CS 7641 Assignment 4

Chris Lundberg

11/27/2021

Introduction

In this paper, two Markov Decision Processes (MDPs) will be explored and solved with the Value Iteration and Policy Iteration algorithms as well as with Q-Learning, a model-free Reinforcement Learning algorithm. The performance of the three algorithms will be compared, with a particular focus on how the algorithms scale as the problem size increases and how Q-Learning compares to the two algorithms that have access to the transition and reward functions.

MDP Descriptions

The two MDPs that will be used in the experiments that follow are:

- Open AI Gym's FrozenLake-v0 environment
- The Forest environment in `pymdptoolbox`.

These two problem settings have a number of interesting properties that should help to highlight the differences between the algorithms under consideration.

Frozen Lake

The `FrozenLake-v0` environment is a so-called grid world problem, where the possible states are positions on a 4x4 grid. The environment is supposed to represent a frozen lake, and the goal is to get from the starting state to the goal state, where the frisbee is located. There are four tile types:

- **S**: The starting point. This state is safe.
- **F**: Frozen. This state is safe.
- **H**: Hole. This state is not safe.
- **G**: Goal. This is where the frisbee is located.

Each of the first three tile types gives a reward of zero and the goal tile gives a reward of one. The episode ends when the agent lands on either the goal or a hole.

With only 16 possible states, `FrozenLake-v0` is a relatively "small" problem, which means that visiting all of the states a large number of times won't be particularly computationally intensive. This means that, while some algorithms may still require more iterations to converge, the clock time required to converge is unlikely to be particularly onerous.

One of the most interesting features of this environment is that the agent's actions are stochastic, meaning that the direction that the agent moves may not be the direction the agent chose to move. In other words, the agent will move in the chosen direction with probability p and in some other direction with probability $1 - p$.

Forest

The `Forest` environment is based on a forest management scenario where the state s is the age (in years) of the forest and the number of possible states S is equal to the maximum age of the forest.

The possible actions are:

- **0**: Wait. Maintain the old forest for wildlife.
- **1**: Cut. Make money by selling cut wood.

If the agent chooses "wait", it receives a reward of zero in all states except the final state (where the forest reaches its maximum age), in which case it receives a reward of four. If the agent chooses "cut", it receives a reward of one in all states except the final state, where it receives a reward of two.

While the actions are deterministic in this environment, the environment is stochastic. Each year, a forest fire may burn the forest down with probability p, so if the agent chooses to wait, the forest will survive the next year with probability $1 - p$.

With S possible states, this problem can be made arbitrarily "large" or "small". In order to compare our results using each of the three algorithms in question to the results obtained on the `FrozenLake-v0` environment, we will use 1,000 states for most of our experiments. Given the large number of states, it is likely that Value Iteration will take a long time (in number of iterations and clock time) to converge, while Policy Iteration may converge more quickly. In this context, the Q-Learning hyperparameters α (learning rate) and γ (discount factor) are likely to play a large role in both how long it takes the algorithm to converge and the nature of the learned policy.

Experiments

In the following experiments, the performance of both model-based approaches will be compared to each other and the performance of Q-Learning will be compared to that of the model-based approaches.

In both problem settings, the convergence criteria for the model-based approaches were:

- **Value Iteration**: the algorithm was stopped when the maximum change in the value function $V(s)$ from one iteration to the next was less than a threshold $\epsilon = 0.01$. This choice of ϵ appeared to provide the best trade off between performance and number of iterations, as seen in figure 1.
- **Policy Iteration**: the algorithm was stopped when there were no changes in the policy from one iteration to the next.

