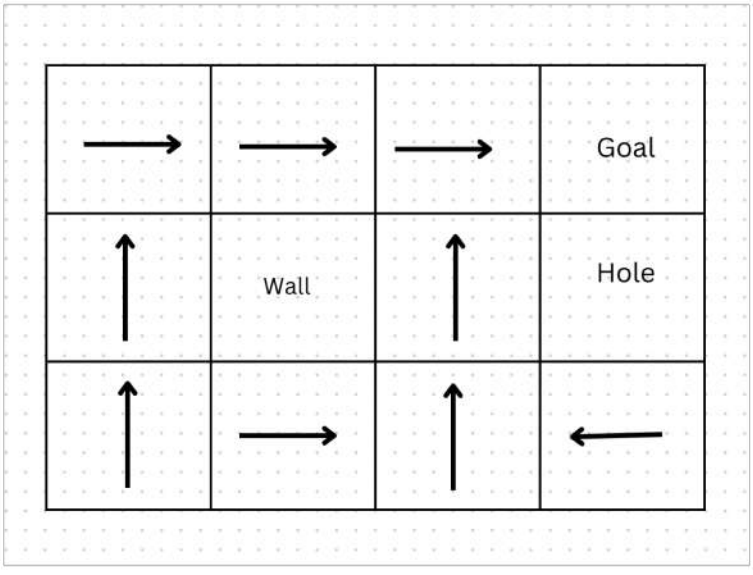

Figure 39: Optimal policy using SARSA Lambda Accumulating

time.

(iv) **Lambda** - The hyper-parameter Lambda is used here to specify the weights of eligibility values used in next iterations which is taken here to be 0.1. Higher values of lambda results in slower convergence

and is kinda biased to previously obtained values. While lower values almost neglects the eligibility trace matrix.
All the above values are kept same to maintain the uniformity of the code.

6. The plot shows us the improvement of this algorithm over the normal SARSA and SARSA Replacing. Also we can see a lag in some of the actions value functions , which is because of the epsilon value chosen. This lag is one of the reason the Q lambda is proffered over this method.

Solution to Problem 8: Q Lambda - Replacing Trace

1. The plot of State-value function of every state is plotted in Figure 34. This is Q Lambda Replacing for finding the optimal policy for RME. The signature is same as given in the lecture slides.

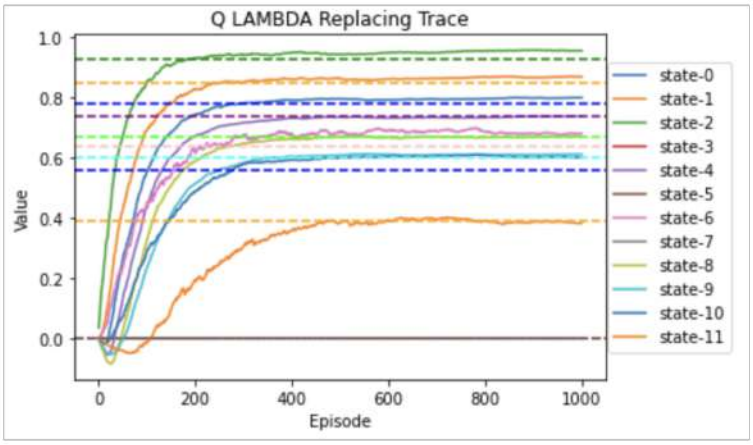

Figure 40: Q Lambda Replacing Value estimates vs episodes

2. The plots of State-Action pair Values for Q Lambda Replacing are plotted in the Figures 35-38. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

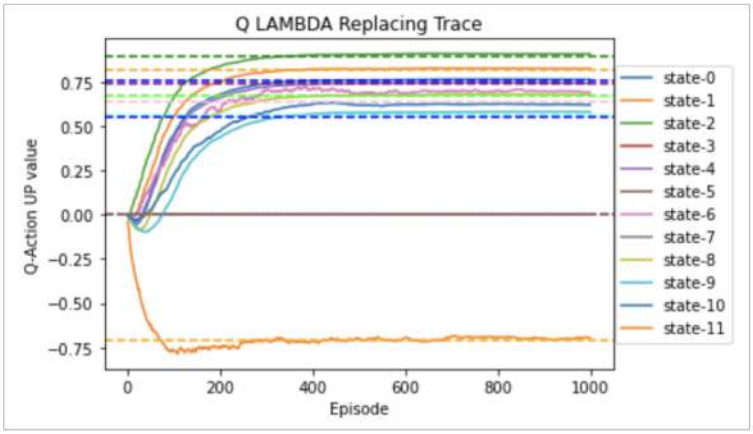

Figure 41: Q Lambda Replacing Action UP estimates vs episodes

3. The number of instances taken in this case is same as in above cases which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

