Markov Decision Processes - Code Link

Joseph Waugh
Georgia Tech ID: 903563084

CS 7641: Machine Learning
Spring 2022

April 18, 2022

1 Abstract

This assignment will cover the implementation of two different Markov Decision Processes (MDPs): Forest Management the Frozen Lake process. In comparing the two problems, the Frozen Lake process is a Grid Search problem, whereas the Forest Management solution is a non-grid world problem. These two MDPs will be solved using three different methods: Value Iteration, Policy Iteration, and Q-Learning (reinforcement learning). A comparison of the three algorithms, including the methodology and results, will be used to clarify the advantages and disadvantages of using a particular algorithm.

2 MDP Environments

2.1 Frozen Lake

The below excerpt explains the Frozen Lake MDP:

Winter is here. You and your friends were tossing around a frisbee at the park when you made a wild throw that left the frisbee out in the middle of the lake. The water is mostly frozen, but there are a few holes where the ice has melted. If you step into one of those holes, you'll fall into the freezing water. At this time, there's an international frisbee shortage, so it's absolutely imperative that you navigate across the lake and retrieve the disc. However, the ice is slippery, so you won't always move in the direction you intend.

For this report, the associated states are in two maps: 8x8 and 20x20. In this Frozen Lake MDP, the agent can move any cardinal direction, as defined in the set A:

A = {0, 1, 2, 3}
where 0 = Left, 1 = Down, 2 = Right, 3 = Up

The entire grid for the 8x8 example can be represented by a set S of the different states:

S = {0,1,2,3,4,5,. . . ,64}

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Figure 1: Frozen Lake - 4x4 Grid States

In traversing the different states based on one of the four directions, there are probabilities associated with traveling a given direction. For example, in state 10 with action 1 (down), our probabilities of moving in each direction are shown below:

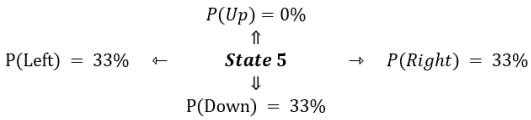

Figure 2: Frozen Lake - State Probabilities - Action 1 (Down)

When traversing through the various states, the associated reward is 0 when traversing through all states not included in the final episode state. These states are defined as the following: Frozen State (F), Goal State (G), Hole State (H), and Start State (S). Using a simpler example, the following grid can traversed to go from S to G, without falling into H.

S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

Figure 3: Frozen Lake Grid Example (4x4 Grid)

Based on this, the following paths can be taken to reach state S to state G without falling into a state H:

1 (Start) → 2 → 3 → 7 → 11 → 15 → 16(*Goal*)
1 (Start) → 5 → 9 → 10 → 11 → 15 → 16(*Goal*)

Figure 4: Frozen Lake - 4x4 Grid Traversal Paths

There are other ways to traverse the above grid, but the following confirms that there are possible methods that can be taken to reach the desired state via a MDP. Given that we're focusing on two separate grids in this report's MDP (8x8 and 20x20), we can evaluate the different number of states in each (8x8: 64 states; 20x20: 400 states) and see the impact in the final results.

2.2 Forest Management

The below excerpt explains the Forest Management MDP:

A forest is managed by two actions: 'Wait' and 'Cut'. An action is decided each year with the objective to maintain an old forest for wildlife and make money selling customers wood. Each year, there is a probability (P) that a fire burns the forest. Let 0, 1, ..., s-1 be the states of the forest, with s-1 being the oldest. Let 'Wait' be action 0 and 'Cut' be action 1. After a fire, the forest is in the youngest state, that is state 0.

This problem isn't explicitly defined using a grid. Instead, there are multiple states that are to be used to solve this problem. For this analysis a 5 state scenario was used for comparison. The following age-classification bins can be used for the 5 different states of this forest management problem:

- age of trees: 0-1 year (state 1)
- age of trees: 1-2 years (state 2)
- age of trees: 2-3 years (state 3)
- age of trees: 3-4 years (state 4)
- age of trees: more than 4 years (state 5)

In this scenario, state 5 is classified as the oldest age-class. At the end of a period t, if state s occurs at period t with the action *Wait*, then the next period will be $min(s+1, 5)$ if a fire has not occurred; however, a probability remains of a fire to occur, thus allowing the option of returning back to state 1 (0-1 year). This report assumes that a probability of a wildfire is 10 percent (0.1), thus the challenge is determining how to manage the goals and maximize the reward associated with the problem.

