Assignment 4: Markov Decision Processes for CS7641 Fall 2021

by Jaeyong Kim

Introduction

Markov Decision Processes and Reinforcement Learning offer a different approach to solving problems compared to the other Machine Learning spaces of Supervised and Unsupervised Learning. While other algorithms work with true examples and learning from those examples, RL and MDP learners work with the environment by observing the outcomes of certain actions very similar to a trial-and-error kind of approach to learning. RL algorithms are often looking for a reward for each action taken within the environment with the overall goal being selecting the best set of actions that may lead to the maximum reward possible. This best set of actions is called the optimal policy and each of the algorithms used in this assignment seek to find the best policy. MDP are defined by a state transition matrix, P, that has probabilities of a transition from some state, s, to some state, s', and a reward matrix, R, that defines the reward of going from one state, s to another state, s'.

Algorithms

Three different algorithms will be analyzed in this paper to find the optimal policy: Value Iteration (VI), Policy Iteration (PI) and Q-Learning (Q).

Value Iteration starts with a random value function and iterate until we find the best value function. Once that value function is found, the optimal policy is then extracted from that value function. Policy Iteration starts with a random policy until it finds the optimal policy at convergence. Like VI, it utilizes a one step look-ahead to see if the next iteration is better than the last one. PI is generally known to take less time and require less iterations to obtain the optimal policy and it will always converge given a discrete MDP. A big parameter used in these algorithms is gamma which is the discount rate. A gamma closer to 1 will prioritize future rewards, while a gamma near 0 prioritizes short-term rewards. Both algorithms will use a transition state matrix so that it knows what the probability of going from one state to another state is given some action.

Q-learning is slightly different where the agent needs to take an actual action given a certain state and will observe if that state and action gives a higher reward than the previous state. Because q-learning does not rely on a transition state matrix or reward state matrix, it is a model-free learner. It is also considered an off-policy learner since at any given iteration, it does not have to observe an action on the policy it is working on, but it can take an action outside of the policy. Q-learning also utilizes gamma as a parameter to obtain the optimal policy. Q-learner also uses alpha which is the learning rate where an alpha of 0 means the learner will only look at the previous states while an alpha of 1 means it will put more weight on the current state. Additionally, epsilon is a parameter used to dictate whether the q-learner will be a greedy learner and take an action that it has already taken or if it will explore to some other random option. For q-learning, parameters such as alpha and epsilon may not always remain constant and can be applied as some sort of varying decay function. For the simplicity of seeing how different values might affect the convergence onto some policy, this paper will only work with constant alpha and epsilon values in evaluating q-learning.

Problem Space

To analyze the three different algorithms, the Forest Management MDP and the Frozen Lake MDP were used. These problems and algorithms were provided by the MDPToolbox and OpenAI Gym library. The forest management problem is defined by a forest with a certain age in the possible states, S. The goal of this problem is to maximize the profit from managing the forest. Trees can be allowed to grow longer before they are cut but waiting longer also introduces a larger possibility of a fire. After a fire, the state of the trees is set to 0 where the trees are too young to be cut. The Frozen Lake MDP is a grid world problem where an agent must navigate a frozen lake from a certain starting point to an ending point where the final reward for getting to the end is 1. There are safe parts of the lake and holes in the lake and falling into one of the holes ends with a reward of 0. With these two problems, this assignment will observe some differences and similarities between the different RL algorithms and their approach to finding the optimal policy.

Forest Management

The forest management problem was observed using two different state sizes to simulate a small state (S = 200) and a large state (S = 2000). The reward for waiting for the oldest state and cutting is 2 and the reward for waiting for the oldest state and waiting is 200. These r1 and r2 values were used to demonstrate how the change in problem size might affect the algorithms. The probability of a fire was kept constant at 0.1. For actions, 0 specifies a wait and 1 specifies a cut.

1. Value Iteration

The charts below show some metrics from running Value Iteration to find the optimal policy on the forest MDP problem. Different values of gamma (the discount rate) were tested with value iteration to observe what the total reward and total iterations would be at convergence. These tests were run where the states of the forest were 200 or 2000. For the problem size experiments, a discount rate of 0.99 was used.