4. The environment diagram is plotted in figure 39.

5. The Hyper parameters used in Q Lambda Replacing are Lambda , alpha, epsilon , gamma and no of episodes.

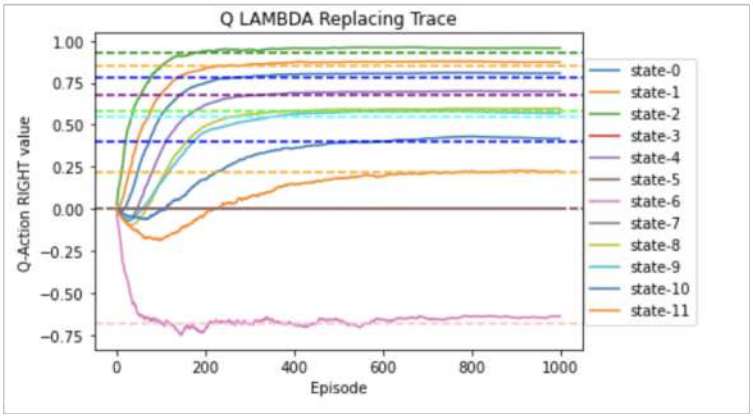

Figure 42: Q Lambda Replacing Action RIGHT estimates vs episodes

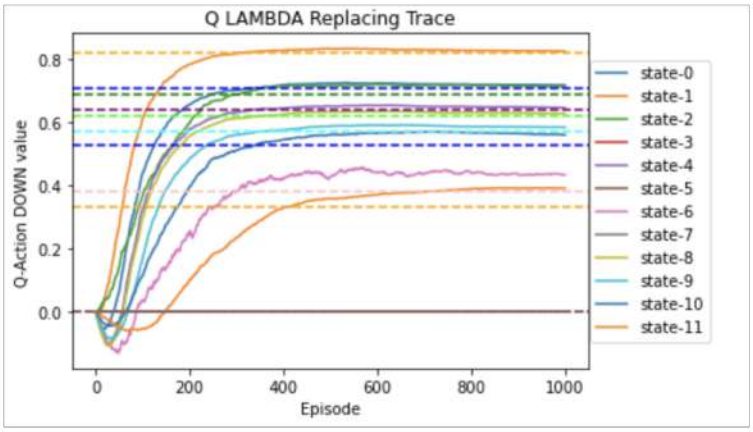

Figure 43: Q Lambda Replacing Action DOWN estimates vs episodes

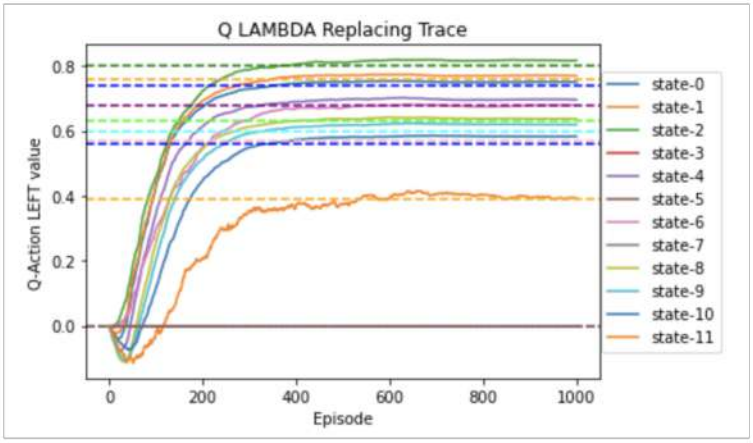

Figure 44: Q Lambda Replacing Action LEFT estimates vs episodes

(i). **Alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting episodes. .The value is taken same as above to maintain uniformity. Same goes for choice of linear over exponential.
(ii) **Epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.9 which decreases

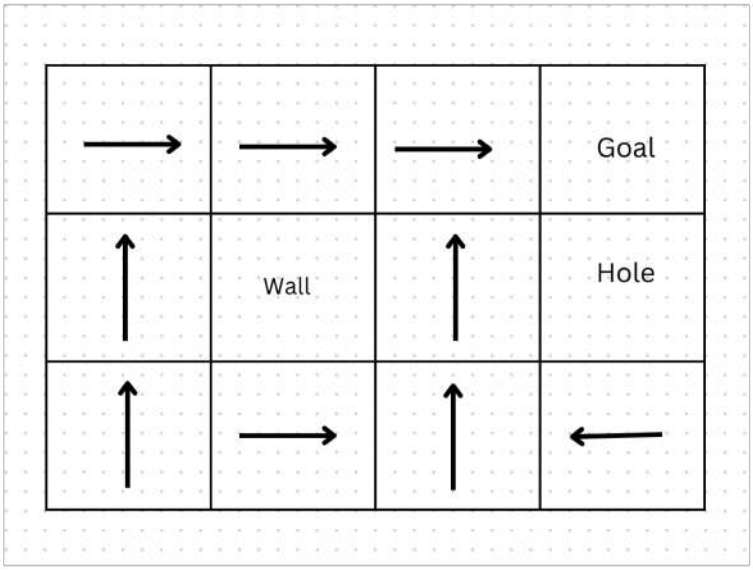

Figure 45: Optimal policy using Q Lambda Replacing

exponentially to 0.1. Technically , we start with exploration and go towards more greedy approach.
(iii) **Gamma and No. of episodes** - The values of both parameters are taken to be the standard values. The gamma is specified as 0.99 and no. of episodes taken here is 1000 as the values near the hole took almost these many steps to converge. Max Steps is not used as TD algorithms takes only one step at a time.
(iv) **Lambda** - The hyper-parameter Lambda is used here to specify the weights of eligibility values used in next iterations which is taken here to be 0.1. Higher values of lambda results in slower convergence and is kinda biased to previously obtained values. While lower values almost neglects the eligibility trace matrix.
All the above values are kept same to maintain the uniformity of the code.

