Assignment 4: CS7641- Machine Learning Yogesh Edekar(GTUID – yedekar3)

April 23, 2021

1. Introduction

As part of assignment 4 I am trying to analyze the applications of reinforcement learning algorithms to solve the Markov Decision Problems (MDP). I am using Value Iteration, Policy Iteration as well as Q Learning in order to solve the two MDPs. The first MDP is a smaller MDP with few hundred states where as the second MDP is a larger MDP with larger steps (i.e. more than thousand steps). Details of those experiments and observations obtained are provided in the following sections.

2 Implementation

The implementation of MDP is done using the mdptoolbox package available in python. ValueIteration and PolicyIteration classes of mdptoolbox were used to find out the optimal strategy for solving the problems. The problems were analyzed based on varying the states and finding out the corresponding rewards for the problem as well as determining the convergence of the problems.

3 Problems

3.1 Forest Management Problem

3.1.1 Introduction

The first MDP problem chosen is a simple Forest management problem chosen from the mdptoolbox example class. This problem deals with two objectives; the first objective is to maintain the forest for preserving wildlife in the forest and the second objective is to cut the trees in order to sell the wood to earn money. There are two possible actions that can be taken **wait** designated by **0** and **cut** designated by **1**. Each year there is a probability of wildfire that burns the forest and hence we get the new generation of the trees corresponding to each new state in the problem.

3.1.2 Dimensions

Forest is a non grid problem where the states are specified by the generation of the trees available in the forest. In order to assess the impact of states on the overall problem I have chosen **smaller number of states** to be **10** indicating 10 generations of trees and **larger number of states** to be **1000** indicating 1000 generations of trees in the forest. Moreover to make the problem interesting I have tested the problem against different values of gamma (discount factor) to observe how it impacts the problem.

3.1.3 Implementation

The forest management problem is represented by the class forest from the mdp.example package. We do not change the default values of the class. We are performing the experiments on forest management problem with **1000 states**, the **reward** when forest is in it's oldest state and action **wait** is performed is **4**, The **reward** for performing action **cut** is **2** and the **probability of wildfire** such that the forest is destroyed and a newest generation of trees is generated is **0.1**

3.2 Frozen lake problem

3.2.1 Introduction

The second MDP problem chosen is the frozen lake problem which is formulated using the environments provided by the gym library. This is a grid problem in which a frozen lake is represented by a grid of specific size. There are certain tiles in the grid that are frozen with ice and hence are safe to step on. There are certain other tiles in the grid which are water holes and hence can not be stepped on. The goal of the agent in frozen lake environment is to start at the starting position and reach till the ending position safely. An episode ends when the agent reaches the goal or falls in the hole. Upon reaching safely towards the goal a reward of 1 is received.

3.2.2 Dimensions

Frozen lake is a grid problem specified by the grid of a specific length and width causing those many states in the environment. Here in order to identify the effects of the size of the grid on rewards we choose two sizes for the frozen lake. For the smaller lake is chosen to be a **4 * 4 grid** having **16 states** and for the larger size the lake is chosen to be a larger **20 * 20 grid** having **400 states**.

3.2.3 Implementation

The Frozen Lake problem is represented using an environment provided by the gym package of python. Moreover the **OpenAI_MDPToolbox** class from **hiiveMDP** package is used to convert the gym environment into the state transition matrix and reward matrix to be used for further algorithm analysis.

4 Algorithms

4.1 Value Iteration

Value iteration begins with selecting random values for the utility of a state. Utility of a state can be defined as the goodness of a state for an agent to be in so that in order to get maximum rewards we will always try to maximize the utility of a state. For each iteration we add the reward obtained by agent from being in the given state and update the utility of a state using the discount factor and the reward obtained by being in the neighboring states. Value iteration converges when the stopping thresholds of maximum iterations or the delta which indicates a limit for change in value function are reached.

4.2 Policy Iteration

Policy iteration starts with a random policy, then the value of that policy is found out and policy is changed such that the value of the policy is improved until threshold is reached. Since ultimate goal of

the agent in MDP is to find the optimal policy instead of finding optimal value policy iteration tries to find out optimal policy that gives maximum reward. Just like value iteration policy iteration is also expected to converge. Both Value and Policy iteration are used in offline planning in the sense that agent knows the transition matrix and reward matrix so can plan beforehand about the actions to be taken from this knowledge.