3 Markov Decision Process Solvers

3.1 Policy Iteration

The policy iteration solver focuses on two primary ideas to determine an optimal solution for the MDP. Specifically, this algorithm uses policy evaluation and policy improvement, where improvement over a prior policy is saved via iteration until the policy is deemed "stable", which means changes are no longer observed when following a specific sequence.

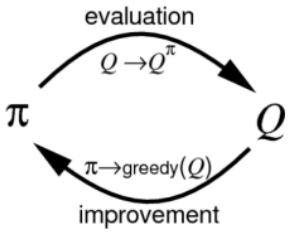

Figure 5: Policy Iteration Cycle

Policy iteration often works well in terms of converging to an optimal result in a fewer amount of iterations; however, each of these iterations requires policy evaluation, which requires a complete evaluation of the entire set of states in multiple iterations, which can be computationally and time consuming before reaching the converged result.

3.2 Value Iteration

The value iteration solver differs from the policy iteration algorithm. While the policy iteration algorithm focuses on finding the best policy given a current evaluation, the value iteration algorithm focuses on achieving the best evaluation given the best available policy. This process utilizes step-by-step processes for policy evaluation and improvement, and ultimately combines the two formulas to solve a MDP. Each iteration doesn't fully complete, but rather it is improved until convergence. Value Iteration still allows for convergence to optimal values, and is especially beneficial for smaller grid-world problems, as fewer states result in a less computationally-expensive process for computing expectimax for each state. With that being said, if the number of states are large, the value iteration algorithm requires huge amounts of computational power and time to converge to an optimal result.

3.3 Q-Learning

The Q-Learning solver focuses on a fixed-policy solution by determining the expected value of a given action in a particular state. This solver is able to compare the expected value, of all actions without having to necessarily model the environment, as the "reward" for a given state traversal (i.e., moving from a single state to a new state, and evaluating whether or not this was a positive or negative move) helps to determine how the algorithm improves via reinforcement learning.

4 Results

4.1 Frozen Lake

For the Frozen Lake algorithm, two different state grid-world sizes were used for evaluation: 8x8 and 20x20.

4.1.1 Policy Iteration

To assess this MDP via policy iteration, various gamma (discount rate) values were used: 0.9999, 0.99, 0.9, and 0.5.

A standard 10,000 episodes were assessed for measuring the impact on the discount rate, and 100 iterations were used for this process.

The first stage of this analysis was determining the impact of discount rate (gamma). The obtained results suggest that a higher discount rate leads to a larger overall average mean reward. Therefore, in this 8x8 grid-world MDP, it makes sense to include a large gamma value for optimized results. Based on the above visualization, a Gamma value of 0.8 results in roughly 80 percent success at arriving at the Goal state without falling into the Hole states in the grid.

Figure 6: Discount Rate Average Reward Mean 8x8 Grid

The same analysis was performed for the 20x20 grid, and thus yielded similar results in terms of a trend, but included much lower reward values due to likely not being able to converge to an optimal solution; however, the trend of a positive reward with a larger gamma value still suggests that a larger gamma value works in reaching the best possible result. To fully assess if this is true for a larger range of gamma values, the same experiment was performed with a new set of gamma values: [0.9999, 0.99999, 0.999999, 0.9999999], which yielded the following results.