6. This plot improves over the other Eligibility trace methods such as SARSA lambda. The values obtained through this method is almost perfect as shown in the plot. This is an On policy algorithm and is an extension to the Q Learning approach. Also there's a certain maximization bias associated to this method which is a result of it's roots from Q Learning.

Solution to Problem 9: Q Lambda - Accumulating Trace

1. The plot of State-value function of every state is plotted in Figure 34. This is Q Lambda Accumulating for finding the optimal policy for RME. The signature is same as given in the lecture slides.

2. The plots of State-Action pair Values for Q Lambda Accumulating are plotted in the Figures 35-38. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

3. The number of instances taken in this case is same as in above cases which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

4. The environment diagram is plotted in figure 39.

5. The Hyper parameters used in Q Lambda Accumulating are Lambda , alpha, epsilon , gamma and no of episodes.
(i). **Alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting

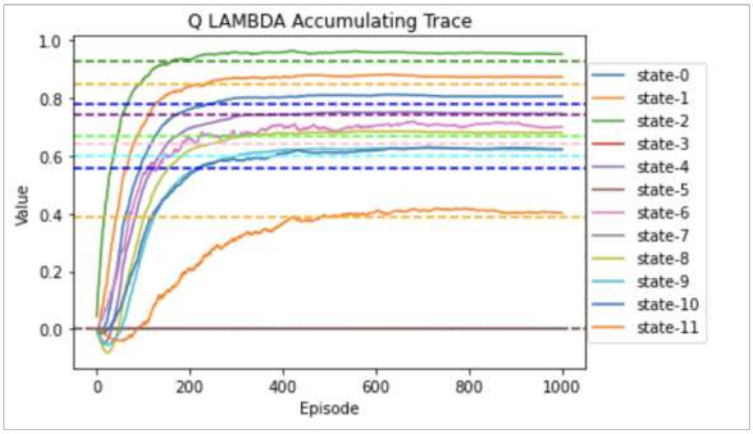

Figure 46: Q Lambda Accumulating Value estimates vs episodes

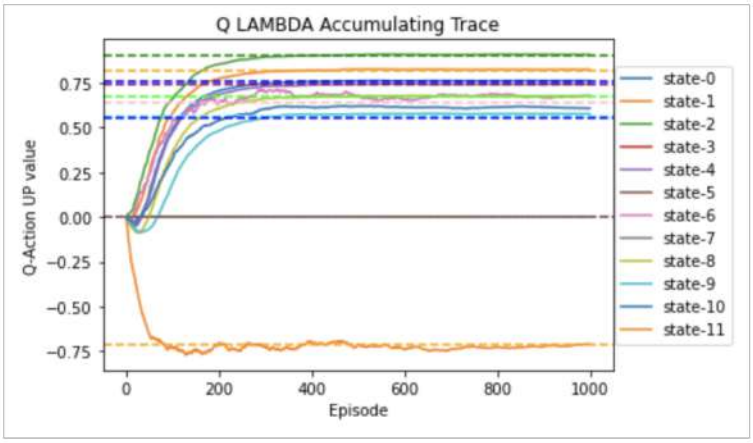

Figure 47: Q Lambda Accumulating Action UP estimates vs episodes

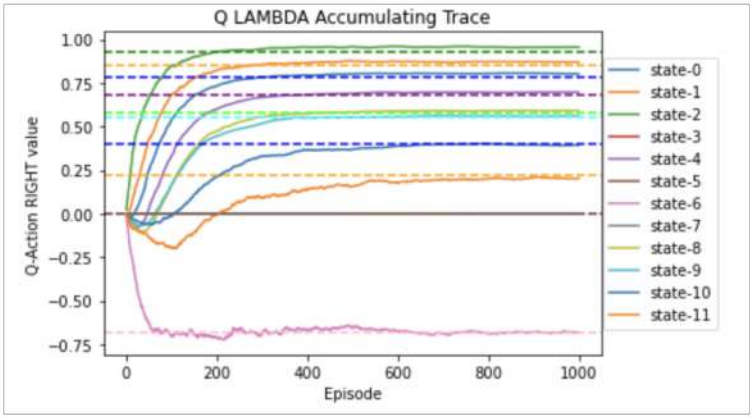

Figure 48: Q Lambda Accumulating Action RIGHT estimates vs episodes

episodes. .The value is taken same as above to maintain uniformity. Same goes for choice of linear over exponential.
(ii) **Epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.9 which decreases exponentially to 0.1. Technically , we start with exploration and go towards more greedy approach.
(iii) **Gamma and No. of episodes** - The values of both parameters are taken to be the standard values. The gamma is specified as 0.99 and no. of episodes taken here is 1000 as the values near the hole took almost these many steps to converge. Max Steps is not used as TD algorithms takes only one step at a