4.3 Q Learning (learning algorithm for reinforcement learning)

In case of Q learning the agent does not know the state transition matrix or the reward matrix. i.e. the agent does not know effect of it's actions on the environment. In this case the agent learns about the environment based on environment's response to it's action. The agent maintains a table called q table to store the rewards obtained to be in a state. This table is initialized with random values and as agent interacts with the environment this table is updated with real values when the reward obtained for a particular state is optimum as compared to the reward for the state given by Q table.

5 Problem Analysis

5.1 Value Iteration

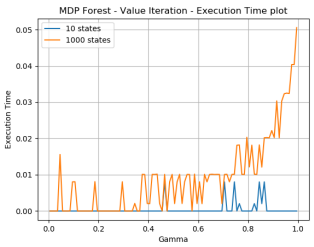
I have used the ValueIteration class from the mdp package to perform value iteration. Two different types of experiments were performed to analyze value iteration for the given MDP problems. In the first experiment value iteration is run with varying discount factors to observe the impact of changing discount factors on rewards and convergence iterations. In second experiment the value of delta is changed to observe the impact on convergence.

5.1.1 Forest Management problem

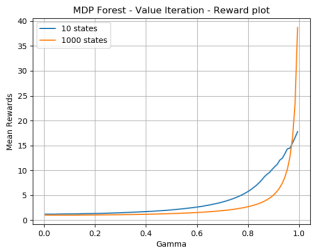
In order to find out the effect of changing states on the problem I tried to vary the states and then run the value iteration on Forest Management problem to observe the number of iterations taken for convergence as well as the reward obtained. The results obtained in this case are given as follows

States	Convergence	rewards	Time taken
20	39 iterations	8.33	0.0009
40	39 iterations	6.64	0.0009
60	39 iterations	6.07	0.0010
80	39 iterations	5.79	0.0009
100	39 iterations	5.62	0.0009

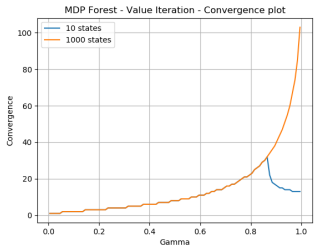
Based on these results we can see that the iterations to converge and time taken in spite of varying number of states is the same with minimal changes in the reward. I am testing the Value iteration algorithm for Forest Management MDP on two states smaller states of 10 and larger states of 1000 to observe the impact of states on overall problem. The value of gamma (discount factor) is varied from smaller to larger to observe the impact of combination of states and discount factor on convergence as well as rewards. The plot of gamma vs. rewards and gamma vs. convergence iterations for forest management problem can be given as follows



Gamma Vs. Clock Time



Gamma Vs. Reward

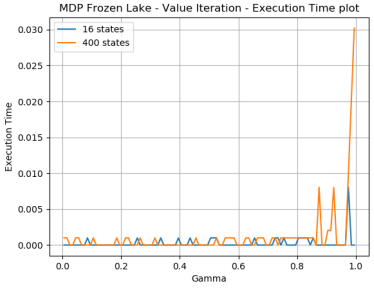


Gamma Vs. Convergence

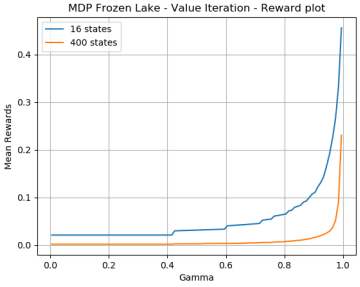
The graph above clearly depicts the impact of number of states on clock time of the algorithm. As expected the smaller number of states of the problem result is the value iteration being run faster on the problem as seen from the first graph hence the clock time of running the algorithm is much faster for 10 states(Blue line) rather than 1000 states (orange line). In general we use gamma the discount factor to solve the MDP by making it finite. Also discount factor helps us make the agent focus on long term or short term goals. A lower discount factor corresponds to more importance to short term rewards rather than long term rewards. This can be seen from the rewards and the convergence plots. Since long term rewards are important for larger discount factors closer to one we can see the rewards accumulating for longer time and hence the cumulative reward increasing in value. Also as the long term rewards are stressed more over short term rewards we see the iterations required to converge spike as we move discount factor closer towards 1.

