CS 7641 Machine Learning HW4

Hao-Jen Wang

Abstract

Reinforcement Learning trains models in an indirect way in which the optimal policy is not known yet the only clue is the reward. In this project I focus on 3 types of learning methods to train on 2 classical RL problems and draw some analysis about their performances intuitively.

1. Introduction

I will train Value Iteration, Policy Iteration, and Q-Learning models on the Frozen Lake and Forest Management problems. Value Iteration consider the utility in a continuous space, so the convergence should be decided by a threshold. It is expected to run with many iterations if such a threshold is small. Policy Iteration focuses on the change of policy, so the utility may not be so precise. It solves N linear equations with the aid of matrix manipulation, so it is expected to run in less iterations. However, if the problem size is huge, the computation may be costly as solving rank N matrix has a complexity $O(N^3)$.

On the other hand, Q-Learning is a model-free method and is suitable to solve unknown or indeterministic policy environments. It updates utilities by actually walking in the environment and learn from experience, much similar to the gradient descent technique in machine learning. To escape from local minimum, it requires an exploration probability to discover unknown states, which is therefore more costly than the other two methods. Since this project assumes two problems with known environments and rewards, the advantages of Q-Learning may not be obvious throughout the experiments.

2. Problems

This section assesses the two RL problems. Frozen Lake is an interesting problem since its transition is very uncertain: only a 0.33 chance to land on the target, and a 0.67 chance to "slip away" from some expected next state, which is challenging. To get an intuition on how hard the problem is, we may apply a random walk agent in N-by-N grid worlds to see if it can reach the goal states with an ideal probability. Simulation is conducted in 10,000 times, and the evaluation is defined by the averaged final reward and the number of epochs to reach to goal. The result of simulation is shown in Figure 1. As we can see, the agent could only reach the goal with a 0.083 chance in 8x8 grid world, and even become just 0.0006 in 16x16. If we train agents outperform such a baseline, that is what we desire.

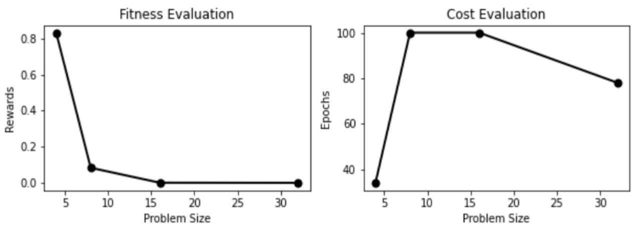

Figure 1. [Frozen Lake] Random Walk Models

Forest Management problem, on the other hand, is a deterministic and non-grid problem. Because the environment is much simpler, it is suitable to assess how good an agent is without other factors interfering.

3. Methods

This section focuses on 3 training schemes based on previous discussed problems. Since different methods have different parameters, we need to figure out some evaluation methods that are compatible across all algorithms. In Frozen Lake, we can just simulate agents in the world with sufficient episodes, as described earlier.

As for the Forest Management, we can also evaluate the final reward with some simulations, but I will explain such a number cannot be large in later discussions. Specifically, I assume that there are 1,000 trees in the forest and all of them have zero ages in the beginning. There will be 100 period for trees to grow, to be cut, or to be burned by a fire with a probability of 0.1. Each tree is growing independently. Eventually, I will measure how agents can at best earn the average rewards among 1,000 trees after 100 episodes.

Now I define the convergence rule for each method. The criterium of Q-Learning is to see whether the received reward is not fluctuated in a sequence of training epochs. Epoch is defined as the agent reaching a goal state. There will be a maximum number of epochs during training, and such a value is increased when the learning curve is not converged. However, the Forest Management problem has no absorbing states. In this problem, therefore, the unit of training is no longer the epoch but the iteration (transition). Both problems consider the received rewards as the convergence criteria.

As for Value Iteration (VI), the convergence criteria consider the delta value (i.e., difference between the last iteration and the current), and it is a common approach. For Policy Iteration (PI), the criteria are measured by the policy change.

4. Experiments

Now we run 3 RL methods on 2 problems. I will first apply the default setting of parameters, and then conduct ablation studies on tuning.

A. Frozen Lake – Q-Learning

The default parameter setting here is: alpha to be 0.1, gamma to be 0.99, epsilon to be 0.2, and decay to be 1e-4. Alpha is the learning rate and is expected to increase iterations with small values. Gamma is the discount and is expected to yield poor rewards for "short-sighted" agents (small gamma). Epsilon is the exploration rate to make agent to discover new states; decay is to decrease the reward if not entering absorbing states. The remaining parts study of different parameters affecting the model.