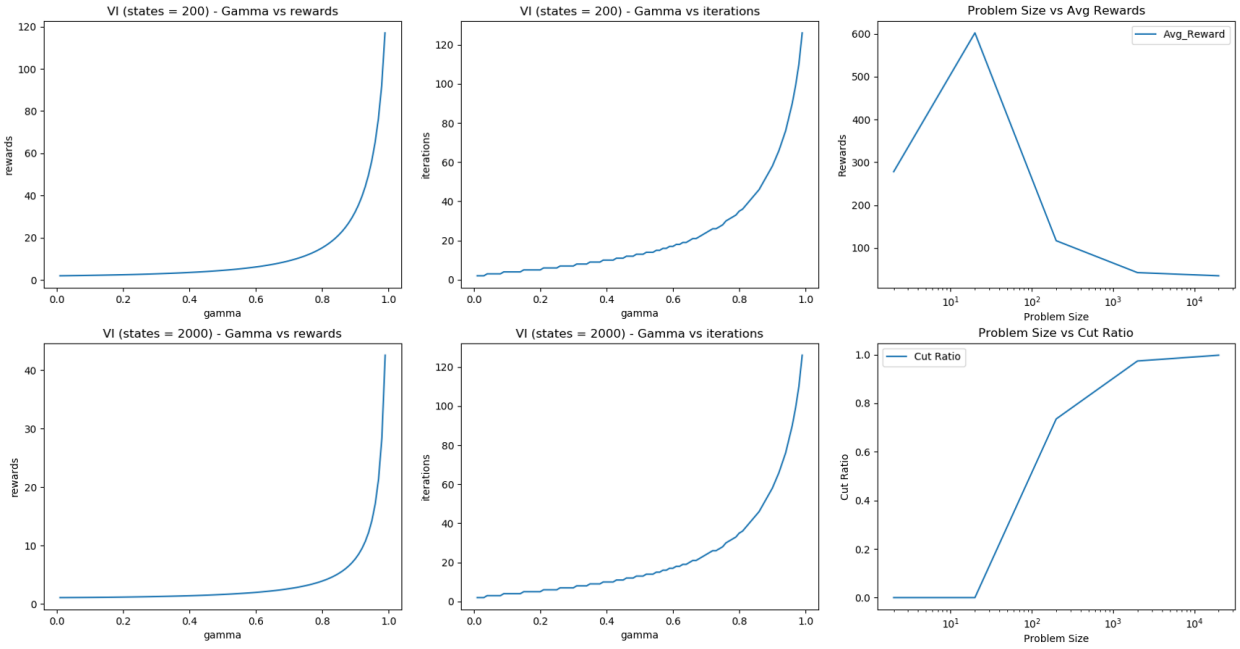

Figure 1: Value Iteration results with different states (200, 2000).

With value iteration, there are no notable differences in the shape of the curves shown above when comparing the larger number of states. The general trend of the curve is that a higher gamma value provides higher return on reward for this MDP. For the forest management problem, future gains and rewards are prioritized because the higher gamma value achieves the highest reward. This makes sense because the reward for waiting is much greater than the reward for cutting when the forest has hit the right state. If there were rewards available for cutting the forest at different states, the curve might look different since there is a reason to value current rewards. Value Iteration also takes around 130 iterations to converge on the optimal policy and this seems to be consistent in the increase in problem size. The difference is that with the optimal policy for 200 states, the maximum reward comes out to 117 and the maximum reward for 2000 states came out to 42.56. The figures at the end of figure 1 show that as the problem size increases, the reward decreases with an increase around a problem size of 10^1. This means that due to the probability of a fire, the learner will prefer to cut the tree when possible, taking the smaller reward. When the states decrease, there is less of a waiting period needed so the probability of the forest being reset by a fire is less. This is shown by the cut ratio shown in the figure above. As the problem size increases, the number of cuts taken by the optimal policy increases as well.

2. Policy Iteration

Results from experimenting with gamma and problem sizes are shown below in figure 2. Like value iteration, the shape of the curves is similar as the problem size increases. Increasing the problem size will decrease the potential reward gained and the optimal policy will give more action to cut the forest rather than waiting due to the risk of fire. PI also produces an optimal policy as gamma approaches 1. At 200 states, the optimal policy from PI at a gamma of 0.99 gives a reward of 130.34 and at 2000 states outputs a reward of 55.92. This is because the policy at 2000 states gives more cut actions achieving a smaller reward. It's important to note that policy iteration only takes around 50 iterations to converge on the optimal policy. Here the reward also decreases as the problem size increases and the popular action taken by the policy is to cut the forest each time that it is ready.