5.1.2 Frozen Lake problem

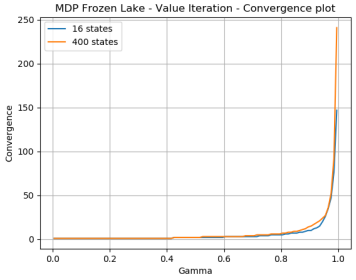
The plots of gamma vs clock time, gamma vs rewards and gamma vs convergence for the frozen lake problem of small size (16 states) and larger size(400 states) are given as follows.



Gamma Vs. Clock Time



Gamma Vs. Rewards



Gamma Vs convergence

As expected the smaller number of states (blue line) take less time to converge than the larger number of states (orange line). Looking at rewards we can see that smaller states are always yielding larger rewards as compared to larger states. This apparently can be attributed to the fact that the complexity of the environment affects the final outcome of the problem. With a larger environment there are more holes and it is difficult to navigate from start to the end. This contributes more towards the larger rewards for smaller states as compared to larger states. In terms of convergence iterations we can see that for smaller as well as larger size of the problem the converge iterations are the same. However upon looking at the combination of reward and convergence plot gives us an interesting observation that for larger problem the

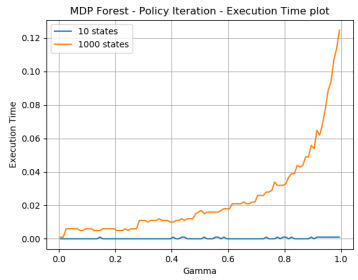
convergence happens in negative scenarios for lower values of gamma i.e. till 0.8 and then we start getting the successful outcome.

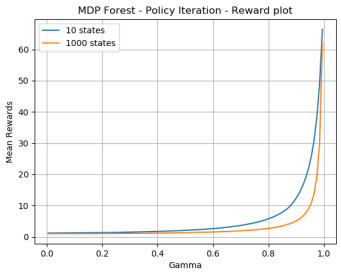
5.2 Policy Iteration

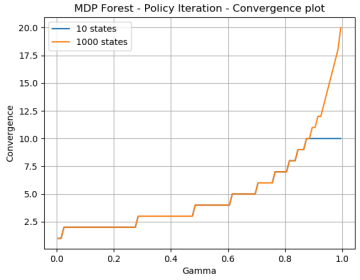
Next we run the policy iteration algorithm for solving both the problems and analyze the results. While running the policy iteration algorithm as well we take the smaller states as 10 and larger states as 1000 for the forest management problem and smaller states as 16 and larger states as 400 for the frozen lake problem.

5.2.1 Forest Management problem

The Reward and convergence plots for policy iteration algorithm for forest management problem can be given as follows



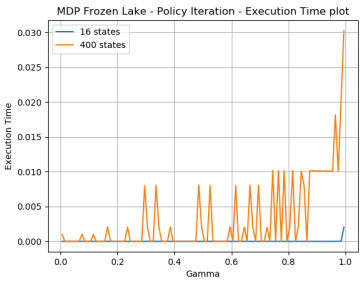


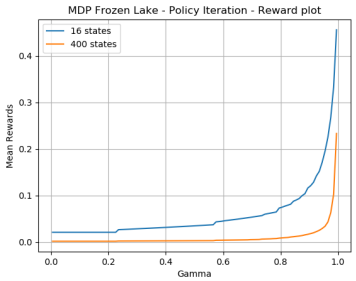


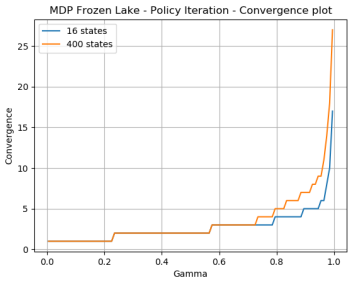
As expected from the time plot we can see that the policy iteration takes more time to converge for 1000 states problem than the 10 states problem indicating that the clock time increases linearly with the problem size. Looking at the reward plot we can see that the rewards for lower number of states are slightly higher as compared to the rewards for higher number of states. This indicates that the optimal policy for lower states includes more of wait action where as the optimal policy for larger states includes cut action. Also we see that convergence iterations are almost the same until gamma crosses 0.8 indicating that for policy iteration the discount factor does not seem to impact the convergence iterations as much for the forest management problem.

