Markov Decision Processes and Reinforcement Learning

Tichakunda Mangono

CS7641 Machine Learning Paper 4, April 15th, 2019

Introduction

Machine Learning is largely made up of Supervised, Unsupervised, and Reinforcement Learning (RL). While the first two involve function approximation and function/data description respectively, Reinforcement Learning is about learning to make decisions and take actions to optimize rewards. The decision-making processes of RL are modeled well using Markov Decision Processes (MDP) a mathematical model of discrete time stochastic control processes where outcomes are partly random and partly under the control of a decision maker[1]. Therefore, RL allows us to model a world where an agent makes these decisions, takes actions and we can observe the rewards and develop the optimal policy for the agent to maximize rewards and achieve desired outcomes. Some of the most popular applications for RL include robotics, gaming (checkers, chess, backgammon, etc.), and even self-driving cars and unmanned flying vehicles. Reinforcement Learning allows us to understand and solve dynamic real-world problems at a lower cost.

Objective

This paper will introduce two important MDP problems and use both value and policy iteration to study the nature, quality and performance during convergence to a solution. Value iteration is computes the optimal state value function by iterating through and improving the estimated value function across all states (note that value includes reward from current state as well as delayed future rewards which are discounted at some rate). Policy iteration, however, directly optimizes the policy starting from a random policy and improving the policy at each state to maximize the value function. This paper will study the effect of problem size, number of states and complexity on the convergence of each algorithm.

Finally, both problems will be solved using Q-Learning, a choice algorithm for reinforcement learning to compare the outcomes. Q-Learning is a model-free type of learning algorithm where the agent is assumed to start from scratch, without any intrinsic knowledge or model of the environment (states, transition, and rewards), but rather the agent will discover positive and negative actions through trial and error. By using policy iteration, value-iteration, and Q-learning, this paper will successfully examine the performance of two planning approaches and one learning approach.

This paper will investigate two interesting problems: one simple one – forest management with a few states and actions, and one hard one – grid-world with several states, actions and obstacles. The problems are described and studied below.

Problem 1: Forest Management over 5 years

The problem of managing forests in fire-prone areas has significant implications for the ecology and species depending on the forest environment. In this simple example, there are two main objectives or end points i.e. to manage an important forest by conserving the trees and therefore the wildlife species that depend on it and to also generate enough income by cutting the trees and selling them for wood in order to invest

[1] https://en.wikipedia.org/wiki/Markov_decision_process

that money to pay for forest maintenance, staff etc. Thus, the forest ranger's two actions are to either "wait" or "cut" the forest each year but there is also a chance that the forest will burn down with some probability specified for each year. The probability/transition matrix was as follows:

```
array([[[0.3, 0.7, 0. , 0. , 0. ],
        [0.3, 0. , 0.7, 0. , 0. ],
        [0.3, 0. , 0. , 0.7, 0. ],
        [0.3, 0. , 0. , 0. , 0.7],
        [0.3, 0. , 0. , 0. , 0.7]],

       [[1. , 0. , 0. , 0. , 0. ],
        [1. , 0. , 0. , 0. , 0. ],
        [1. , 0. , 0. , 0. , 0. ],
        [1. , 0. , 0. , 0. , 0. ],
        [1. , 0. , 0. , 0. , 0. ]]])
```

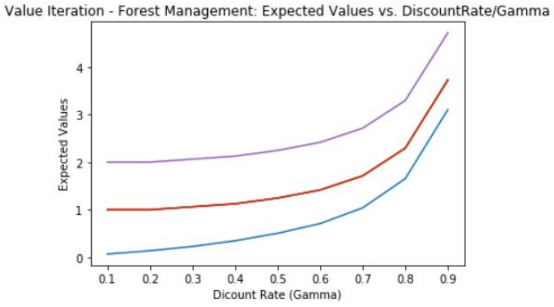
And the rewards matrix was as follows, increasing the value of the forest in the later years.