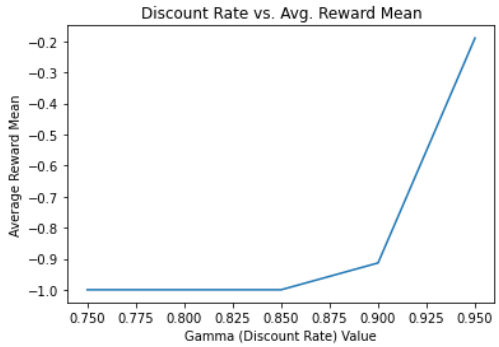

Figure 7: Discount Rate Average Processing Time 8x8 Grid

Again, we continue to see a positive trend in terms of the average reward based on the increased gamma values. Another area to research is the impact of increasing Gamma with regards to processing time to converge to the optimal solution (or reach the maximum number of iterations specified). The same experiment was performed for our initial 8x8 grid, and suggests that a larger value for Gamma results in increased processing time at what appears to be a log scale.

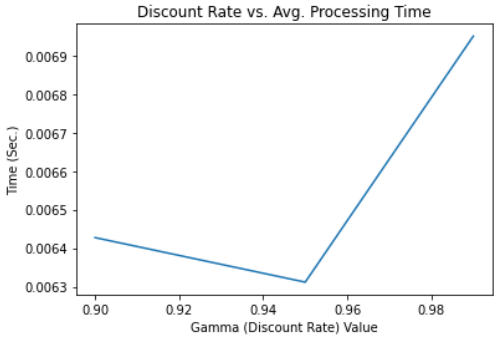

Figure 8: Discount Rate Average Processing Time 8x8 Grid

When performing the same experiment for our 20x20 grid, these results are further confirmed that an increased Gamma value increases the processing time. Therefore, this becomes an important consideration to factor in the reward of increased accuracy, while also having to expend more time and computational power to achieve these results with the higher values for Gamma. While the results obtained here suggest that an increase in Gamma, which yields a positive impact in terms of the success of the final result, there's only a marginal increase in the overall time required to converge to an optimal solution. This suggests that this marginal increase isn't enough to warrant any considerations towards decreasing gamma for smaller grid problems (assuming our 8x8 and 20x20 are both considered small); however, if we exponentially increase the size of our grid, this can play a significant role in the computational resources required to process the problem.

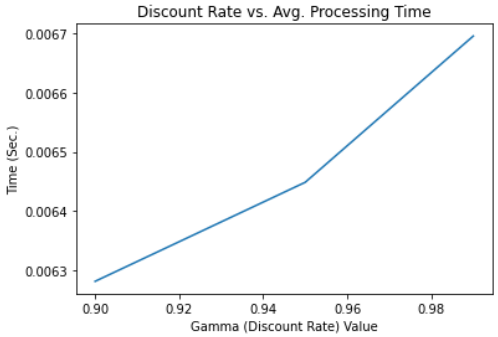

Figure 9: Discount Rate Average Processing Time 20x20 Grid

4.1.2 Value Iteration

For the Value Iteration solver, again the 8x8 grid and the 20x20 grid were assessed in terms of the discount rate vs. average reward mean and processing time.

In assessing this MDP via value iteration, the following gamma rates were used: 0.9999, 0.99, 0.9, and 0.5. Again, our 10,000 episodes and 100 iterations were used in this process. In beginning our initial analysis for the impact of gamma versus the discount rate, we were again able to

see that for this solver, a larger Gamma rate leads to an increased average reward mean.

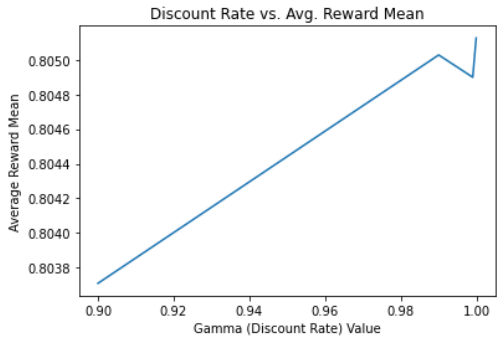

Figure 10: Discount Rate Average Reward Mean 8x8 Grid

Again, iterating through this process with our larger 20x20 grid, we see the same trend: a larger Gamma value results in an increased average reward.