5.2.2 Frozen Lake Problem

The reward and convergence plots for policy iteration for frozen lake problem can be given as follows.







The clock time plot has been consistent through out all the experiments depicting a linear relationship between the problem size and the time taken for convergence. Interestingly the reward plot for frozen lake is shows higher rewards for smaller state (16 states) as compared to larger states (900 states). This is exactly in consistence with the observations I had for Value iteration. Also the convergence graph

does not show much of a difference until a high value of gamma is reached. For the frozen lake problem we see almost identical results running value iteration and policy iteration algorithms.

6 Policy iteration Vs. Value iteration

Upon comparing the results for both the problems from policy iteration and value iteration we can see that from iteration standpoint Policy iteration algorithm converges in less iterations as compared to the value iteration algorithm. This can be attributed to the fact that both algorithms work differently. Policy iteration chooses a random policy and makes changes to the policy so that the utility value of the optimum policy for that iteration is better than the previous value until the epsilon threshold is reached. Whereas value iteration simply tries to find the maximum utility value in each iteration. Then it finds the policy mapping to that value. In case of policy iteration the calculations per iteration are complex and more as compared to value iteration. This clearly gives an indication of why policy iteration works in less iterations than value iteration as more work is done per iteration for policy iteration. However this results in policy iteration taking more wall clock time as compared to value iteration.

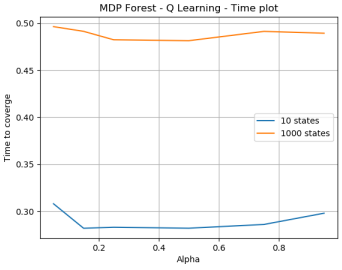
7 Q Learning

In order to analyze the impact of Q learning on both the problems I have ran the Q learning experiment for small states and larger states on both the problems. Furthermore in order to analyze the impact of different Q learning parameters I have analyzed the results on both the problem by changing the learning rate (alpha). The expectation here is as we vary the learning rate from low to high the agent learns about the environment and starts to optimize the rewards. However after reaching a certain learning rate agent starts to converge faster before getting an optimal policy. We will try to validate this behavior from the readings and plots.

7.1 Forest Management problem

In order to identify the effect of Q learning on the forest management algorithm I have changed the learning rate(alpha) of the Q learning algorithm so that alpha is gradually increased form low to high. With increasing learning rate I expect the reward to grow slightly as the environment is learned and the new rewards are weighed more over the older rewards. However once agent learns the environment sufficiently enough it starts to converge quickly thus lowering the reward after certain time. The reward and time plots for the forest management algorithm can be given as follows.

Q learning reward plot



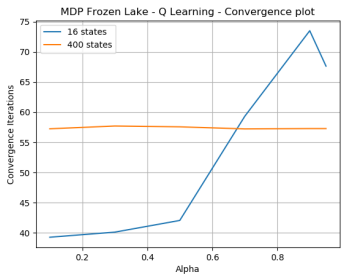
Q learning time plot

As expected and observed in all the previous algorithms the problem with less states 10 takes very less time to converge as compared to the larger problem with 1000 states. Also we observe that the reward obtained for the smaller problem is more than the larger problem for moderate values of alpha and

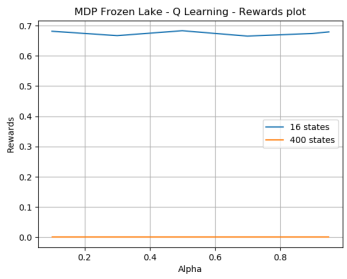
once the agent learns the policy completely then the reward starts to go down due to the fact that convergence is obtained very early for very high learning rates. Thus we can say learning rate (alpha) when used appropriately in correct moderation really helps the agent learn the environment well and take the optimal decision to find optimal policy in Q learning.