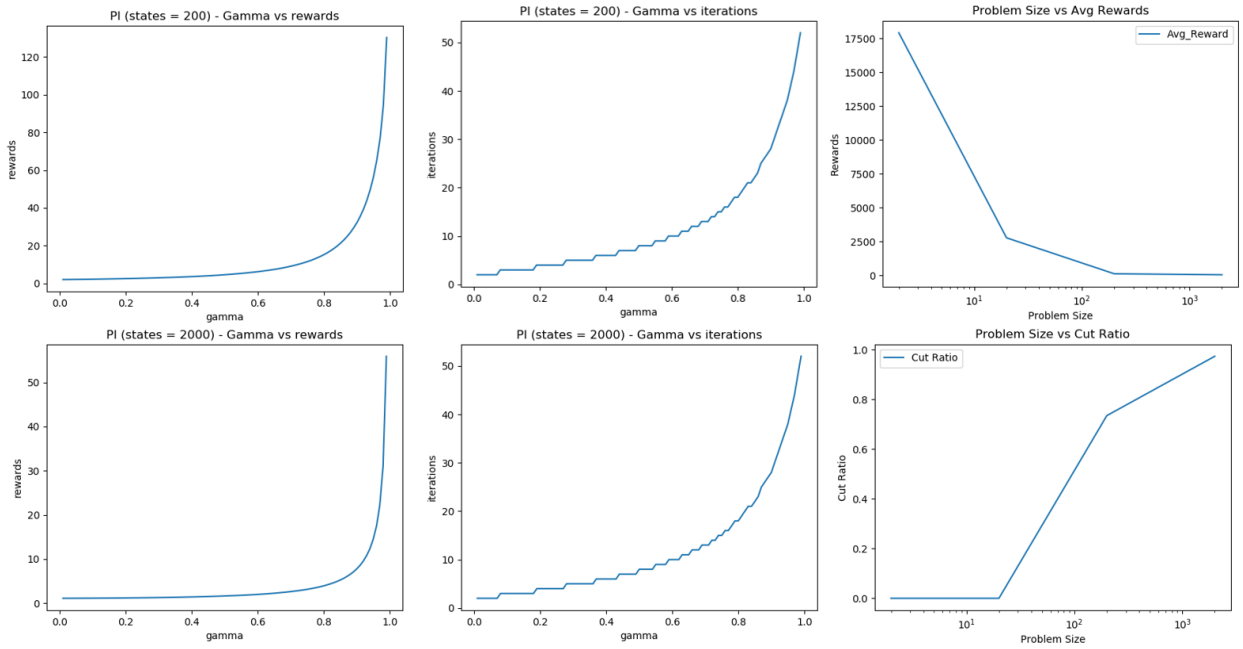

Figure 2: Policy Iteration results with different states (200, 2000).
Reward vs Problem Size and Cut Ratio vs Problem Size

Compared to value iteration, policy iteration converges in less iterations. In the previous figures, VI took about 130 iterations while PI only took around 55 iterations to converge on the optimal policy. This is because VI is a more computationally heavy algorithm since it must look through all actions to find the optimal one in updating the value function at each iteration. The policy iteration algorithm does not need to look through every action possible but will perform one look ahead to a new action and see if that improves the policy and its value function. If there is improvement, then there will be a change to the policy.

3. Q-Learning

Q-Learning produces a low reward amount as gamma increases achieving 0.19 reward for 200 states and 0.0058 for 2000 states. The general trend is that as gamma approaches 1, the reward increases, but the highest reward is achieved around a gamma = 0.6 for 200 states and gamma = 0.8 for 2000 states. Plotting alpha vs reward also produces a similar trend where alpha increasing tends to improve the reward achieved. Alpha dictates whether the learner will prioritize prior states and knowledge over the most recent or current states. From the charts above, an alpha value of around 0.5 provides the optimal reward for 200 states and alpha of 0.9 provides optimal rewards when the space increases.