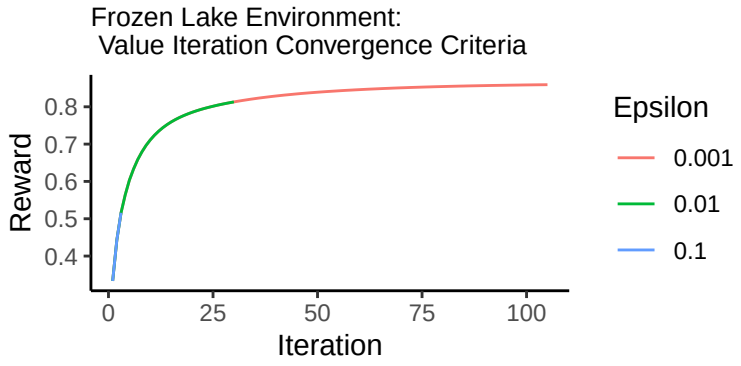

Figure 1: Value Iteration convergence on the Frozen Lake environment. Gamma was set to 0.99 for each epsilon value.

Value Iteration vs. Policy Iteration

On the `FrozenLake-v0` environment, Policy Iteration converges in fewer iterations than Value Iteration (see the table below). This is true across all three γ values, and the difference between algorithms becomes more pronounced as γ increases. This is because larger values of γ place more importance on rewards further into the future, which slows convergence for Value Iteration but has relatively little impact on Policy Iteration.

Table 1: Iterations until convergence on the Frozen Lake environment

Gamma	Policy Iteration	Value Iteration
Gamma = 0.9	5	10
Gamma = 0.95	5	14
Gamma = 0.99	6	30

Policy Iteration is able to converge in fewer iterations than Value Iteration because while Value Iteration only performs a single loop to directly optimize the value function $V(s)$ and Policy Iteration alternates between policy evaluation and policy improvement, there are a finite number of deterministic policies (vs. an infinite number of potential value functions). Since there are an infinite number of value functions that are compatible with the optimal policy, directly optimizing the policy requires fewer iterations.

The learning curves for each algorithm are shown in figure 2. Clearly, Policy Iteration requires a similar number of iterations across γ values, while Value Iteration requires significantly more iterations as γ increases.

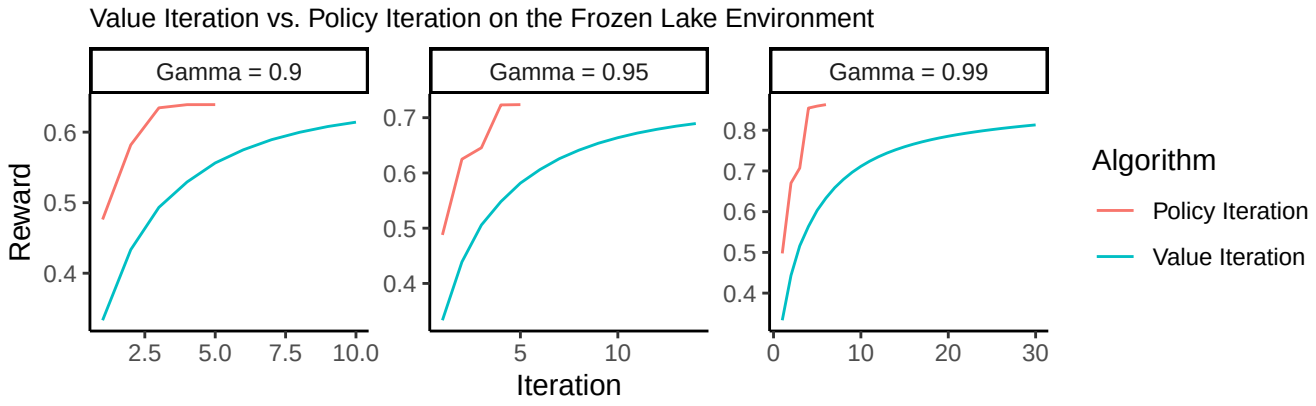

Figure 2: Value Iteration and Policy Iteration learning curves on the Frozen Lake environment.

Upon inspection, when $\gamma = 0.99$, the policies learned by the two algorithms differ in only one of the sixteen states. In the first row and third column of the 4x4 grid world, the policy learned by Policy Iteration returns the action "up", while the policy learned by Value Iteration returns the action "left".

When we apply Policy Iteration and Value Iteration to the `Forest` environment, we observe very similar performance. The table below displays the performance of each algorithm using three different values for γ. In all cases, $p = 0.1$ and the environment contains 1,000 possible states.