7.1.1 Frozen Lake Problem

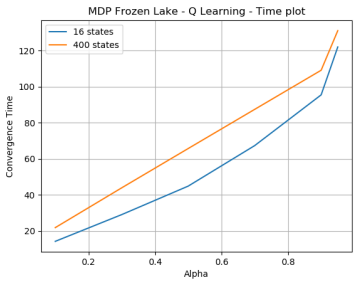
In order to identify the impact of Q learning on Frozen Lake problem, I have applied the similar technique as the Forest Management problem where alpha is increased gradually from lower values to higher values. With the increasing learning rate I expect the reward to increase gradually as effect of more learning. The reward and time plots for smaller and larger environments for frozen lake problem are given as follows



Convergence Plot



Rewards Plot



Time Plot

From the above plot it is clear that we are getting a high conversion iterations as the agent learns the environment with higher alpha. Also since we are taking more iterations to converge the clock time required to converge is also more. We don't see a particular pattern in terms of rewards for the smaller problem , however I am not able to plot the rewards graph for larger problem as I am getting all 0 values for rewards. I was not able to change the reward structure of the environment to reward at every step which would have given me the reward value for the larger environment.

8 Comparison of Policy Iteration, Value Iteration and Q learning

8.1 Forest management problem

8.1.1 Smaller states (10)

The table of iterations to converge, reward and clock time for policy iteration, value iteration and Q Learning for gamma = 0.95 can be shown as follows

Label	Policy iteration	Value Iteration	Q Learning
Iterations to Converge	10	12	
Reward	65	18	32
Clock time	0.00001	0.00001	0.3

Thus we can see for small number of states for the forest management problem Policy iteration takes smaller number of iterations to converge with maximum reward.

8.1.2 Larger states (100)

The table of iterations to converge, reward and clock time for policy iteration, value iteration and Q Learning for gamma = 0.95 can be shown as follows

Label	Policy iteration	Value Iteration	Q Learning
Iterations to Converge	20	84	
Reward	62	32	10
Clock time	0.13	0.041	0.52

Thus we see a consistent behavior of policy iteration being a winner for Forest Management problem with higher reward value as well as lower iteration count. One very important observation in this case is even though policy iteration takes less number of iterations to complete it still takes more amount of clock time. This indicates that the calculations done by policy iteration are complex and more time consuming causing it to run slowly in terms of wall clock time when compared with Value iteration.

8.2 Frozen Lake Problem

8.2.1 Smaller states (4 * 4 grid)

The table of iterations to converge, reward and clock time for policy iteration, value iteration and Q Learning for gamma = 0.95 can be shown as follows

Label	Policy iteration	Value Iteration	Q Learning
Iterations to Converge	16	150	67
Reward	0.46	0.45	0.67
Clock time	0.0002	0.000001	120

From the above table it is clear that Q learning performs the best for frozen lake problem which is a subset of a grid navigation problem in terms of the score whereas policy iteration as expected takes the least number of iterations. Because of the iterative process to learn the environment Q learning seems to be taking more time everywhere. Hence for Q learning seems to sacrifice time for better rewards.

8.2.2 Larger States (20 * 20 grid)

The table of iterations to converge, reward and clock time for policy iteration, value iteration and Q Learning for gamma = 0.95 can be shown as follows

Label	Policy iteration	Value Iteration	Q Learning
Iterations to Converge	27	247	57
Reward	0.22	0.22	COULD NOT BE FOUND
Clock time	0.03	0.03	0.51

Thus above table gives a clear indication that while VI and PI match with each other on rest of the parameters solely on the basis of the iterations the winner here is policy iteration.

9 Conclusion

Form the overall analysis in this experiment we can say that the policy iteration usually runs with low number of iterations but takes more time to run as compared to value iteration due to the fact that it

does more complex calculations per iteration. Q learning gives us maximum rewards at the cost of more time owing to the fact that it takes that time to learn the environment and take actions accordingly.

10. References

https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa

https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc

https://pymdptoolbox.readthedocs.io/en/latest/api/mdp.html

https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/#:~:text=Epsilon%2DGreedy%20is%20a%20simple,a%20small%20chance%20of%20exploring

https://stats.stackexchange.com/questions/322933/q-learning-when-to-stop-training

https://github.com/maximecb/gym-miniworld