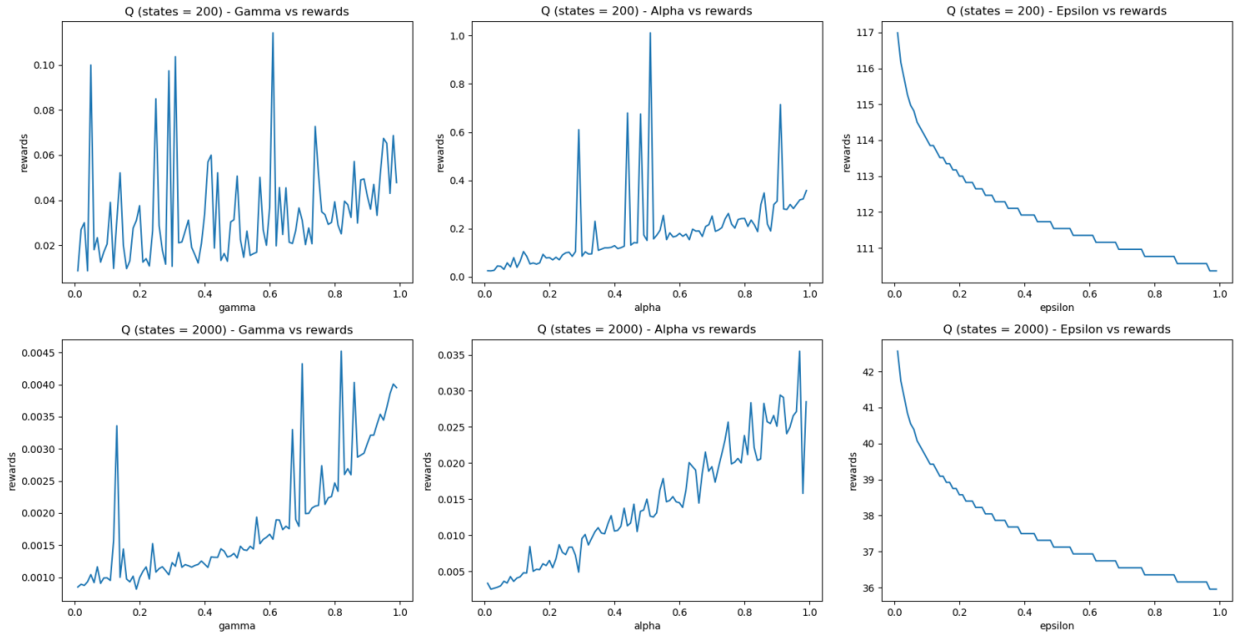

Figure 3: Q-Learning results with different states (200, 2000).

In terms of epsilon, a smaller epsilon value gives the best reward in both cases. A smaller epsilon value means the q-learner is greedier and will lead to exploitation tendency where it will choose the action with a high Q-value. This greedy behavior will lead to the learner to pick actions from ones it has already attempted rather than exploring other possible random actions.