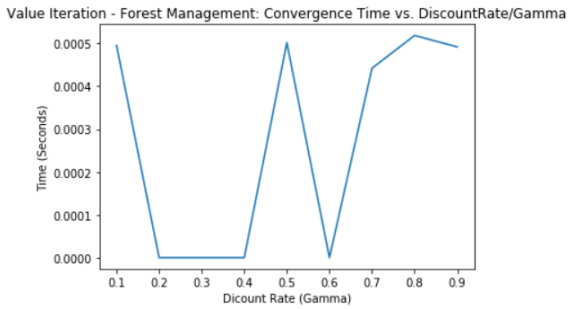
```
array([[0. , 0. ],
       [0. , 1. ],
       [0. , 1. ],
       [0. , 1. ],
       [0.3, 2. ]])
```

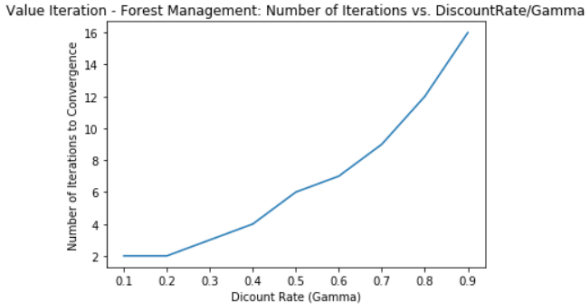
Problem 1 - Value Iteration Solution

Value Iteration - Forest Management

Discount/Gamma	*Year1*	*Year2*	*Year3*	*Year4*	*Year5*
0.1	0	1	1	1	1
0.2	0	1	1	1	1
0.3	0	1	1	1	1
0.4	0	1	1	1	1
0.5	0	1	1	1	1
0.6	0	1	1	1	1
0.7	0	1	1	1	1
0.8	0	1	1	1	1
0.9	0	1	1	0	1





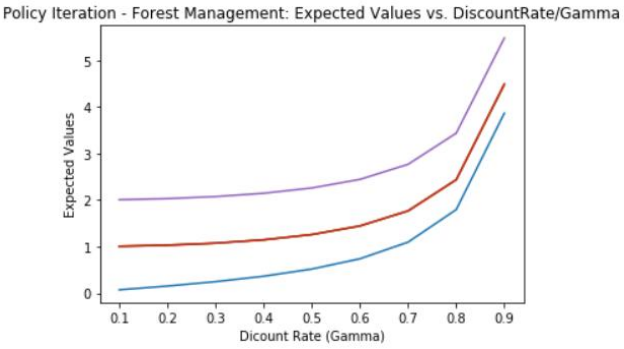


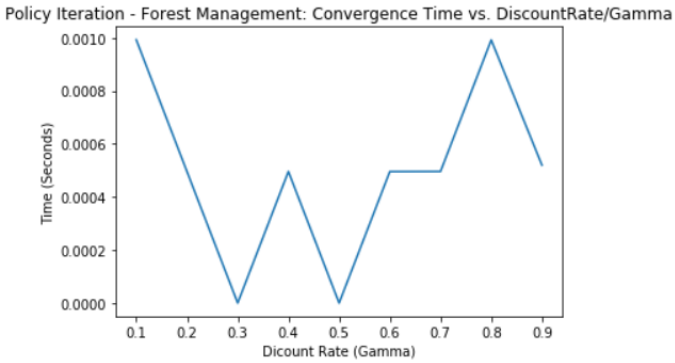
As shown above the value iteration solution converged to a policy recommending to let the forest grow in the first year and then cut it every year thereafter. This policy is optimal for all levels of the discount factor except the high level of 0.9 when the optimal policy is to (0,1,1,0) i.e. ("wait", "cut", "cut", "wait", "cut"). The expected values increase monotonically each year as time passes, since the optimal policy picks the maximum value action each year. The expected values also generally increase with the discount value, this is because a higher discount factor emphasizes future rewards more. The convergence time for value iteration is generally 500 microseconds (0.0005s) for more discount values, but it falls to 0 for discount factors of 0.2,0.3,0.4, and 0.6. In terms of number of iterations, the value iteration method takes as few as 2 iterations to converge when discount factor is low at 0.1 or 0.2, then this increases to 16 when the discount factor is high at 0.9. This is because high discount factors require the algorithm to search over a bigger space of future actions and rewards as they are more significant in this scenario. While the relationship between number of iterations and discount factors starts out looking linear at the lower ends, it starts to curve and scale upwards steeply in almost a polynomial fashion.

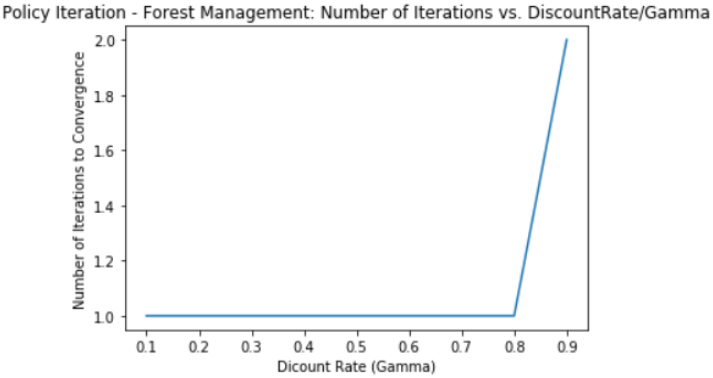
Problem 1 - Policy Iteration Solution

Policy Iteration - Forest Management

Discount/Gamma	*Year1*	*Year2*	*Year3*	*Year4*	*Year5*
0.1	0	1	1	1	1
0.2	0	1	1	1	1
0.3	0	1	1	1	1
0.4	0	1	1	1	1
0.5	0	1	1	1	1
0.6	0	1	1	1	1
0.7	0	1	1	1	1
0.8	0	1	1	1	1
0.9	0	1	1	0	1