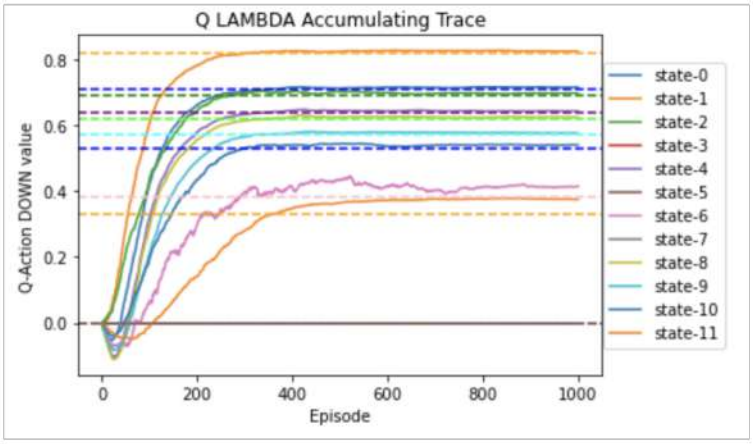

Figure 49: Q Lambda Accumulating Action DOWN estimates vs episodes

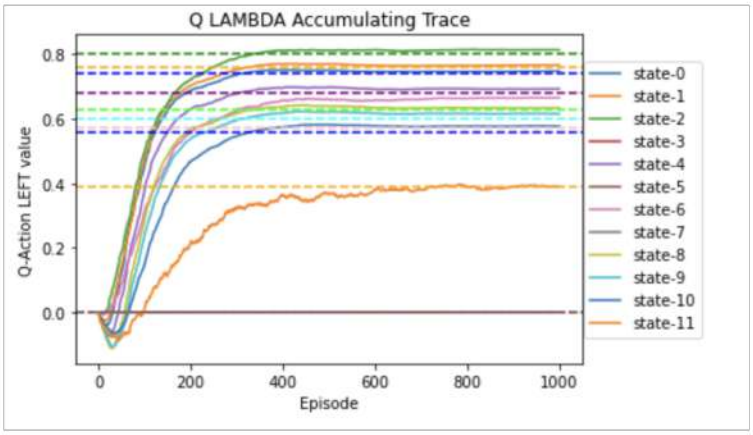

Figure 50: Q Lambda Accumulating Action LEFT estimates vs episodes

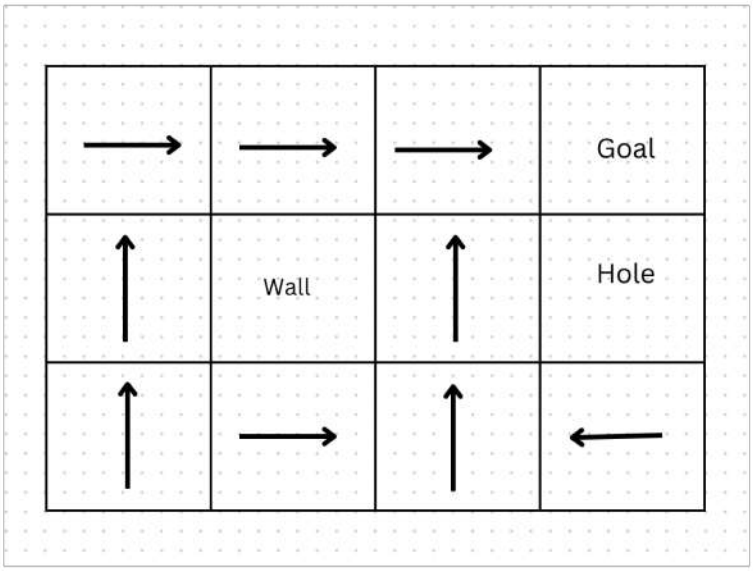

Figure 51: Optimal policy using Q Lambda Accumulating

time.
(iv) **Lambda** - The hyper-parameter Lambda is used here to specify the weights of eligibility values used in next iterations which is taken here to be 0.1. Higher values of lambda results in slower convergence

and is kinda biased to previously obtained values. While lower values almost neglects the eligibility trace matrix.
All the above values are kept same to maintain the uniformity of the code.

6. This is another version of the Q Lambda in which the Eligibility traces are not replaces thus giving some variance in the plot. The plots are also quite similar to Q lambda replacing traces. Also , Similar to that we can find the values overshooting a bit because of Q algorithm used.

Solution to Problem 10: DYNA - Q

1. The plot of State-value function of every state is plotted in Figure 34. This is DYNA Q for finding the optimal policy for RME. The signature is same as given in the lecture slides.

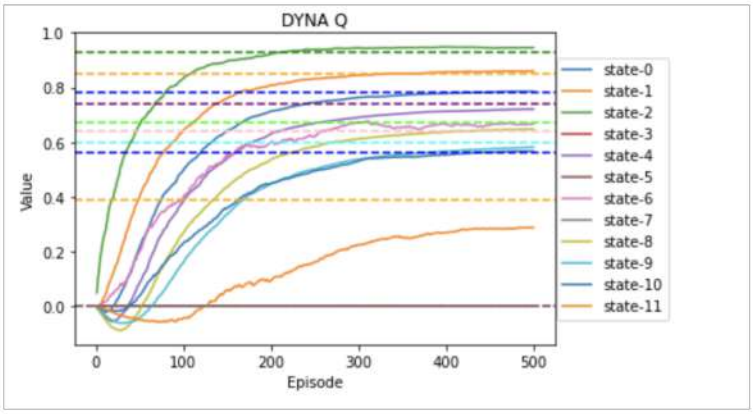

Figure 52: DYNA Q Value estimates vs episodes

2. The plots of State-Action pair Values for DYNA Q are plotted in the Figures 35-38. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