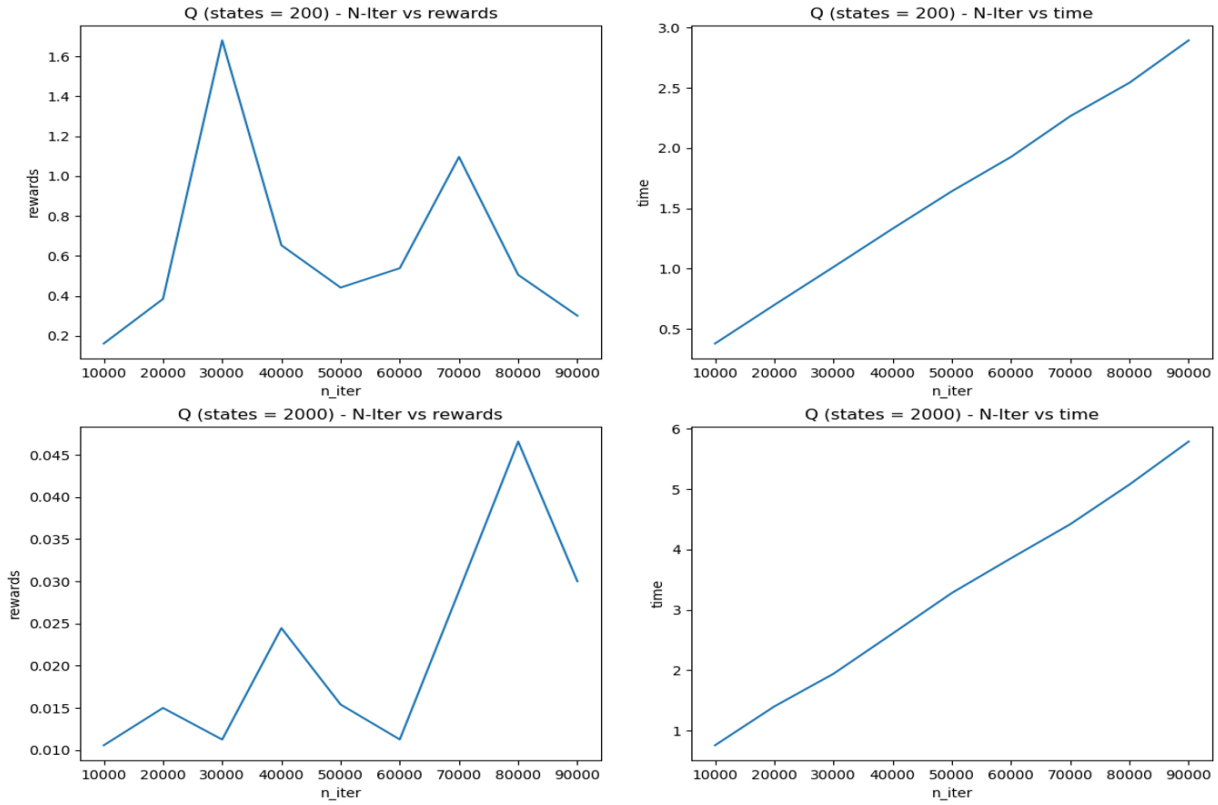

Figure 4: Number of Iterations vs Rewards and Wall Time

For a small number of states, it seems like a smaller number of iterations produces the most reward and for a larger amount of states a larger number of iterations are required to converge on some policy. The difference in iterations makes sense because there are more states and actions that the learner needs to choose from. This shows the slight randomness in Q-learner as well since it does not always converge on the same policy after a certain number of iterations. This is likely due to effect of epsilon above where there is some probability that the agent takes a random action rather than a previous action.

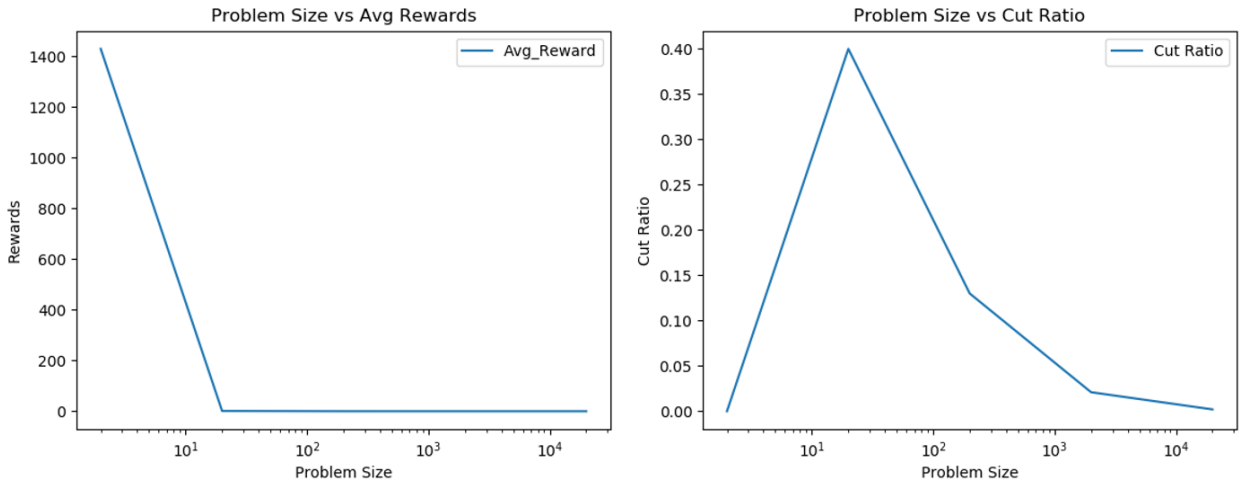

Figure 5: Reward vs Problem Size and Cut Ratio vs Problem Size

Interestingly, while the reward decreases as the problem size increases, the number of times the policy says to cut also seems to decrease as shown in figure 4 above. An explanation for this behavior might be that q-learner has not found an optimal policy for those larger states so all that it does is wait without considering the effects of a fire occurring if it always chooses wait. If the learner was able to find the optimal policy at each of the states, the curve would likely look like that of VI and PI where more cuts are done.

4. Comparison

Q-Learner performs poorly in this environment compared to algorithms like VI and PI. Q-learning does not seem to converge on the optimal policy in the given range of hyperparameters tuned. The scale of the rewards attained by the converged policy is widely different for VI, PI and Q-learning as VI and PI obtain rewards 2 orders of magnitude higher than q-learning in both sizes of the problem. This could be because Q-Learning is a model free algorithm, and it does not utilize a transition state matrix like VI and PI. Rather, it will take actions within the environment it is in and learn from the reward or consequence it receives from that action. Depending on values such as epsilon, alpha and gamma it will iteratively choose the next best action to take whether be an exploration step or an exploit step. With this forest problem, the chance of the fire likely makes it hard to accurately learn from the previous steps it has taken since even with a specific action, there is still a chance of a fire. Additionally, it seems like keeping some hyperparameters such as alpha and epsilon constant are not optimal for creating the best learner in q-learning. Upon printing the optimal policies from VI and PI, they came out to be the same policies and this shows in the figures

above as they both generally output the same reward. The main takeaway however, is that PI takes less iterations to converge than VI as expected since PI does not have to evaluate every action repeatedly. PI is the more optimal learner in learning the optimal policy as it achieves the highest reward in less iterations than both VI and Q.

