Markov Decision Processes

Spring 2021 CS7641 Assignment 4

Hung-Hsi Lin hlin83@gatech.edu

ABSTRCT

This paper aims on exploring value iteration, policy iteration and one reinforcement learning algorithm, Q-learning to solve two Markov Decision Processes (MDP), frozen lake and forest management problems. The performances metrics of each algorithm will be compared and analyzed on different type and size of MDP problems.

INTRODUCTION

This paper will first introduce the theory of three algorithms, value iteration, policy iteration and Q-learning, also the settings of two Markov Decision Processes (MDP) problems. Then the application of three algorithms is implemented in small and large size of frozen lake and forest management problems, respectively. The performance of each algorithm, including convergence rates, rewards, runtime and influence of settings will be analyzed.

Value Iteration

Value iteration use Bellman equation to iteratively update expected value functions in all steps until optimum values are reached. The iteration terminates while the values between new state and old state is smaller than a threshold. This algorithm is guaranteed to find and then converge to the optimal values. [1]

Policy Iteration

In policy iteration, first, initiated policies are randomly generated for each state, then the values are calculated with Bellman equation. [2] Different from value iteration, the "policy" instead of values for each state will be greedily updated while the value function could be improved by other policies. The policy iteration terminates while the policy values are converged.

Q-Learning

Q-Learning is a model-free method, it first creates a matrix called Q-table for each state and action to store the Q-value calculated from immediate and delayed rewards. [3] The agent has capability to either exploit from previous results or explore new actions to update the Q-values. Finally, the probability of exploration will gradually become negligible and the Q-values are mostly updated to optimum. The termination of the algorithm could be set a max iterations or threshold of improvement just like the other two policies.

Frozen Lake

Frozen lake is a grid and discrete finite MDP problem [4]. A square map is created first, and the starting point (S) locates at top-left corner. The motion of the robot includes up, down, right or left. The goal is to reach the corner of right-bottom (G) and get reward +1. Because we assume the agent is on a frozen lake, represented by the symbol F, the floor is slippery, therefore the movement accuracy will be only 33%, and the other 33% will be distributed to the left and right side of agent, respectively. Standing on the frozen floor will have 0 reward, however, there are holes in the gird (H) which will cause -1 reward and end the game. This problem is interesting because it is like the robot working in a farm or warehouse, which already happens in our daily life. And the motion might not be 100% accurate sometime due to slippery floor or unexpected obstacles. In real life, the size of the problem could be small or huge, therefore, the size I put in this paper includes 8x8 and 30x30 maps, so that we could tell how reinforcement learning algorithms works in different scale of problems. The possibility of frozen floor is set to 0.8, means there is 20% chance of creating holes in the map. This lake problem was created using gym module.

Forest Management

Different from the previous one, the forest management is a non-grid problem. [5] The goal for this agent is to maintain a wild forest and cut word for profits, this problem is interesting because it is the real challenge many countries are facing now and it would be great to have an optimum algorithm for action suggestion instead of executing randomly. For each step (year), the agent has two actions to select, **Cut** or **Wait**. As the trees reaches to the oldest states, the agent could have the reward with r1(4) or r2(2) with wait and cut action, respectively. However, there's a possibility (p = 0.1) that each year wildfire occurs that kills all tree. In conclusion, the agent should learn if he's willing to

take a risk or not. The size of states ranges from 4 and 1000 for studying the influence of problem size on the performance metrics in all different algorithms.

SECTION 1: FROZEN LAKE PROBLEM

SMALL SIZE - 8X8 GRID

At the beginning, we created the frozen lake grid with size of 8x8 using gym modules, with probability of frozen floor with 0.8. In value iteration, two important parameters have to be optimized first, discount (gamma) and epsilon. Discount means the discount factor on the future rewards, and the epsilon is the stopping criterion. In Figure 1, the influence of epsilon and discount on rewards and iterations required for convergence are summarized. In Figure 1.a to 1.c, the epsilon was set as 1e-2, 1e-6 and 1e-10 under variance of discount ranging from 0.1 to 1.0. The blue line indicates the mean rewards of 1000 runs (1 is maximum means reaching to goal), and the red line represents the mean iteration until convergence. We could clearly see if the epsilon is too close to zero, while it is too easy to terminate the iterations, the mean rewards is worse. From Figure 1.d to 1f, the discount of 0.5, 0.9 and 0.99 are set under variance epsilon. We could also see clearly that the discount should be large enough (0.99 in our case) to have the reward reaching to optimal maximum. As the discount is high enough, we will have more tolerance on selecting larger epsilon so that the convergence could happen earlier.