We see that as γ increases, both algorithms require more iterations to converge, but Value Iteration suffers more than Policy Iteration. This is likely because in this problem setting the reward for waiting and never cutting is the highest possible reward, so the less this reward is discounted, the harder Value Iteration has to work to propagate this information backwards to previous states.

Table 2: Iterations until convergence on the Forest environment with 1,000 states

Gamma	Policy Iteration	Value Iteration
Gamma = 0.9	10	39
Gamma = 0.95	13	58
Gamma = 0.99	18	92

The learning curves for the above experiment are shown in figure 3. Value Iteration requires more iterations to converge as γ increases, and this increase is much more rapid than that of Policy Iteration, but for problems with 1,000, 2,000, and 10,000 states, the policies learned by the two algorithms are the same.

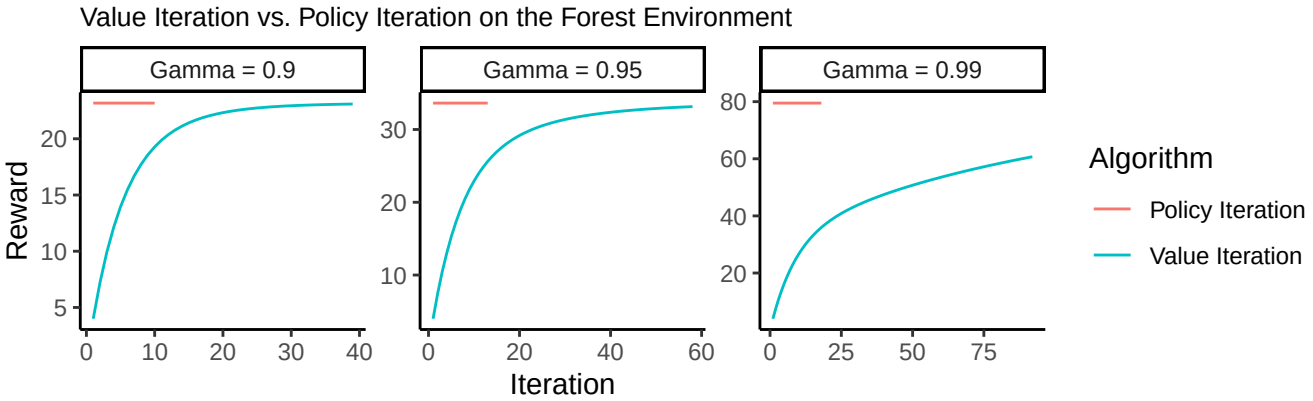

Figure 3: Value Iteration and Policy Iteration learning curves on the Forest environment.

How do Policy Iteration and Value Iteration perform as the problem size increases? The `Forest` environment can be made arbitrarily "large" by changing the maximum potential age of the forest, enabling us to investigate how each algorithm performs as the problem size increases. Interestingly, the number of iterations required for convergence does not change as the number of states increases! The table below shows the number of iterations for each algorithm across various problem sizes. In all cases, $p = 0.1$ and $\gamma = 0.99$.

Table 3: Iterations until convergence on the Forest environment

Size	Policy Iteration	Value Iteration
20 States	18	92
200 States	18	92
1000 States	18	92
2000 States	18	92
5000 States	18	92
10000 States	18	92

But what about the runtime of each algorithm as the problem size scales? Figure 4 is somewhat surprising given that both algorithms converge after the same number of iterations regardless of problem size, as noted above.