Frozen Lake

To experiment with the Frozen Lake problem and the algorithms, a 4 x 4 frozen lake grid was used using a combination of the MDPToolbox and OpenAI gym libraries. The figure below shows the example 4 x 4 grid used in experimenting with the different algorithms. When the goal is reached, a reward of 1 is attained.

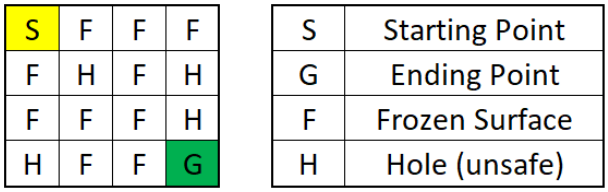

Figure 6: 4 x 4 Frozen Lake Rendering

The algorithms below were used to find the optimal policy and then subject to string of episodes to observe how the optimal policy would perform if it was run for 10000 episodes. The optimal policies were extracted from each of the learners based on experiments with values for gamma, alpha and epsilon where they applied.

1. **Value Iteration**

The figures below show that value iteration took about 300 iterations to converge on the optimal policy for a gamma value of 0.99. The associated reward for this policy came out to 0.4 and this shows a similar trend to the previous problem where valuing the future reward more (gamma value closer to 1) will return a higher reward in the end. This makes sense since the agent can only gain a reward for getting to the end of the goal without falling into the hole. Value Iteration is more consistent in producing some optimal policy because there are no peaks or oscillations in the curve like there are for policy iteration in figure 8 below. This is likely because value iteration does not extract the optimal policy until the optimal value function is converged on. Because of this, the number of iterations it takes to converge in VI is significantly greater than PI as VI takes around 300 iterations at a gamma of 0.99.

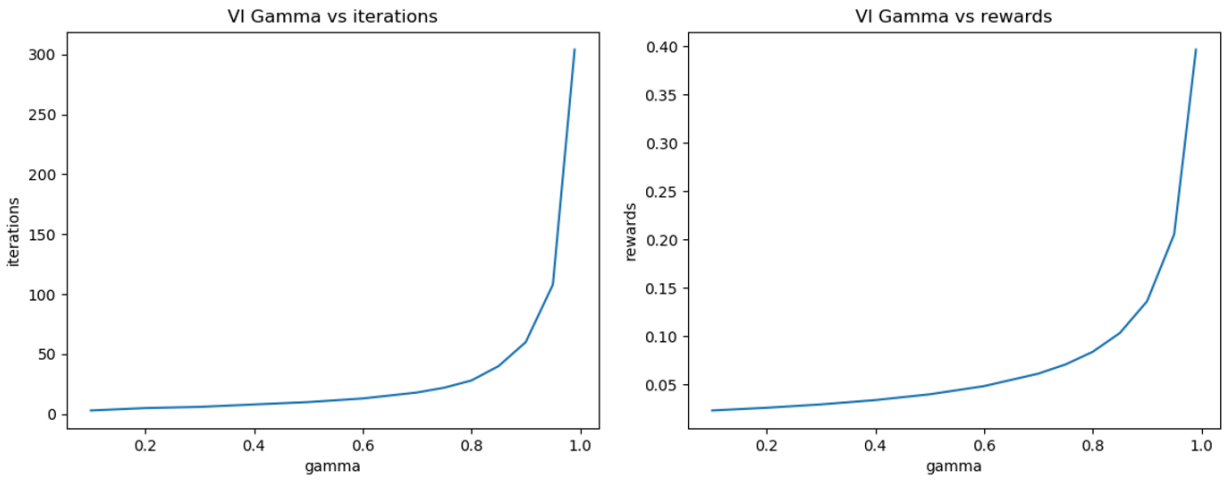

Figure 7: Iterations and Rewards per different Gamma Values

2. Policy Iteration

Policy Iteration also took less iterations to converge on the optimal policy shown in the figure below. There seems to be a slight peak in iterations around a gamma value of 0.85 but later when gamma is around 0.9 and higher, the number of iterations it takes to get to converge is significantly less than that of VI in the previous figure. There is a slight spike in iterations around a gamma value of 0.85 for PI. This could be because of the randomness of PI as it starts with some random policy and will iterate until it converges on the optimal policy. It seems at that gamma value, the agent started at a very suboptimal policy and had to iterate a lot to get to the optimal policy. The optimal value for gamma is shown to be 0.99 and at this discount value it takes 7 iterations for PI to converge on the optimal policy.