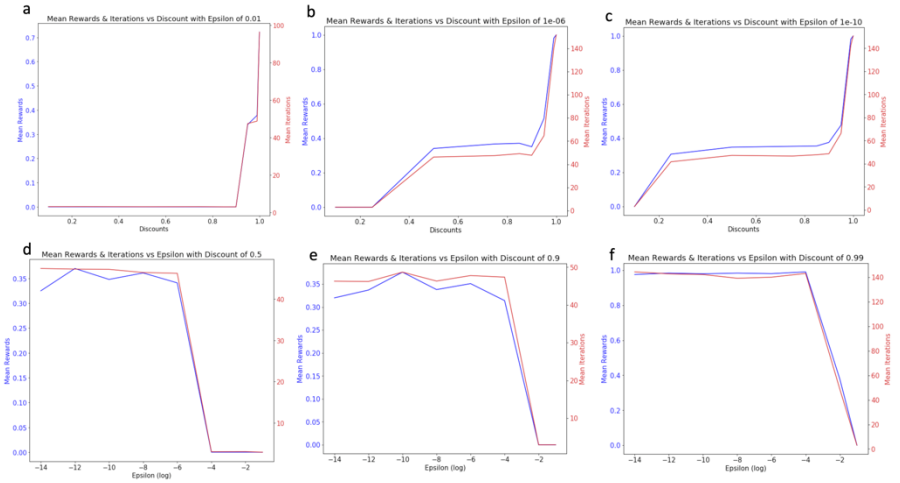

Figure 1 – Mean rewards (blue line) and iteration (red line) with different settings of discounts for (a) 0.01 (b) 1e-5 (c) 1e-10, and with different discount of (d) 0.5 (e) 0.9 and 0.99 with value iteration algorithm.

The same optimization works are also done for policy iteration algorithm, and the optimized discount and iteration are 0.99 and 1e-10, respectively. To compare the performance between value iteration and policy iteration, the 8x8 grid frozen lake problem was solved with epsilon of 1e-2, 1e-6 and 1e-10, and the discount is set as 0.99 for both cases. The convergence was determined while the difference of old/new value is lower than the set epsilon. Results are concluded in the Figure 2., while 2.a to 2.c show the rewards under each iteration using value iteration, and the figure 2.d to 2.f are the results ran by policy iteration. First, the reward from both algorithms reaches to optimum (i.e., 1.0) for the epsilon of 1e-10 and 1e-6, supporting the fact that both value and policy iteration should converge to optimum if the settings are proper. When the setting of epsilon is too large, the results converge too fast, neither value iteration nor policy iteration could output good rewards, and thes results are shown in Figure 2.c and 2.f. Another observation should be noted is that value iteration requires more iteration (> 400) than policy iteration (<10) until converges. Even the epsilon is small (1e-2) for value iteration, it still takes tens of iteration until converge, while policy iteration could converge in just few steps. In this frozen lake problem, policy iteration should update the action in each state to make sure maximum of value function is reachable. There are only 4 actions could be selected in this problem, therefore, not too much iteration is needed for policy iteration. On the other hand, in each round for value iteration, values for all states have to be optimized and updated, large number of iterations could be expected.