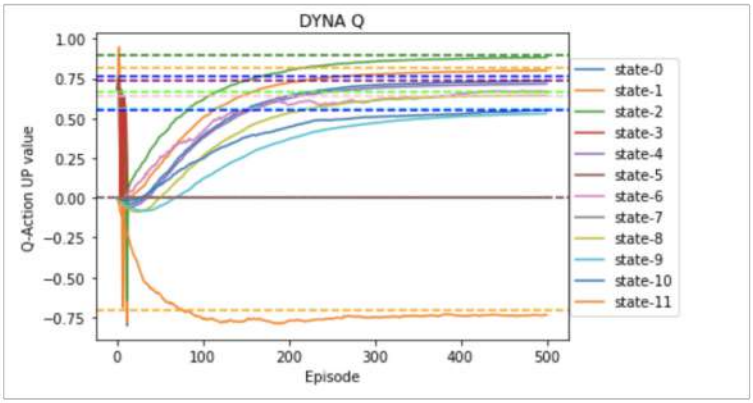

Figure 53: DYNA Q Action UP estimates vs episodes

3. The number of instances taken in this case is same as in above cases which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

4. The environment diagram is plotted in figure 39.

5. The Hyper parameters used in DYNA Q are No. of planning steps , alpha, epsilon , gamma and no of episodes.
(i). **Alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which

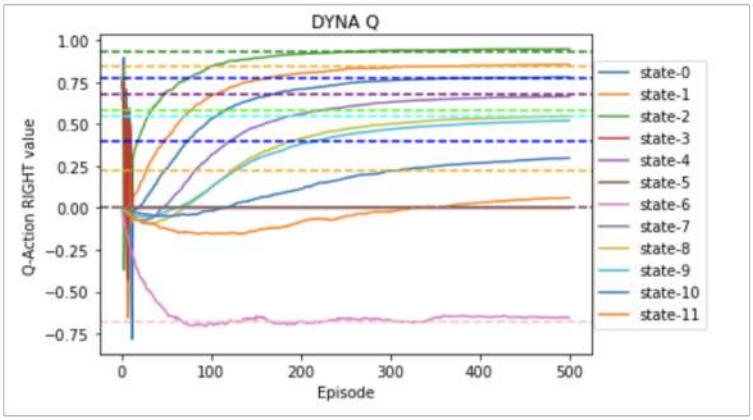

Figure 54: DYNA Q Action RIGHT estimates vs episodes

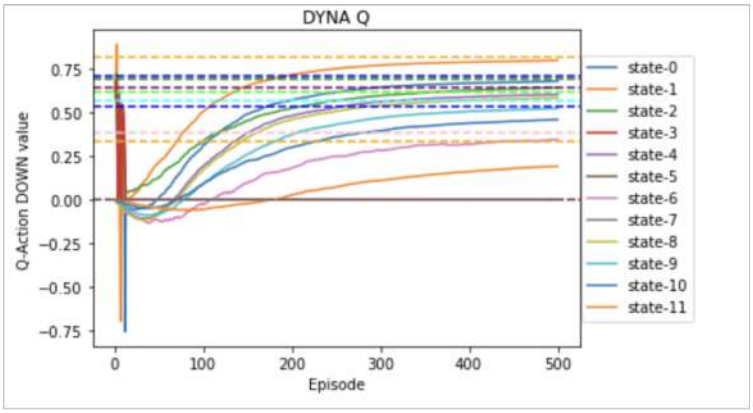

Figure 55: DYNA Q Action DOWN estimates vs episodes

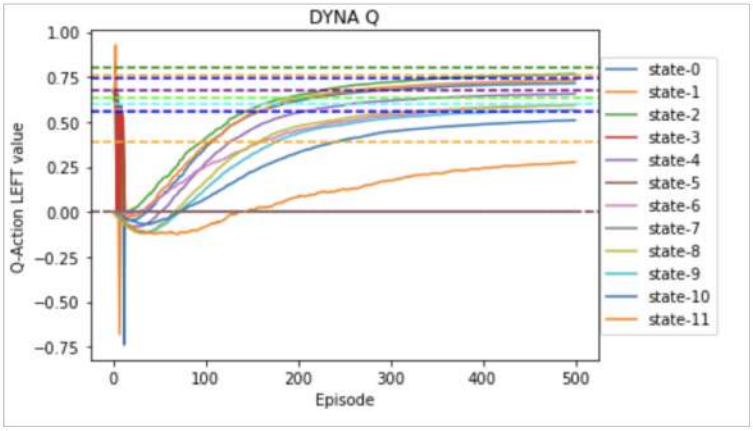

Figure 56: DYNA Q Action LEFT estimates vs episodes

exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting episodes. .The value is taken same as above to maintain uniformity. Same goes for choice of linear over exponential.

(ii) **Epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.9 which decreases exponentially to 0.1. Technically , we start with exploration and go towards more greedy approach.

(iii) **Gamma and No. of episodes** - The values of both parameters are taken to be the standard values.