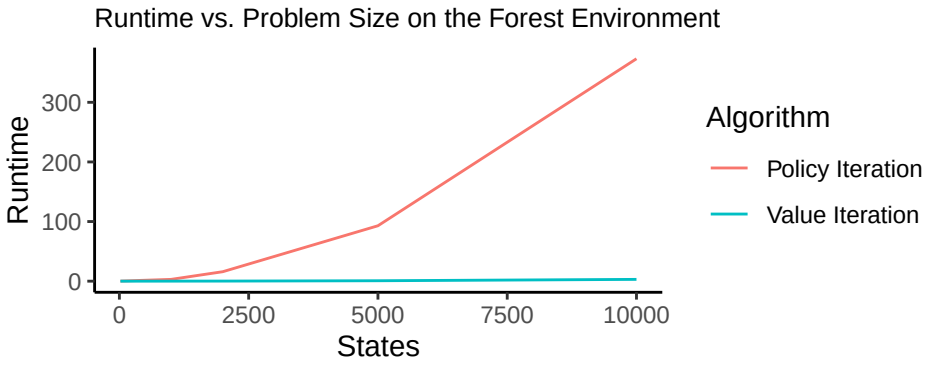

Figure 4: Value Iteration and Policy Iteration runtime vs. problem size on the Forest environment.

What's going on here? The time complexity of a single iteration of Value Iteration is $O(|S|^2|A|)$, where $|S|$ is the number of possible states and $|A|$ is the number of possible actions. In contrast, the time complexity of a single iteration of Policy Iteration is $O(|S|^3 + |S|^2|A|)$. So, while Policy Iteration requires only a few iterations and Value Iteration requires many iterations, the time complexity of Policy Iteration is dominated by the term $|S|^3$. Therefore, as the number of states increases, the runtime of Policy Iteration will increase much faster than that of Value Iteration.

Q-Learning vs. Model-based Algorithms

Now that we have a sense of the strengths and weaknesses of each of the model-based approaches discussed above, it is possible to compare them to the performance of Q-Learning, a model-free reinforcement learning algorithm. How will Q-Learning perform relative to algorithms that have access to the transition and reward functions of the MDP?

First, we'll experiment with different Q-Learning hyperparameters on the `FrozenLake-v0` environment.

In figure 5 the performance of various α decay parameters is depicted. These plots can be thought of as learning curves, as they depict the mean and maximum values of $\hat{V}$, the estimated value function, at each iteration. From these plots, we see that $\alpha = 0.99$ is results in the fastest convergence of the three parameter values considered. This is likely because a slower decay schedule allows the algorithm to make larger updates to the estimated value function early on.

This may be be largely a function of this specific problem's reward structure, where only one of the states has a non-zero reward.

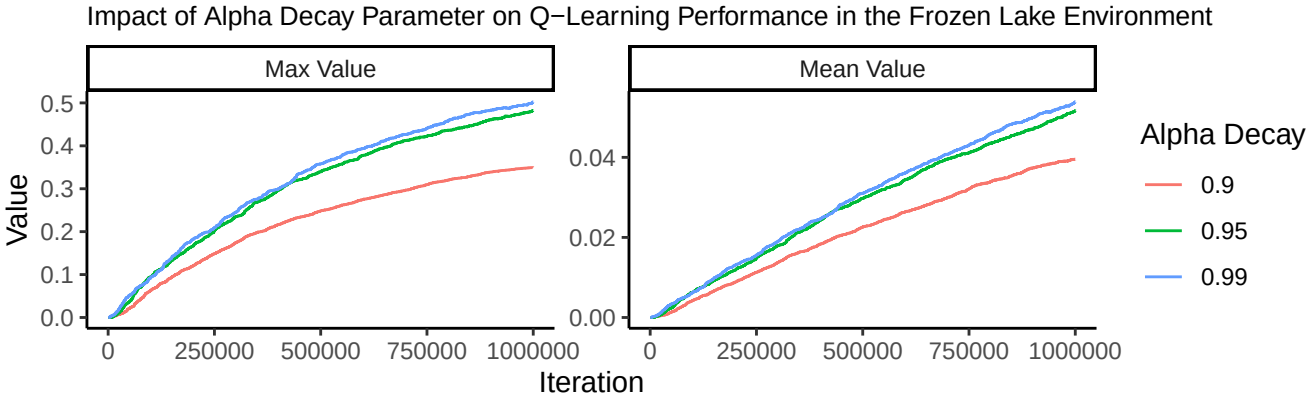

Figure 5: Learning curves using various alpha decay parameters in the Frozen Lake environment.

Figure 6 shows the impact that various ϵ values have on the algorithm's performance. While the α decay controls the rate of learning, ϵ controls the tradeoff between exploration and exploitation. Specifically, at each time step, the Q-Learning agent chooses a random action with probability ϵ and a greedy action with probability $1 - \epsilon$.