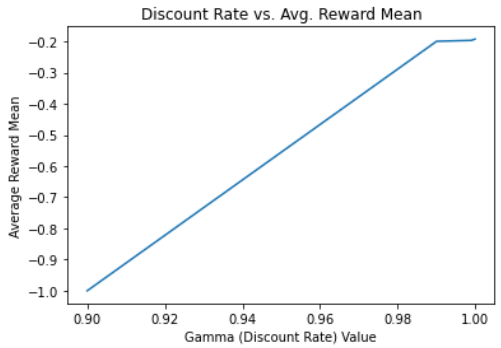

Figure 11: Discount Rate Average Reward Mean 20x20 Grid

Next, our second test focused on the impact of our discount rate vs. the average processing time. Similar to what we viewed in the Policy Iteration results, a higher gamma rate results in increased processing time. Again, we are able to see with the 8x8 grid an increased gamma rate also increased the required time to converge to an optimal result.

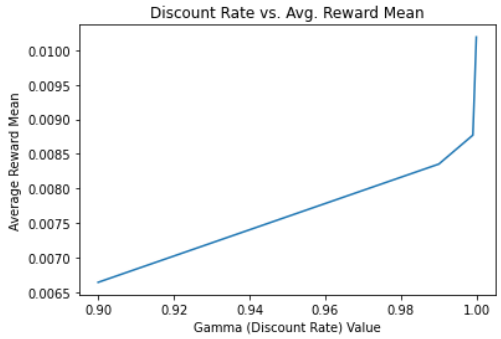

Figure 12: Discount Rate Average Reward Mean 8x8 Grid

This same conclusion is the result for the 20x20 grid. Again, this consideration doesn't have a huge impact on smaller grid-world MDP problems, but rather should be an area of consideration for larger grids.

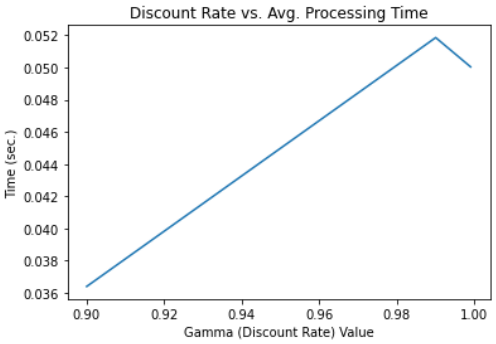

Figure 13: Discount Rate Average Reward Mean 20x20 Grid

4.1.3 Q-Learning

To analyze the Q-Learning algorithm, I focused on changes to the number of episodes, decay rate, and time required for convergence. To begin, the initial starting parameters that were used include our 8x8 grid and 20x20 grid. The default number of iterations in this portion of the work was 900.

In our 8x8 grid, we're able to see that the increased number of episodes has a positive impact on the final mean target value. This visualization is shown below, specifically with 900 and 100 episodes.

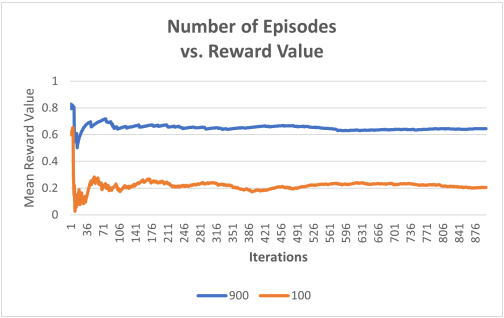

Figure 14: Number of Episodes Average Reward 8x8 Grid

The same trend can be observed when focusing on the 20x20 grid. This trend holds true for this example; however, the issue here is that the results fail to converge, as shown in the average target value hovering around -1.

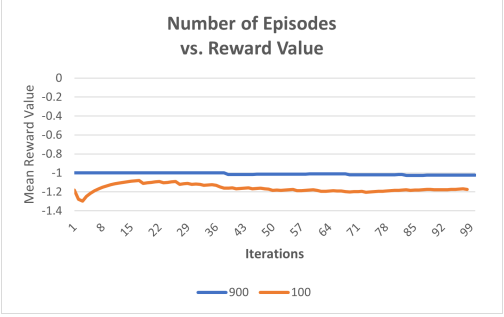

Figure 15: Number of Episodes Average Reward 20x20 Grid

Nonetheless, the conclusion of an increased number of episodes still suggests that this will provide an optimal result. The resulting path traversal obtained for our 900 iteration test with Q-Learning in the 8x8 grid is shown below:

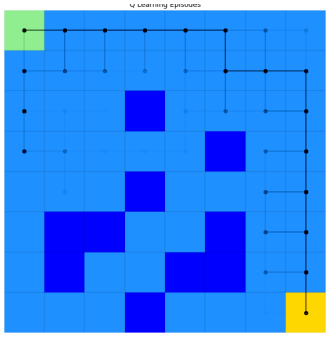

Figure 16: Q-Learning Path Traversal Example 8x8 Grid

The next item to review is the impact with decay rate on the overall target score for this solver. Specifically, when focusing on our example with the 8x8 grid, we can see a trend of a decreasing alpha and increasing number of iterations leads the result towards convergence, as 250 iterations were required to reach convergence.

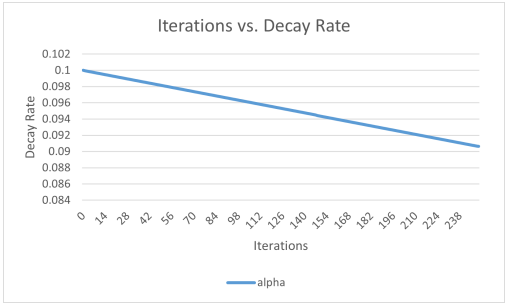

Figure 17: Q-Learning Iterations and Decay 8x8 Grid

This same trend was observed for our 20x20 trend, and it could likely be attributed to a couple of items: first, the decaying alpha value allows the convergence to the optimal result to not miss the target value as more iterations pass, wheras early on in the process the model can take larger "steps" in order to close the potential gap of where this result might be found.

In order to compare the time of this Q-Learning algorithm, I reviewed the number of iterations required for convergence in addition to the overall time required to reach convergence among the three solvers. Specifically, the Q-Learning method takes the longest in terms of iterations and in the average time required to reach convergence.

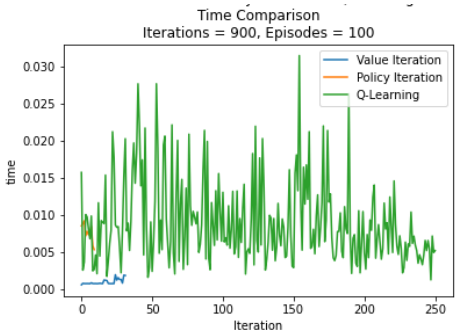

Figure 18: MDP Mean Time Required 8x8 Grid

In looking at these two graphs, it's clear to see that the overall time required (looking at the Y-axis) is the highest among the three MDPs, and the number of iterations required also is the largest for the Q-Learning MDP.

4.2 Forest Management

For the Forest Management MDP, I utilized a 5-state environment for comparative analysis. In addition, I used a variable discount rate (ranging from 0.5 to 0.95) in order to test the incremental changes to the model. In a future study, it would be possible to increase the $r_1 and r_2 values in order to reach the final state and still maintain a positive state environment for comparative analysis to analyze a larger-state problem.$

4.2.1 Policy Iteration

To begin, the Policy Iteration implementation via mdptoolbox was able to output the same policy continuously in ¡ 0.1 seconds. In terms of iterations, our initial implementation was able to achieve a result in 4 iterations of testing. Our implementation with a lower decay value resulted in less iterations required to reach the optimal result. Nonetheless, the small number of iterations required may mean that the incremental small changes to the policy in each iteration is the optimal methodology when utilizing this type of environment.

In focusing on our 500 state example, we were able to achieve a successful result in only 2 iterations, equalling 0.019 seconds to reach the optimal solution. Given that the large increase in states doesn't amount to significant computational power changes required, this appears to be a good algorithm that can be utilized regardless of state size.

4.2.2 Value Iteration

In comparison to Policy Iteration, Value Iteration performed very similarly to PI, with the focus here being an increase in the required iterations to determine the optimal policy. For example, the same optimal policy was identified when utilizing Policy Iteration, but the below chart showcases that the number of iterations in the VI method is larger. In situations where larger state sets need

to be traversed, this can potentially require much more computational power versus a different methodology.