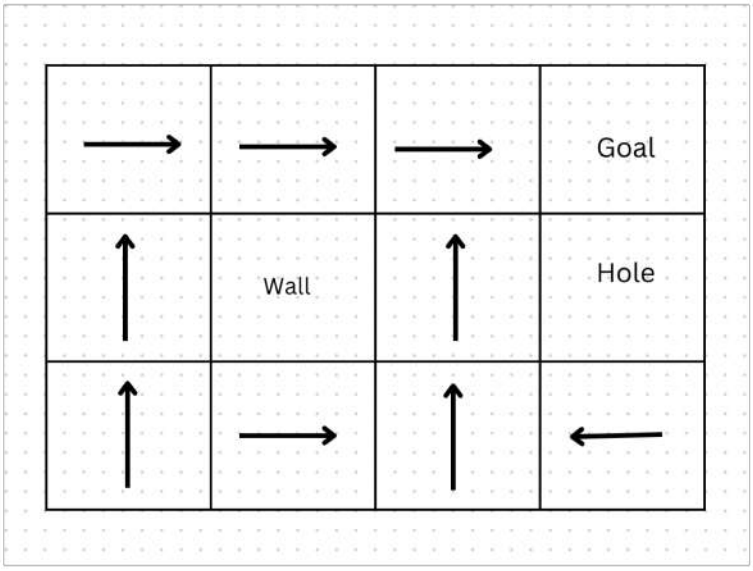

Figure 57: Optimal policy using DYNA Q

The gamma is specified as 0.99 and no. of episodes taken here is 500 as the values near the hole took almost these many steps to converge and also this is an time consuming algorithm. Max Steps is not used as TD algorithms takes only one step at a time.
(iv) **No. of planning steps** - This is required to calculate the model parameters on every iteration. More is the value , more expensive is the operations. In this case , No. of steps is taken as 10.
All the above values are kept same to maintain the uniformity of the code.

6. The number of episodes taken here is only 500 because is it a Model based technique which is quite computationally expensive as few iterations requires time to compute the transition probabilities and reward function. The algorithm is based on Q learning which is the reason of it's similarities to the plots.

Solution to Problem 11: Trajectory Sampling

1. The plot of State-value function of every state is plotted in Figure 34. This is Trajectory Sampling for finding the optimal policy for RME. The signature is same as given in the lecture slides.

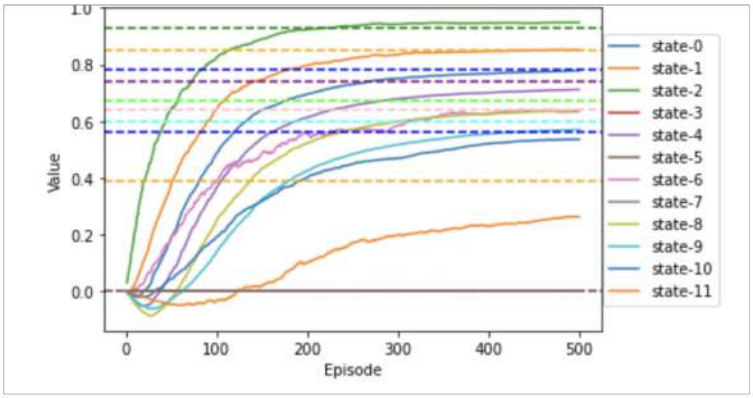

Figure 58: Trajectory Sampling Value estimates vs episodes

2. The plots of State-Action pair Values for Trajectory Sampling are plotted in the Figures 35-38. The plots are of all Four actions vs episodes. (Plotting them on single plot would make the graph congested , so the plots are made on 4 different graphs)