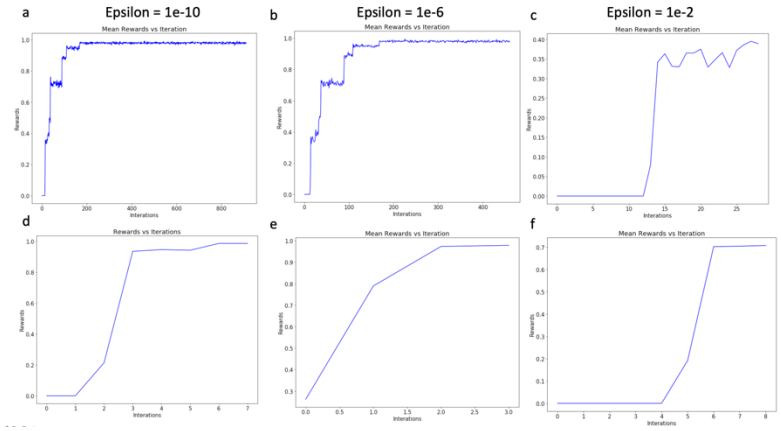

Figure 2 – Rewards vs Iterations in 8x8 grid using value iteration with epsilon of (a) 1e-10 (b) 1e-6 (c) 1e-2 and using policy iteration with epsilon of (d) 1e-10 (e) 1e-6 (f) 1e-2.

It would be interesting to see how the policies changes in each iteration. In Figure 3., the policies at different rounds are shown for value iteration in 3.a – 3.d and policy iteration in 3.e – 3.h. For The green block means frozen floor, purple region represents the hole and yellow block is the goal. For value iteration, the figure 3.a shows the initial states, while the policies are mostly incorrect. With iteration of 50 (3.b), some of policies are improved to optimum, and for the iteration of 180 (3.c), over 95% policies are already optimized, so most of them are similar to the iteration of 700, which is closed to converged reward higher than 0.98. This observation is supported by the results shown in Figure 2.a. For the policy iteration, we can see the policies change faster than the cases of value iteration in just few rounds. It makes sense because in policy iteration, the values in each state are calculated and only check if the policy could be further improved or not. Results from policy iteration converges in 6 iterations (3.h). We can see the converged results for value iteration (3.d) and policy iteration (3.h) are the same, proved that both algorithms work well in this 8x8 grid frozen lake problem.

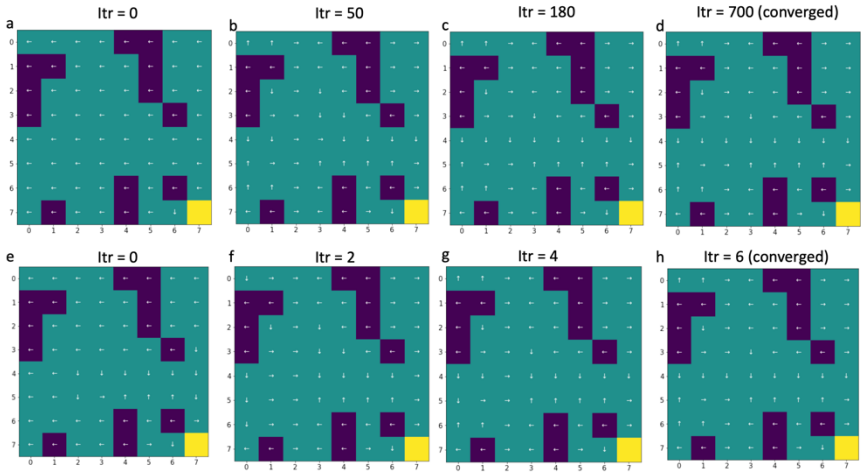

Figure 3 – Policies in 8x8 frozen lake problem using value iteration (a - d) and policy iteration (e - h) with different iterations (itr).

LARGE SIZE - 30X30 GRID

To see if these two algorithms work also well for larger size of the frozen lake problem, the size is then increased to 30x30. In Figure 4., we can see that the results obtained from value iteration and policy iteration are close (~0.89 for VI and ~0.87 for PI), and we can see the value iteration suffers as the size of states is large because the algorithm has to visit every state for value calculation and optimization. The iteration for VI is 3400 and 5400 with epsilon of 1e-6 and 1e-

10. And for policy iteration, the advantage is that it only searches in a finite space, not possible infinite space like in value iteration case, so it is expected to converge in less iteration (< 50) and faster.