While the choice of ϵ doesn't appear to have the same impact that the choice of α decay does in this problem setting, it appears that a starting value of $\epsilon = 0.25$ leads to the best performance.

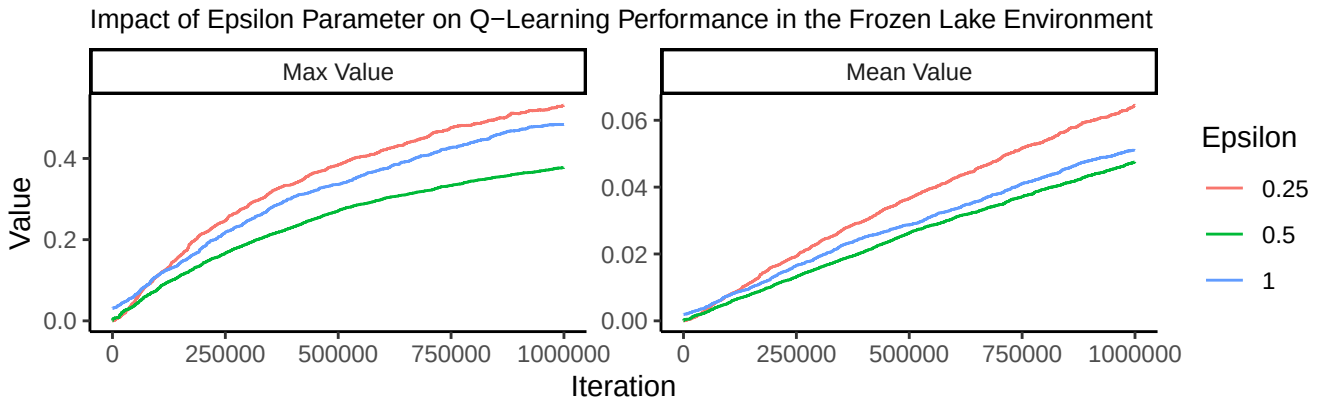

Figure 6: Learning curves using various epsilon parameters in the Frozen Lake environment.

How does Q-Learning compare to the performance of the two model-based algorithms discussed earlier on the `FrozenLake-v0` environment? After training the Q-Learning agent for three million iterations with $\gamma = 0.99$, $\alpha_{decay} = 0.99$, and $\epsilon_{starting} = 1.0$, the policy learned by the agent differs in three states from the Value Iteration policy and by four states from the Policy Iteration policy.

Given that the Q-Learning agent doesn't have access to the transition and reward functions of the underlying MDP, this is no small feat. It seems likely that given more training iterations and more hyperparameter tuning, Q-Learning could achieve performance equal to that of a model-based approach.

Q-Learning achieved reasonable performance on a grid world problem with a small number of states, but how does it perform on larger non-grid world problems? Learning curves for various values of ϵ are shown in figure 7. It appears that initial ϵ values of 0.5 or greater lead to quicker convergence and better performance in this context.