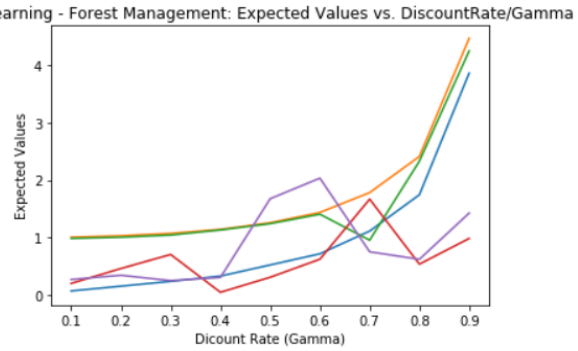


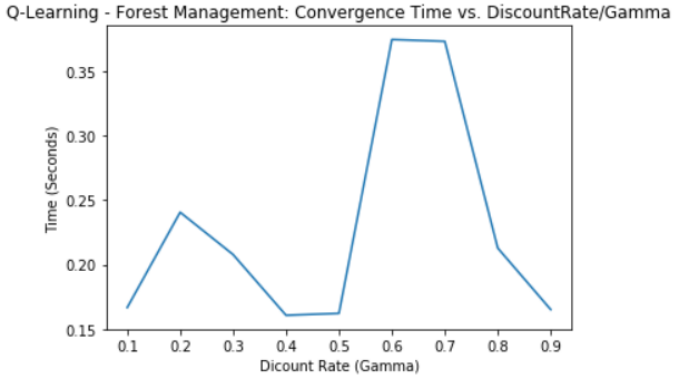
Using Policy Iteration, the solution/policy converged to a policy recommending the same as was recommended by the value iteration algorithm i.e. wait the first year then cut thereafter except when discount factor is high at 0.9 when it's best to wait in Year4. The expected values charts for Policy Iteration show the same monotonically increasing patterns across the years and as discount factor increases, the only difference in that the increase in expected values is much steeper in from 0.8 to 0.9 for the policy iteration. The convergence time for policy iteration ranges from 400 microseconds (0.0004s) to 1000 microseconds (0.001s), but is very low (almost 0) for discount factors of 0.3 and 0.5. This is comparable to value iteration. The number of iterations for policy iteration takes only 1 iteration to converge for all discount factor expect 0.9 when it needs two iterations to converge. This is significantly lower than value iteration because there are only a few tweaks to be done to the initial policy to find the optimal one i.e. there are less policies/paths permutations than there are values. So the relationship between number of iterations and discount factors for policy iteration looks like a step function.

Problem 1 - Q-Learning Solution

Q-Learning - Forest Management

Discount/Gamma	Year1	Year2	Year3	Year4	Year5
0.1	0	1	1	1	0
0.2	0	1	1	1	1
0.3	0	1	1	1	0
0.4	0	1	1	0	0
0.5	0	1	1	0	1
0.6	0	1	1	0	1
0.7	0	1	0	1	0
0.8	0	1	1	1	0
0.9	0	1	1	1	0



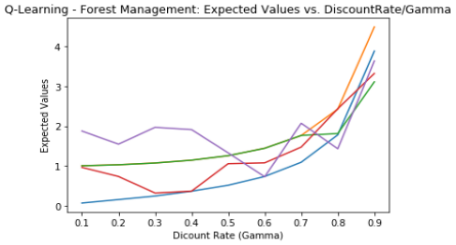


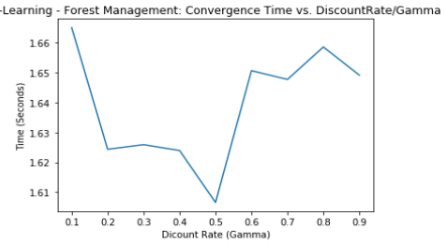
After applying the planning methods of value and policy iterations, a learning approach (Q-Learning) was applied to problem and it had significantly different results. The optimal policy solution recommended to wait in year1, cut in year2 and cut in year3 (except when discount factor is 0.7). This is mostly in agreement with the value and policy iterations. The Q-learning algorithm recommends less cutting of trees in year4 and year5 than the value and policy iterations. There is no clear patterns on the expected values except that they tend to increase slightly as discount factor grows but they show a lot more variation than the monotonically decreasing ones for both value and policy iteration. The time to computation time/convergence for Q-Learning is 150,000-400,000 microseconds (0.15-0.4 s) which is very long and ~1000X times both the time to convergence for both value and policy iteration.