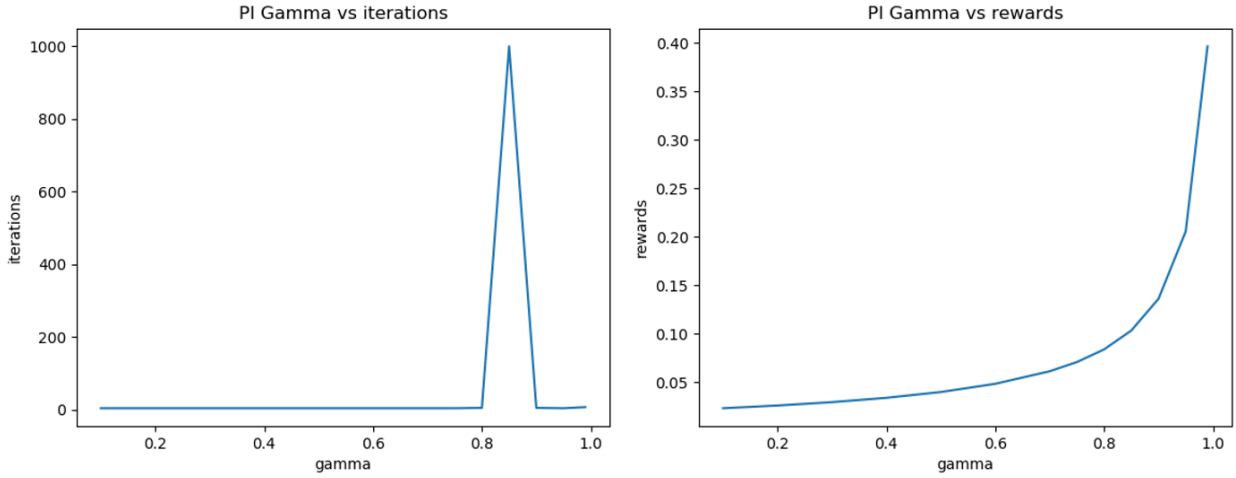

Figure 8: Iterations and Rewards per different Gamma Values

3. Q-Learning

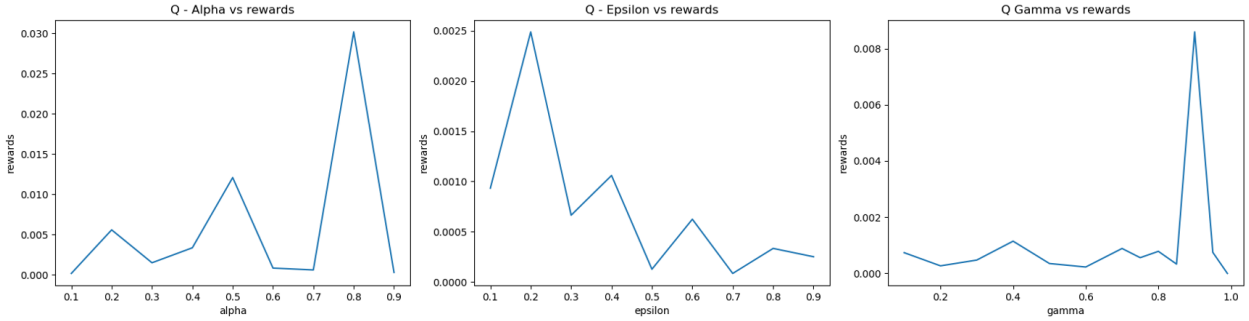

Figure 9: Rewards per epsilon, alpha and gamma

The figures above show that an optimal alpha value is when the alpha value is high around 0.8. This means the q-learner can learn a better policy by prioritizing current information gained rather than the previous q-value gained from an action taken. A lower epsilon value also gives a higher reward showing that a greedier algorithm may produce the more optimal policy. The chart above shows a generally decreasing trend in rewards versus an increase in epsilon meaning that q-learner will obtain better results given a greedy exploitation strategy rather than an exploration strategy. The peak seems to be 0.2 for epsilon so there is still some possibility that the learner may take a random action rather than taking an action that it has already taken so far. Like the behavior of the other learners, the highest reward occurs when the learner prioritizes the future rewards over the current rewards since a higher gamma value around 0.9 seems to produce an optimal policy.