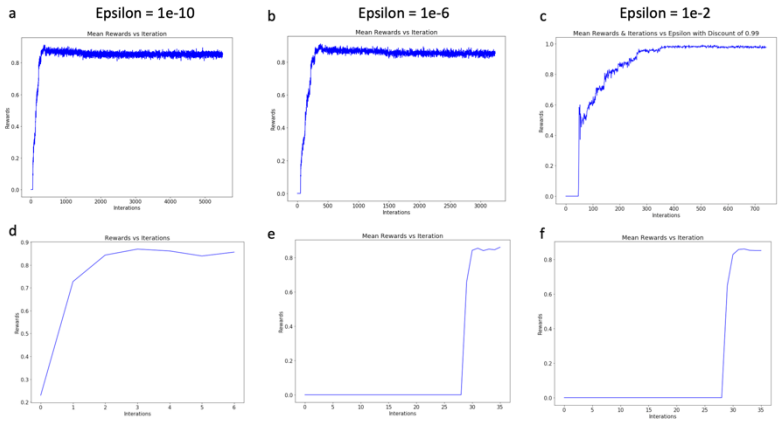

Figure 4 – Rewards vs Iterations in 30x30 grid using value iteration with epsilon of (a) 1e-10 (b) 1e-6 (c) 1e-2 and using policy iteration with epsilon of (d) 1e-10 (e) 1e-6 (f) 1e-2.

To study how much time and iterations required for converging to the optimum rewards, the time and iteration are concluded in Figure 5. Figure 5.a shows the runtime for VI and PI in the grid size of 8x8 and 30x30, respectively. As we could see, value policy takes more time to converge in both grid size cases compared with policy iterations. And in Figure 5.b, we can also see that the required iterations for value iteration is way larger than the policy iteration. The required iterations also scaled exponentially as the size of the problem became larger. In conclusion, the VI and PI could both converge to the policies that provide the optimum reward. In both case, epsilon should be small enough so that the algorithm won't converge too early before reaching to the optimum goal. The gamma (discount) should be large enough so that the algorithm could put more weights and trust on the future rewards, it also helps converging faster and in smaller iterations. One last important message from this test is that VI and PI could both converge to the same optimum rewards, however, VI takes more time and iterations in this problem in updating values in all states, on the other hand, the possible action in this question is not much so PI could converge way faster and less iterations. Another observation should be noticed is that the required runtime and iterations for VI in frozen lake problem grows exponential while PI is growing slower. Policy iteration is a better candidate in this problem.

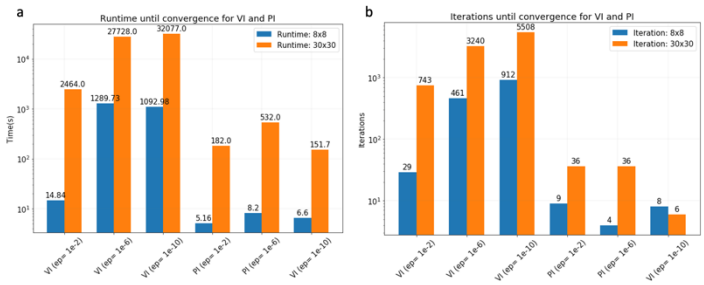

Figure 5 – (a) Runtime (b) Iteration required for convergence using value iteration and policy iteration with variant epsilon in 8x8 (blue) and 30x30 (orange) grid frozen lake problem.

In Q-learning, discount, iteration, learning and decay rate (gamma) has to be optimized. As the iteration is too low (1e4), there is not enough iteration to let Q-table find the optimize and converged Q-values. In my model, 1e6 for iterations in Q-learning algorithm was set. For learning rate, I tried the values of 0.1, and 0.01, and decay rate of 1e-3 and 1e-5. Figure 6 shows the rewards over 1e6 iterations using Q-learning on 8x8 frozen lake problem. Compared with red/ lightcoral and blue/light blue color, we can see that with fixed decaying rate, as the learning rate is larger, means the agent tends to have more weights on new value, the convergences happen in early iterations. The converged rewards are also lower in the case of low learning rate. Compare with red/blue color, as the decay rate is larger (red:1e-3), which means the importance of future rewards is higher, it accelerates the learning and convergence happens in shorter iterations. The ultimate reward from Q-learning in 8x8 case is ~0.8, which is worse than both PI and VI (both > 0.98). Compared with policy iteration, Q learning is more time-consuming, which is linearly dependent on iteration, and it seems work bad as the size of problem gets larger. I tried to implement Q-learning in a 30x30 frozen lake problem but failed to converge to a reasonable reward. Since Q-value calculation in the table is based on exploiting and exploring neighbors with higher values, I found that in 30x30 case, all of the values are zero, means there's no agent successfully reaching the goal. The Q-learner seems work bad in the problem that size is large and not too much rewards information could be obtained (in our case, only 1 reward point out of 900). Model-based VI and PI works better in this frozen lake problem. If more reward information could be added into this environment, for example, stay alive on frozen floor + 1, reach to