In order to investigate different learning tactics, I looked at increasing the number of iterations for the Q-Learning algorithm since the version I was using (pymdptoolbox) did not have learning rate or step as a hyperparameter.

Q-Learning - Forest Management

Discount/Gamma	Year1	Year2	Year3	Year4	Year5
0.1	0	1	1	1	1
0.2	0	1	1	1	1
0.3	0	1	1	0	1
0.4	0	1	1	0	1
0.5	0	1	1	1	1
0.6	0	1	1	1	0
0.7	0	1	1	1	1
0.8	0	1	0	1	0
0.9	0	1	0	0	0





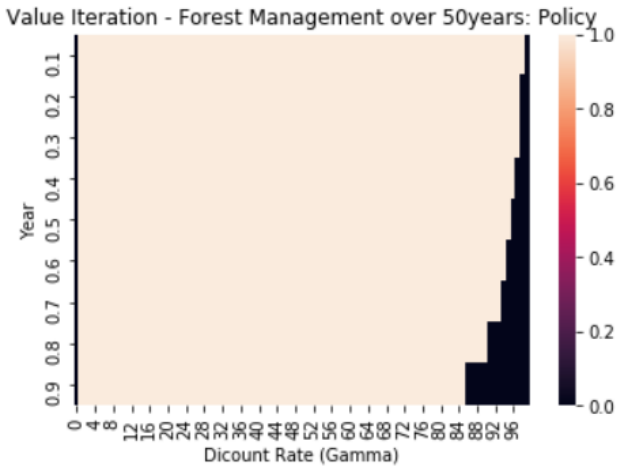
Theoretically, the more iterations, the more information the Q-Learning algorithm learns thus the closer it gets to the real optimal solution. I increased the number of iterations from 10,000 to 100,000 to reveal different results again. As such, it didn't seem like the Q-Learning solution is stable as the number of iterations increase. The new policy solution suggests to wait and not cut the forest, especially when the discount factor is very high (0.8 or 0.9) and when the forest is older (year3-yea5) which makes sense because the forest accumulates value and that value is emphasized by a high discount factor. Obviously,

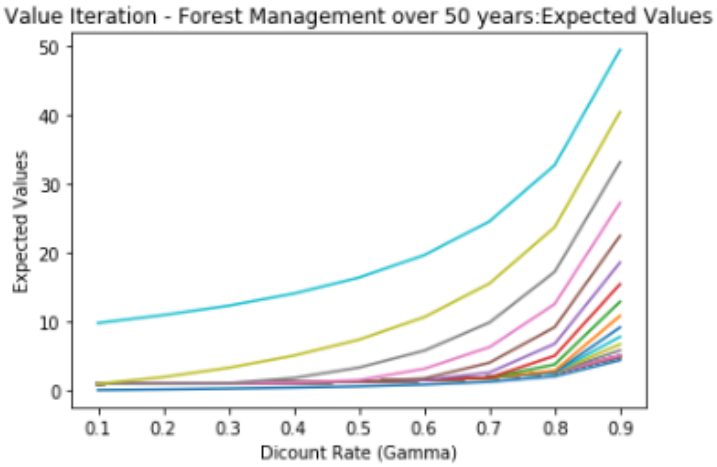
more iterations mean more computational time, now ~1.6 million microseconds (1.6 s). So a 10x increase in iterations resulted in a 4X increase in computational time.

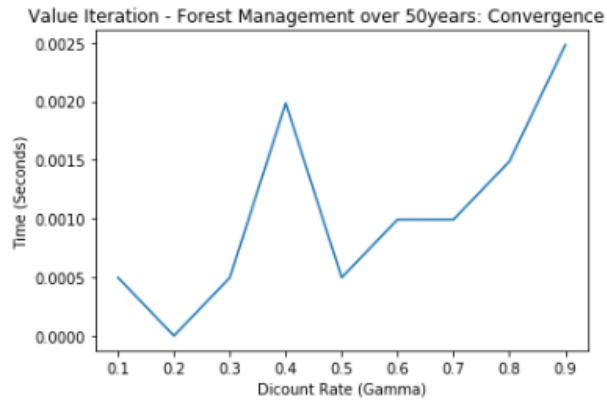
Problem 2: Forest Management over 100 years