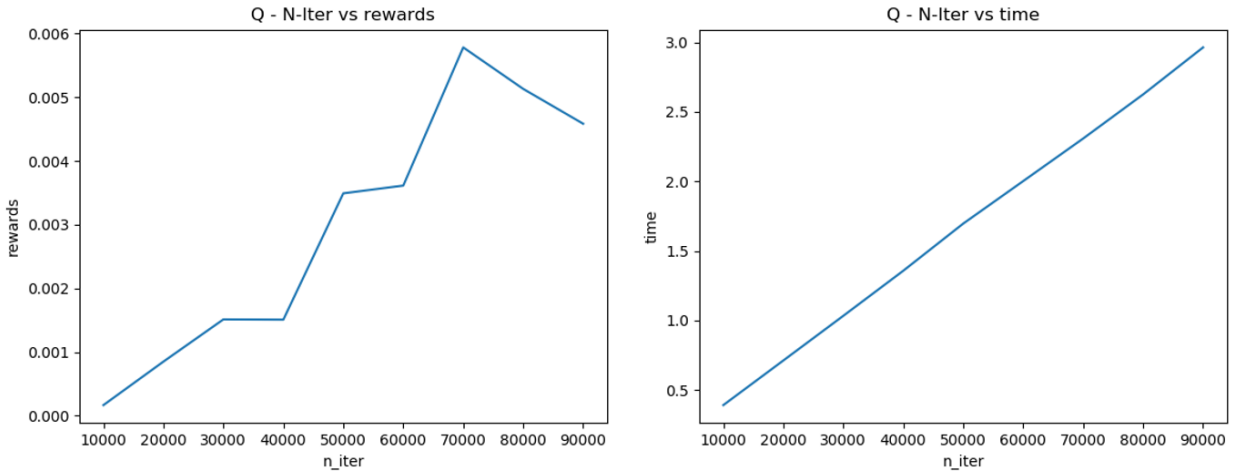

Figure 10: Number of Iterations vs Rewards and Wall Time

Figure 9 above also shows the relation between the number of iterations and rewards and wall time it takes for q-learner to find a good policy. Q-learning seems to take many iterations to converge on an actual policy. The MDPToolbox requires n-iter to be at least 10000 iterations.

4. Comparison

	Total Reward	Avg Number of Steps	Win %	Wall Time (s)
Q-Learning	0.033	16.23	3.51%	1.7
Value Iteration	0.52	38.42	74.15%	0.0069
Policy Iteration	0.52	38.4	74.90%	0.002

Table 1: Performance of the best policy from each learner

The table above shows the performance of the optimal policies extracted from each of the learning algorithms. The total reward, win % (number of times the problem was solved), steps taken, and wall time are shown over 10000 episodes. Policy Iteration and Value Iteration are the superior techniques in finding the optimal policy as they continuously converge on the same optimal policy and can solve the grid problem 75% of the time. Compared to Q-learning, they also take much less time as shown in the wall time measured above. The results above further show how VI and PI will converge on the optimal policy while Q-learning does not seem to do so. Q-learning takes many iterations and time to converge on a policy but it is suboptimal compared to VI and PI. Not shown above, but from running the q-learner algorithm multiple times, there was a large swing in performance likely due to the randomness in the algorithm as it uses epsilon to determine whether it will explore a new action or take a previously known action. In figure 9 above, q-learning finds the optimal rewards at 70000 iterations which is way more than the amount of iterations it took VI or PI to find an even more optimal policy.

Conclusion

The Forest Management Problem and the Frozen Lake problem help explore a new area of learning where labeled data or structured continuous and categorical data are not required. Reinforcement Learning offers a new way to interact by observing how the actions of an agent change the state of the environment and affect the agent. Three different algorithms were explored across these two problems to understand their behavior as they converge on the optimal policy to maximize the reward from the environment they are in. Generally, VI and PI will behave similarly often converging on the same optimal policy and producing the maximum reward. PI takes less iterations than VI and is observed to be the least computationally expensive algorithm. Q-learning was the far inferior algorithm in these problems often not converging on the same optimal policy and unable to produce the same rewards. In addition, q-learning tends to be more computationally expensive requiring many iterations to converge upon some policy.