Next, focusing on the 500 state example, we were able to achieve a successful result in 4 iterations with a processing time of ¡ 0.1 seconds. Similar again to Policy Iteration, the computational power doesn't have any sort of limiting factor with an exponential increase in the number of states.

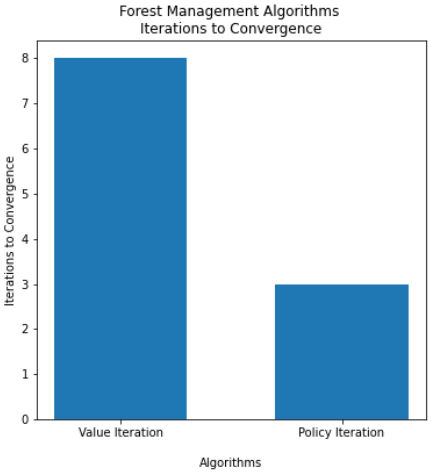

Figure 19: Convergence Iterations
5-State Environment

4.2.3 Q-Learning

Q Learning differs from the Policy and Value Iteration strategies, given that the environment needs to be traversed. In order to test the differences for this type of solver, various types of "learning rate decay" values were tested to see which provides the optimal result. The overall Q-Learning result appears to differ from Value and Policy Iteration results, given that the results appear to change significantly in the outer states.

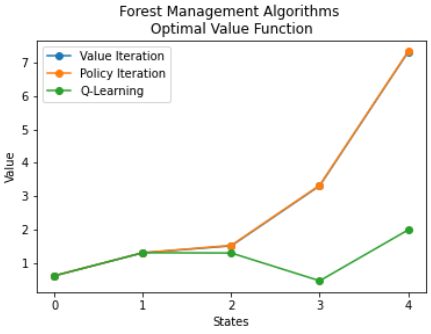

Figure 20: Optimal State Values (Decay = 0.25)
5-State Environment

In comparing these results to a higher decay value, we can see that the results remain similar across the three models in the same 5-state environment. Therefore, the higher decay values allows for increased values for each progressive state, in addition to allowing all of the algorithms to perform similarly.

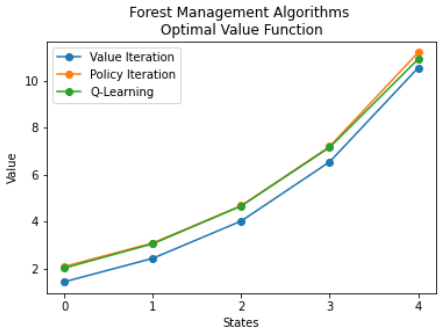

Figure 21: Optimal State Values (Decay = 0.7)
5-State Environment

Next, comparing these results to the results obtained in the 500-state instance, we see massive increases in processing time required with the change in number of states. In our original 5-state solution, the associated processing time was 0.23 seconds, whereas for our 500-state solution, the new processing time is now 0.5 seconds. The processing time has more than doubled, and thus, this should be a consideration when determining which algorithms are optimal for larger-state environments.

5 Conclusion

In conclusion, there are three different methods that were used in this Markov Decision Process analysis. Both model-based solutions (Policy Iteration and Value Iteration) benefited due to utilizing inputs from the model in order to produce optimal policies. The optimal nature of these models is shown by the strong mean reward values for each of these, in addition to the mean time spent for training each of these models.

Q Learning proved to be more difficult, purely as a result of the environment having to react to the model rather than having a direct interaction with just the model instead. Due to having to interact with the entire environment, there are instances where the Q-Learning algorithm fails to rach the optimal solution (i.e., the Frozen Lake environment with the larger grid size.

When analyzing the Frozen Lake vs Forest Management environments, it's clear to see that the grid-based environment is easier for the Policy Iteration examples due to the "connectedness" of the policies. In these instances, increasing the overall size of the environment doesn't play a massive role in the achieved result; however, this proves to be a difficult strategy for Q-Learning instead.

6 References

- Markov Decision Process (MDP) Toolbox link