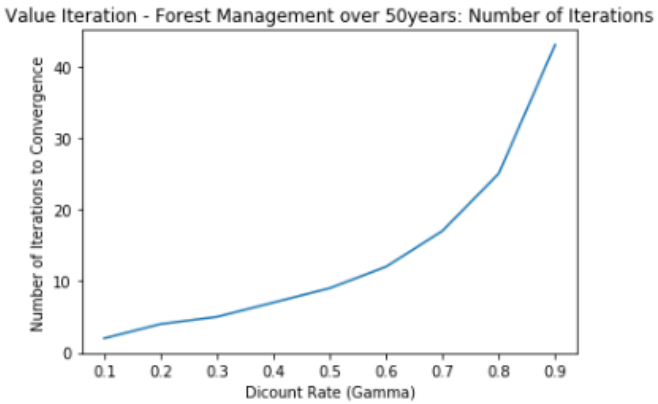
Having investigated the forest problem over 5 years, it made sense to see how this would change outcomes and performance over the longer horizon i.e. forest management over 100 years. This problem is the same as the first problem, except that this is now over a longer time horizon of 100 years, thus making the problem more complex with higher number of states, and different transition and rewards configurations. The general direction of increasing value was maintain i.e. the forest increases in value. However, the probability of a fire gutting down the forest in any given year was set to 10%, which is lower than the 30% and 70% from the previous problem