goal becomes +20, then the agent will tend to stay away from the holes and eventually get to the goal using Q-leaning algorithm.

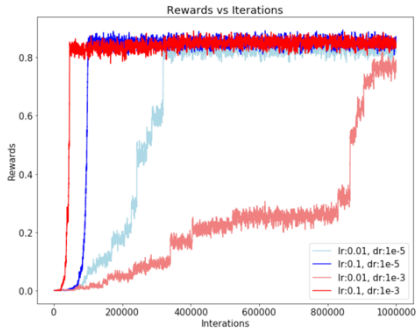

Figure 6 – Policies in 8x8 frozen lake problem using value iteration (a - d) and policy iteration (e - h) with different iterations (itr).

SECTION 2: FOREST MANAGEMENT

In forest management problem, I first started with the states of 1000, to see the influence of discount (gamma) and epsilon on performance metrics. In Figure 7, the rewards (7.a), iterations (7.b) and runtime (7.c) are summarized with the implementation of all three algorithms. In figure 7.a, we could see that the value and policy iteration almost have the same rewards (34) with discount ranging from 0.1 to 1, and Q-learner has relative bad rewards (8) performance compared to the other two algorithms. The trend is similar to previous problem, the rewards increase as the discount is closer to 1. In Figure 7.b, the iteration of Q-learning is set to be 1e6, so it will not be plotted here. As we can see, the iteration grows exponentially especially in value iteration, just like in frozen lake problem, the required iteration for finding optimum values in policy iteration is smaller than value iteration because in both problems, the action options for policy is small, there's no need for expensive calculation in each round of policy updating. In Figure 7.c, the runtime for all algorithms is plotted, Q-learning takes way more time than the other two method because it requires more iteration (1e6) to construct the optimized/converged Q-table. In terms of runtime, seems like this problem doesn't have too many states/actions like grid problem, so the runtime VI and PI are both smaller than 1 seconds.

The influence of epsilon is concluded in Figure 7.d to 7.f. The influence of epsilon is small for policy iteration, but the performance of the value iteration is highly dependent on the epsilon. This observation might result from the fact that choice

of action (cut/wait) is not much, so policy iteration could converge to the optimum easily. In figure 7.e, same story as before, the iteration for value iteration algorithm is more than using policy iteration, because it needs to keep propagating the future rewards, making it harder to convergence. For runtime, both algorithms are fast and the difference shown in Figure 7.f is at noise level.

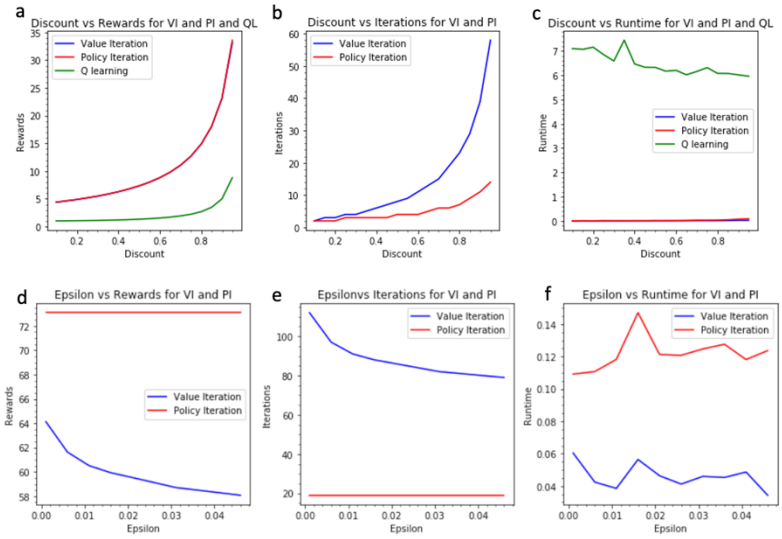

Figure 7 – Influence of discount on (a) rewards, (b) iteration, (c) runtime using 3 algorithms, and the influence of epsilon on (d) rewards, (e) iterations and (f) runtime in value and iteration policy.