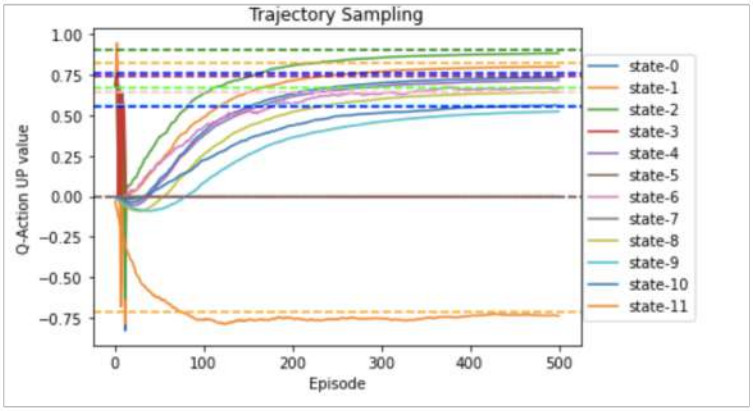

Figure 59: Trajectory Sampling Action UP estimates vs episodes

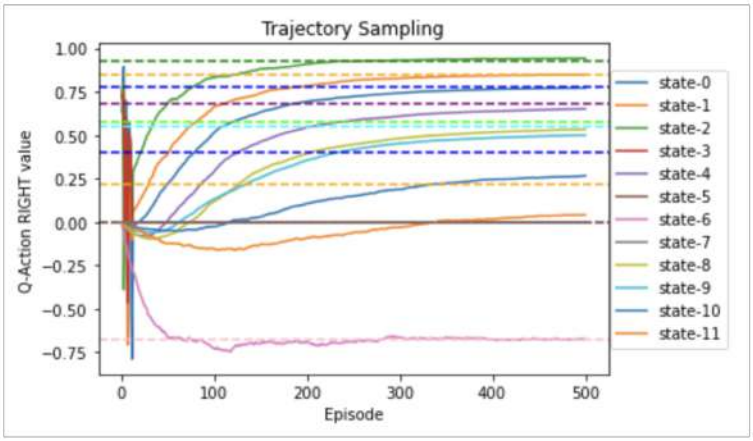

Figure 60: Trajectory Sampling Action RIGHT estimates vs episodes

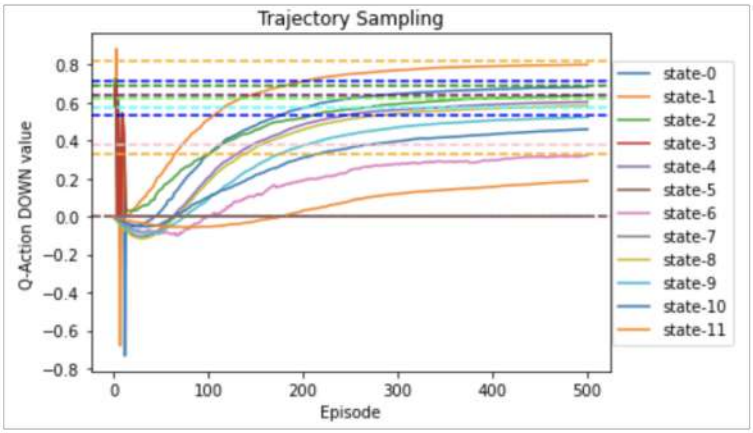

Figure 61: Trajectory Sampling Action DOWN estimates vs episodes

3. The number of instances taken in this case is same as in above cases which is 20. The resultant values are the averaged out values over these instances with total 1000 episodes on x axis. The seeds are numbered 1-20. This averaging out significantly reduced the fluctuations in the values.

4. The environment diagram is plotted in figure 63.

5. The Hyper parameters used in Trajectory Sampling are No. of planning steps , alpha, epsilon , gamma and no of episodes.
   (i). **Alpha** - The alpha value taken in the current working code is same as last question i.e 0.1 which