Problem 2 -Value Iteration Solution



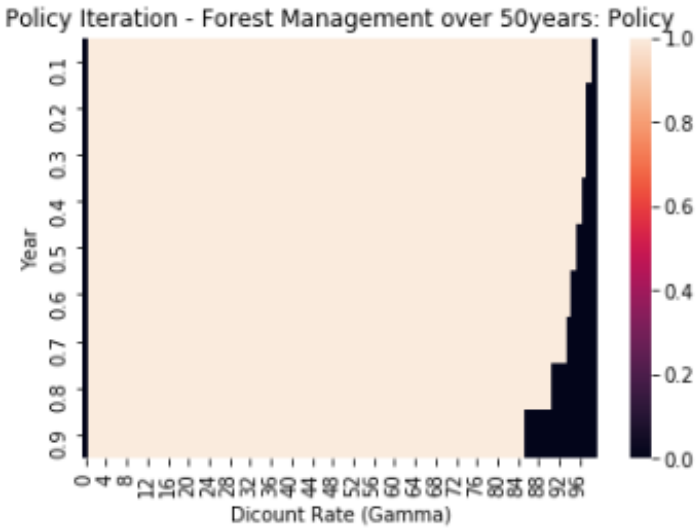


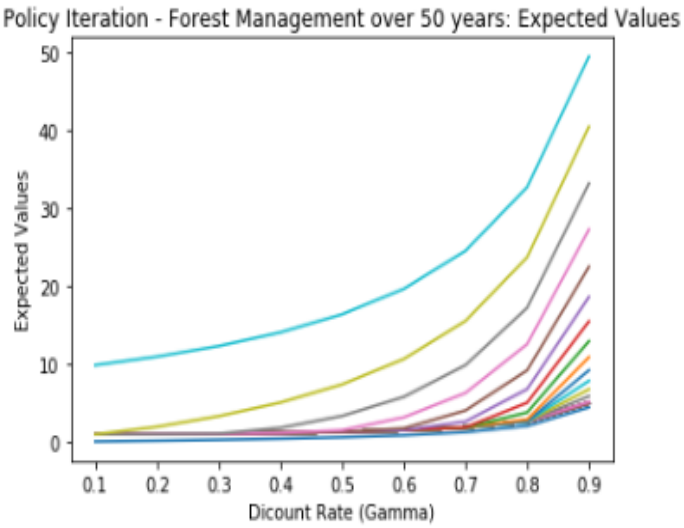


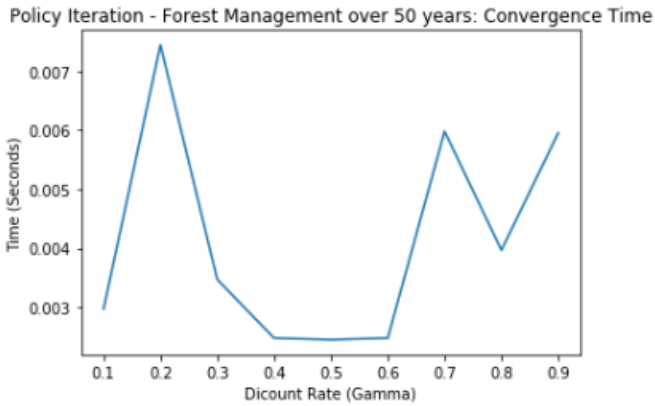


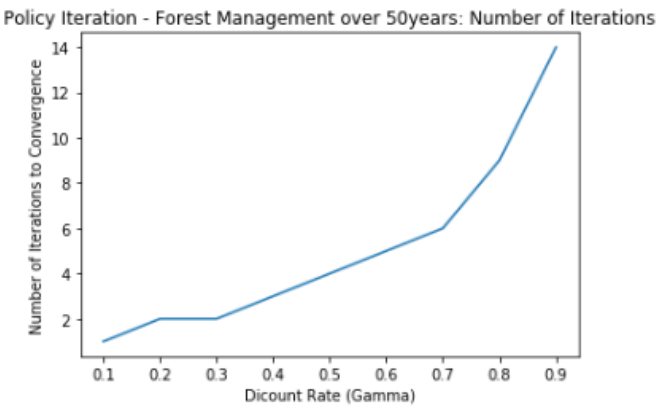
The value iteration solution for 100 year forest management converged to a policy recommending to let the forest grow in the first year and then continue to cut it until the final years when it has gained a lot of value. When discount factor is low, you generally cut the forest for more years until the very last years. But when discount factor is high, you stop cutting the forest around year84/85. Similar to the 5 year management problem, in the 100 year problem, the expected values increase monotonically with both the year and discount value because a higher discount factors emphasizes future rewards more. The convergence time for value iteration ranges from 500-2500 microseconds (0.0005-0.0025s). In terms of number of iterations, the value iteration method takes as few as 2-5 iterations to converge when discount factor is low at 0.1 or 0.2, but this increases to 40 when the discount factor is high at 0.9 showing similar but more pronounced polynomial-like scaling pattern just like the value iteration in the 5-year problem.