After studying the influence of discount and epsilon in the case of 1000 states in forest management problem using all 3 algorithms, it is interesting to see how these algorithms work in different size of the problem. In Figure 8.a to 8.c, the rewards, iteration and runtime are calculated in two algorithms using value iteration and policy iteration, and the size of the problem ranges from 4 to 1000. In Figure 8.a, we can see that the rewards in policy iteration (72) are slightly better than value iteration (65) and independent of the problem size. The reason is that the value iteration is too sensitive to epsilon (Figure 7.d), if the epsilon is slightly off from the best parameter, then the reward will decrease. In Figure 8.b, the iteration is higher in value iteration (120) as usual compared with policy iteration (20). In Figure 8.c, the runtime is closed between two algorithms, but we can see the increasement of time for policy iteration is higher. As the size of problem gets larger, for example 1e5 or more, the runtime for policy iteration is expected to be more than value iteration. One interesting observation in this forest management problem is that the size didn't affect the performance on both

algorithms that much, possibly because of the small action option and states compared to frozen lake problem.

Q-learning was also implemented in this problem and the results are concluded in Figure 8.d and 8.e. The required runtime vs iteration in Q-learning is summarized in 8.d, and as expected, the runtime is linearly dependent on the iteration set for Q-learner. The rewards are studied under iterations ranging from 1e4 to 1e7, and the result is shown in Figure 8.e. The reward first increases dramatically then shows no more after 5e5. It means the iteration could help improving the accuracy of Q-learning, but after a specific point, no more enhancement could be obtained. The final converged rewards in Q-learning is 48, which is less than value iteration (64) and policy iteration (72), and it's also more time-consuming. Therefore, we can conclude that in forest management problem, policy iteration exhibits the best performance in terms of rewards and iterations, and Q-learning works relatively bad.

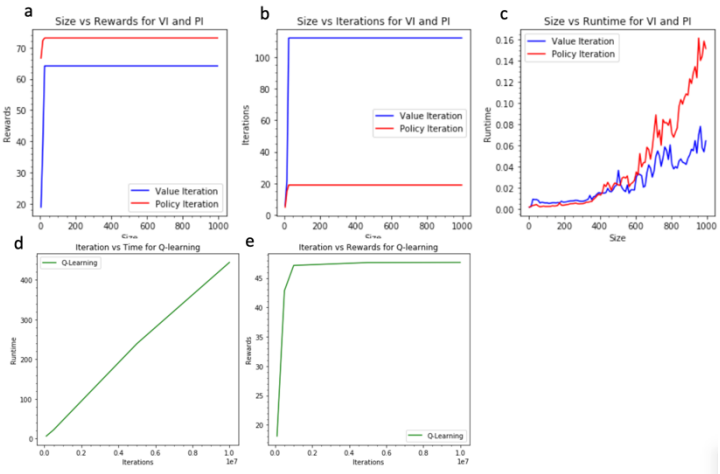

Figure 8 – Influence of size on (a) rewards, (b) iteration, (c) runtime using VI and PI algorithms, and the influence of iteraions on (d) runtime and (e) rewards in Q-learning.

4 REFERENCES

1. https://www.cs.ubc.ca/~kevinlb/teaching/cs322%20-%202008-9/Lectures/DT4.pdf
2. http://incompleteideas.net/book/first/ebook/node43.html
3. https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56
4. https://reinforcement-learning4.fun/2019/06/16/gym-tutorial-frozen-lake/
5. https://cdnsciencepub.com/doi/abs/10.1139/cjfr-2020-0447