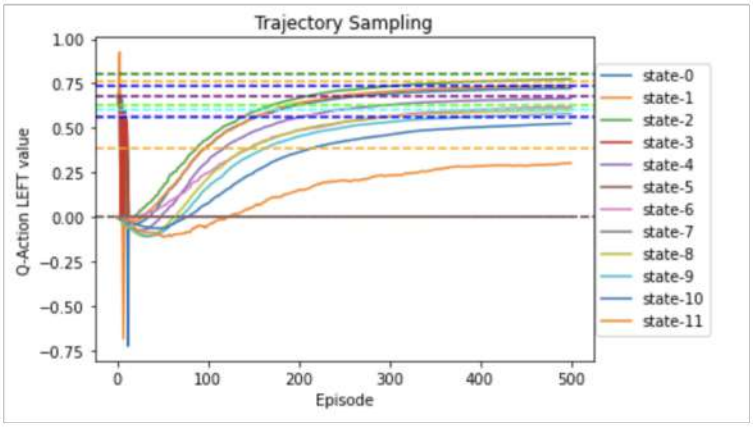

Figure 62: Trajectory Sampling Action LEFT estimates vs episodes

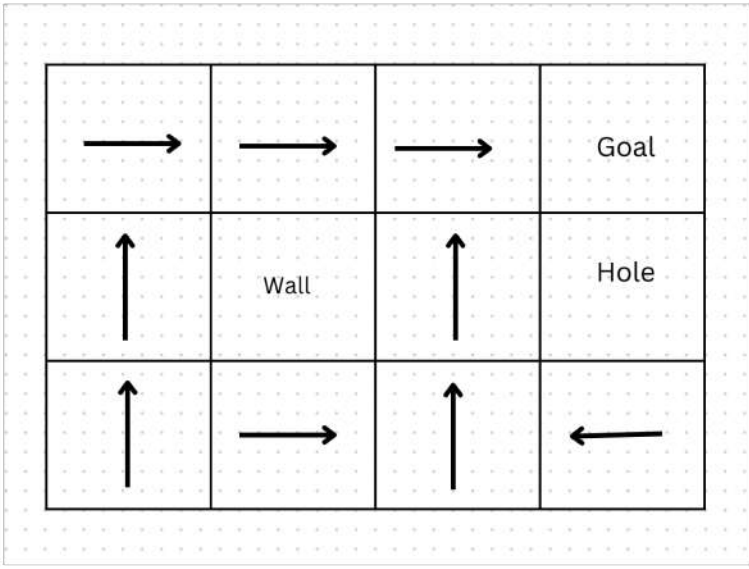

Figure 63: Optimal policy using Trajectory Sampling

exponentially decays to 0.01. Here we had two things to select, first the type of decay and second the value of decay. The main reason for selecting both parameters for alpha is quite the same i.e dealing with high variance and fluctuations with the state values. Lower value of alpha gives less variance in the starting episodes. .The value is taken same as above to maintain uniformity. Same goes for choice of linear over exponential.
(ii) **Epsilon** - The epsilon value is kind of an adjuster between exploration and exploitation which helps us in choosing the next action in the environment. The epsilon value taken here is 0.9 which decreases exponentially to 0.1. Technically , we start with exploration and go towards more greedy approach.
(iii) **Gamma and No. of episodes** - The values of both parameters are taken to be the standard values. The gamma is specified as 0.99 and no. of episodes taken here is 500 as the values near the hole took almost these many steps to converge and also this is an time consuming algorithm. Max Steps is not used as TD algorithms takes only one step at a time.
(iv) **No. of planning steps** - This is required to calculate the model parameters on every iteration. More is the value , more expensive is the operations. In this case , No. of steps is taken as 10.
All the above values are kept same to maintain the uniformity of the code.

6. This algorithm is also similar to DYNA Q as it is also a model based method. The convergence of values is same as in DYNA Q as both algorithm only differs in the calculations of model parameters.

Solution to Problem 12: Comparing Control Algorithms

1. The plot is given as in Figure 64 :

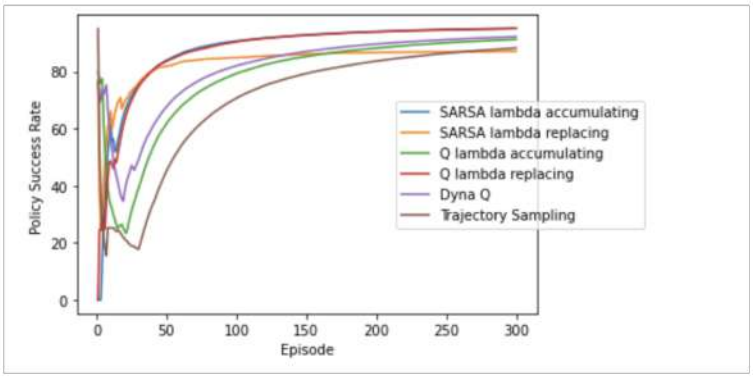

Figure 64: Policy Success rate (%) vs Episodes

2. We can see in the plot the Q based algorithms are the ones which have high success rate and faster convergence. SARSA gives a different policy initially which makes it slow compared to others.

3. The plot is given as in Figure 65 :

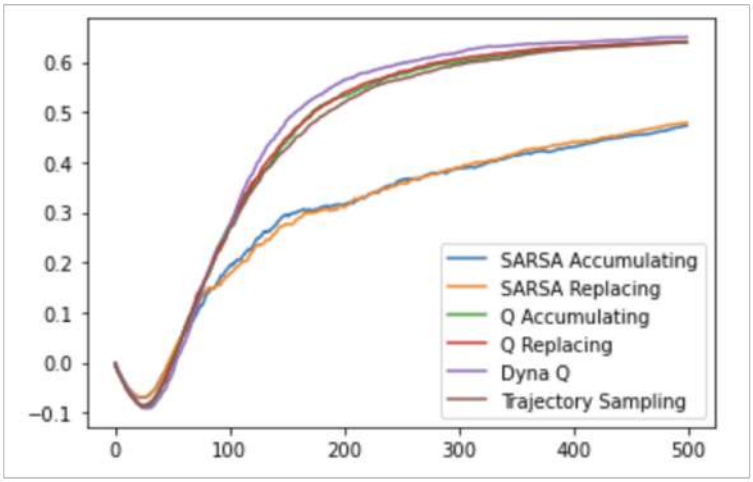

Figure 65: Estimated Expected Return (from start state) vs Episodes

4. We can see how quickly Q learning estimates the true return of the starting value while SARSA still requires more episodes to reach their value. We can say now for sure that Q based methods are the best in value based methods. The plot shows how quickly Q based methods reaches the valuea nad how SARSA lags behind.

5. The plot is given as in Figure 66 :

6. As expected Q based algorithms gives almost 0 error after only 300 episodes while SARSA is still improving. We can see 4 algorithms performing almost same which is because they are using same Q learning approach to calculate the value of the state.

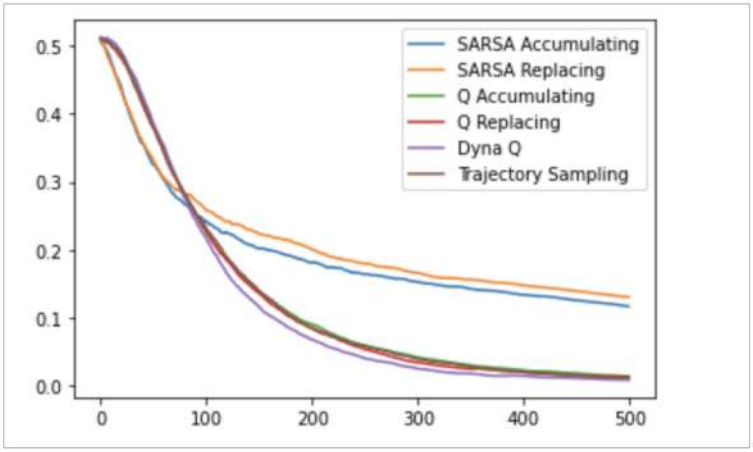

Figure 66: State Value Function Estimation Error vs Episodes