Problem 2 - Policy Iterations Solution



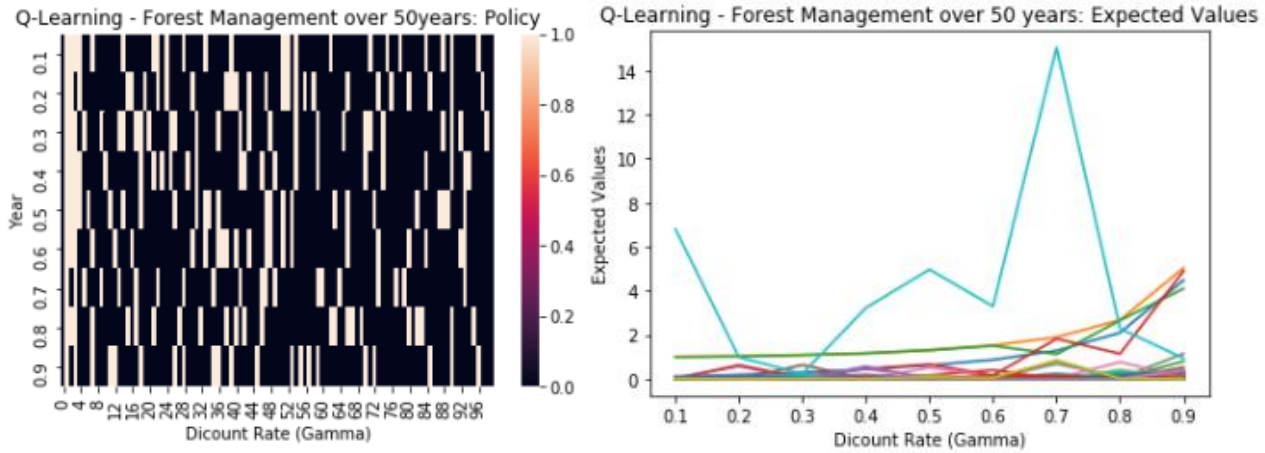


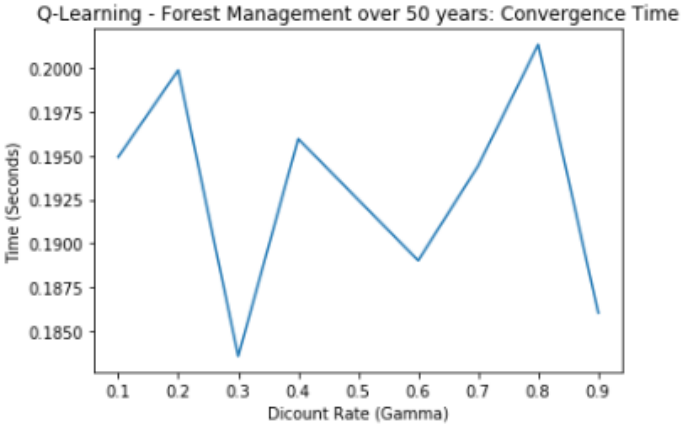




For this 100-year problem, policy Iteration converges to an identical optimal policy solution to the value iteration just like for the 5-year management problem. Likewise, the expected values over time and as discount factor increases is similarly increasing, which validates the convergence to an identical solution. The convergence time for policy iteration ranges from 3000-7000 microseconds (0.003-0.007s) which are all higher (1.5X-4X) times than value iteration. So as size of problem increases, the policy iteration starts to take longer than value iteration for this problem. This is because increasing the size of the problem to 50 means there are at most 2^50 distinct policies (exponential increase) while the number of values to be iterated through for the value iteration does not increase this much. For the same reason the computation time for convergence of 50 year solution with policy iteration is more than for the 5 year problem with policy iteration. In terms of number of iterations, the policy iteration method takes from 2-14 iterations, increasing with the discount factor for the same reasons explained before in previous sections.

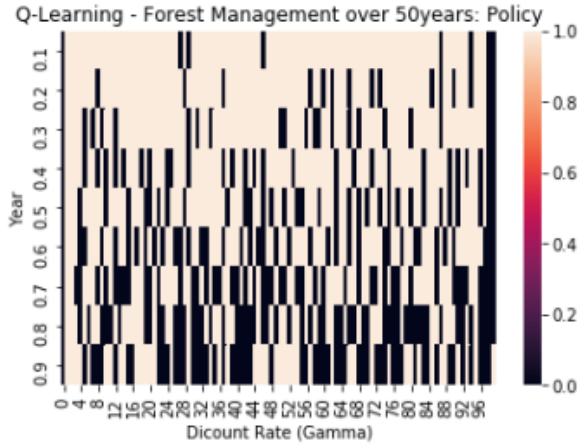
Problem 2 - Q-Learning Solution

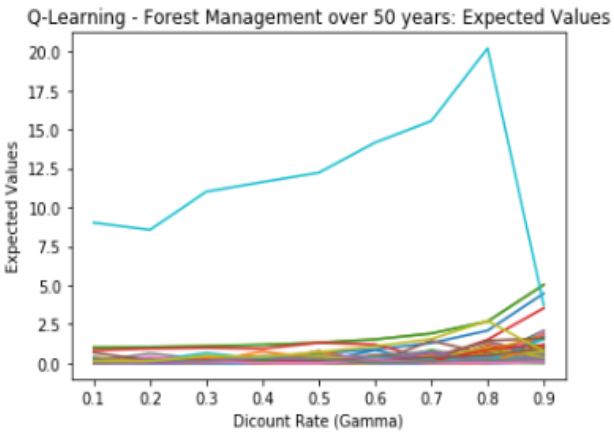


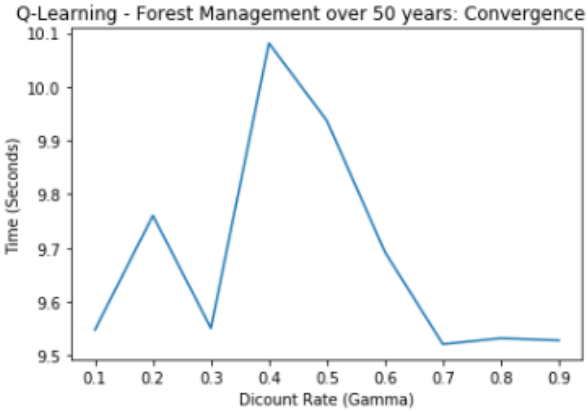


Using Q-Learning to solve the problem (with 10,000 iterations), it gave a very different solution from value and policy iteration. There was no discernable pattern in the expected values except that the they tended to be high for later year, especially the last two years or so. The algorithm took a full 0.18-0.20 seconds to converge which is >20X times than the Q-Learning for the 5-year problem and therefore much bigger in scale than the convergence time of any other algorithms seen thus far. This is still about 400X longer than the corresponding value and policy iterations.