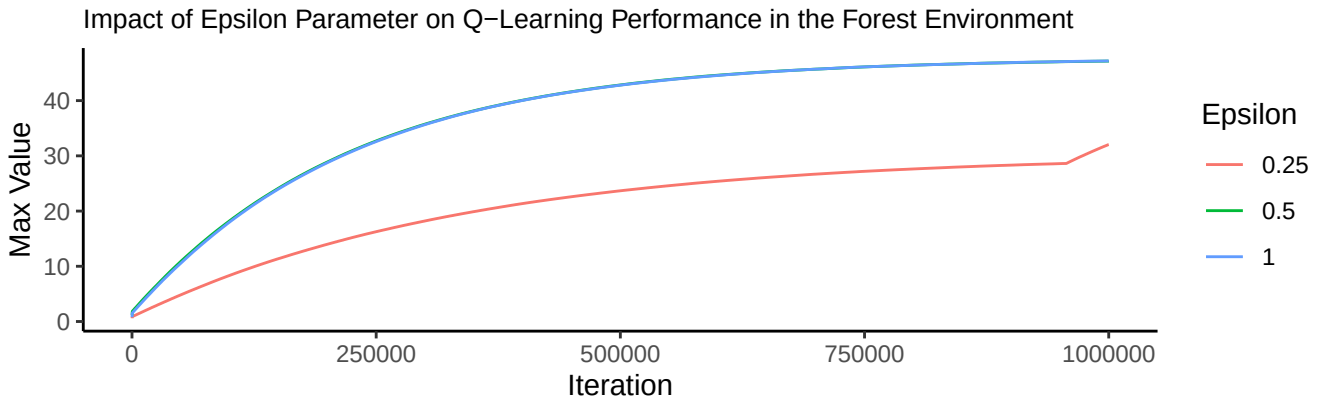

Figure 7: Learning curves using various epsilon parameters in the Forest environment.

This makes sense, as the problem contained one thousand states in these experiments. Since the largest rewards occur at the end of this sequence of states, more exploration early on is likely to lead the Q-Learning agent to discover the potential value of waiting in states near the end of the sequence.

What about the runtime of Q-Learning as the problem size increases? Figure 8 plots the runtime vs. the number of iterations for various problem sizes for the `Forest` environment. As expected, for a given number of iterations, problems with more states take longer to run. It appears that runtime increases linearly with problem size.

Note that Q-Learning is able to complete one million episodes in half the time it takes Policy Iteration to converge on the `Forest` environment with 10,000 states.

How does Q-Learning compare to the performance of the model-based algorithms on this task? Recall that on a problem with one thousand states, Policy Iteration and Value Iteration return identical policies, both of which choose "cut" in 981 of the states. In contrast, after five million episodes, the policy returned by Q-Learning chooses "cut" in 642 of the states.

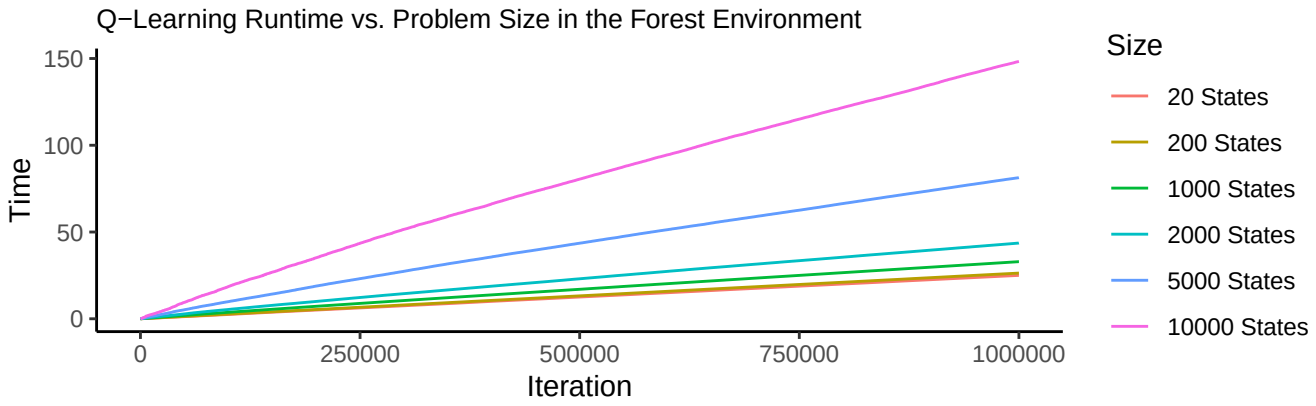

Figure 8: Q-Learning runtime vs. problem size in the Forest environment.

While that means that the policy returned by Q-Learning is quite far from the policies returned by the other two algorithms, running the Q-Learning algorithm for more episodes and performing more hyperparameter tuning would likely allow a Q-Learning agent to learn a policy that is competitive with the model-based policies.

Conclusion

In this paper, both model-based and model-free algorithms for learning optimal policies for Markov Decision Processes were explored and compared. Important strengths and weaknesses of each algorithm were discussed and directions for further work in a model-free setting were suggested.