I also decided to experiment with the number of iterations (increased from 10,000 to 500,000) for the Q-Learning algorithm and got a different solution which began to show more clearly the pattern of waiting cut the forest when the forest is older and when the discount factor is higher. The expected values began to show more of a pattern but the convergence time of course increased to 9-10 seconds, which is about 50X times the time it took with 10,000 iterations. See charts below or on next page:

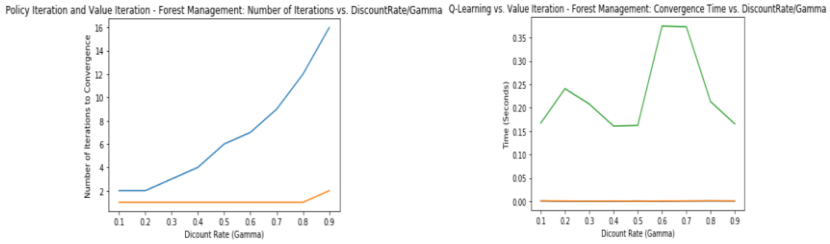


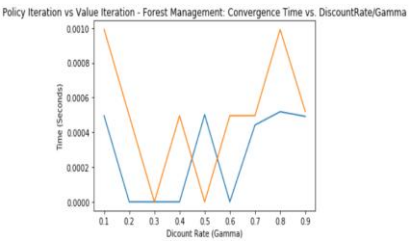




Final Comparison & Conclusion

Problem 1: Forest Management over 5 years





Looking at the time and number of iterations for a smaller problem, value and policy iteration had comparable computation time. However, policy iteration always had less number of iterations than value iteration. Q-Learning took significantly longer than both and had different solutions which tended to agree with value and policy iteration as you increased the number of iterations (i.e. as the algorithm learned more).

See more analysis below:

Value Iteration - Forest Management

Discount/Gamma	Year1	Year2	Year3	Year4	Year5
0.1	0	1	1	1	1
0.2	0	1	1	1	1
0.3	0	1	1	1	1
0.4	0	1	1	1	1
0.5	0	1	1	1	1
0.6	0	1	1	1	1
0.7	0	1	1	1	1
0.8	0	1	1	1	1
0.9	0	1	1	0	1

Policy Iteration - Forest Management

Discount/Gamma	Year1	Year2	Year3	Year4	Year5
0.1	0	1	1	1	1
0.2	0	1	1	1	1
0.3	0	1	1	1	1
0.4	0	1	1	1	1
0.5	0	1	1	1	1
0.6	0	1	1	1	1
0.7	0	1	1	1	1
0.8	0	1	1	1	1
0.9	0	1	1	0	1

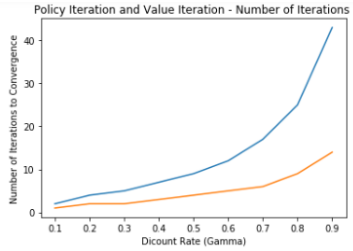
Q-Learning - Forest Management

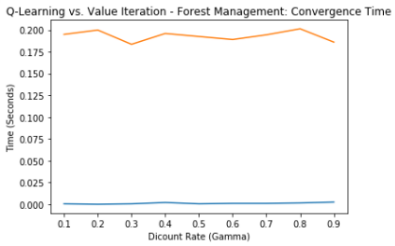
Discount/Gamma	Year1	Year2	Year3	Year4	Year5
0.1	0	1	1	1	0
0.2	0	1	1	1	1
0.3	0	1	1	1	0
0.4	0	1	1	0	0
0.5	0	1	1	0	1
0.6	0	1	1	0	1
0.7	0	1	0	1	0
0.8	0	1	1	1	0
0.9	0	1	1	1	0

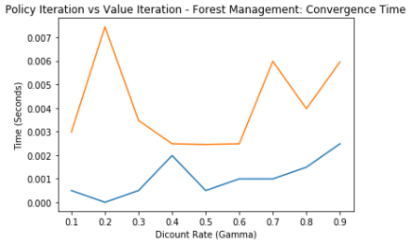
Q-Learning - Forest Management

Discount/Gamma	Year1	Year2	Year3	Year4	Year5
0.1	0	1	1	1	1
0.2	0	1	1	1	1
0.3	0	1	1	0	1
0.4	0	1	1	0	1
0.5	0	1	1	1	1
0.6	0	1	1	1	0
0.7	0	1	1	1	1
0.8	0	1	0	1	0
0.9	0	1	0	0	0

Problem 2: Forest Management over 5 years







Looking at the computation time for a bigger problem, policy iteration had more computation time until convergence than value iteration. Like before Policy iteration always had less number of iterations than value iteration. Q-Learning still took significantly longer than both and had different solutions which tended to agree with value and policy iteration as you increased the number of iterations (i.e. as the algorithm learned more).