First, I tune decay value for non-absorbing states. Without decay, models are expected to have more epochs to reach the goal because the default do not consider ending the game earlier. Figure 2 shows how decay values influence the averaged rewards for each episode, and it turns out that different decay values do not obviously effect on the number of epochs to reach the goal, but on the stability of training. If too large (0.1), the agent is forced to end the game quickly and unable to find the route to the goal if such a path is long enough. Eventually, it yields almost zero reward in simulation. As decay becomes smaller, reward curves be less fluctuated. Gamma 1e-4 has the lowest reward variation and thus optimal.

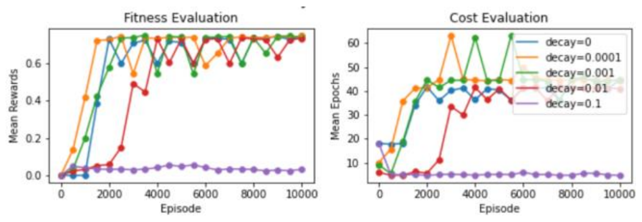

Figure 2. [Frozen Lake] Q Decay Evaluation

Second, we assess the alpha value. Figure 3 shows alpha greater than 0.5 makes the training stage become unstable, whereas alpha smaller than 0.01 needs too many episodes to converge.

As a trade-off, the optimal alpha here is 0.1.

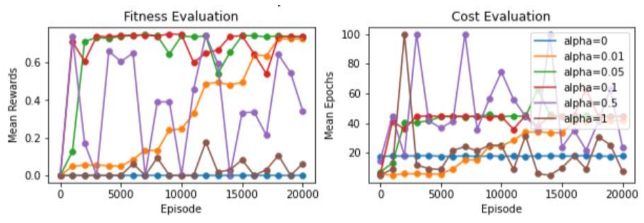

Figure 3. [Frozen Lake] Q Alpha Evaluation

Third, I evaluate gamma (discount factor) in this problem. Larger gamma implies long-term considerations. The result is shown in Figure 4, and only 0.99 yield a stable reward curve, while lower value makes the training stage unstable or even unable to learn some effective policy.

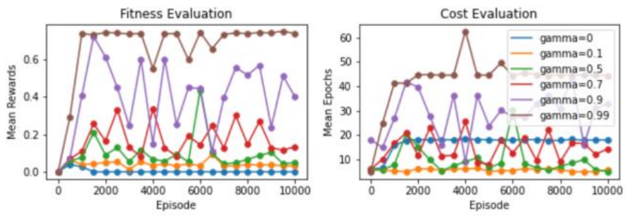

Figure 4. [Frozen Lake] Q Gamma Evaluation

Next, I measure how epsilon (exploration rate) affect the model. Without exploration, the model cannot discover new states in Q-Learning scheme. Figure 5 shows with sufficient epsilon, the training curve will be stable. The curve with gamma 0.2 yields the lowest reward variation.

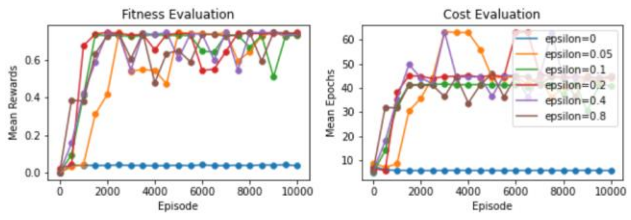

Figure 5. [Frozen Lake] Q Epsilon Evaluation

To this end, we have an optimal parameter setting for a 4x4 Frozen Lake default map. Let us visualize how the optimal policy is produced in training. Figure 6 shows the update history, where the upper figure is the policy update in different episode, and the lower one shows the learning curve. In the lower, the blue curve is the evaluation with exploration, and it is always worse than the red curve which only exploit the policy. On the other hand, the upper one shows how agent is at best trying to avoid the holes by moving toward opposite directions if holes are nearby. Despite it may require more episode to reach the goal, the agent learn how to avoid the uncertainty of slipping away. In the last episode, the agent receives 0.7436 average reward at the cost of 44.29 average number of walks.

```
#0     #1000  #2000  #3000  #4000  #5000  #6000  #7000  #8000  #9000  #9999
---------------------------------------------------------------------------
SFFF   ←↑←←   ←↑←←   ←↑←←   →↑↑↑   ←↑↑↑   ←↑↑↑   ←↑↑↑   ←↑↑↑   ←↑↑↑   ←↑↑↑
FHFH   ←←→←   ←←→←   ←←←←   ←←←←   ←←→←   ←←→←   ←←→←   ←←←←   ←←←←   ←←←←
FFFH   ↑↓←←   ↑↓←←   ↑↓←←   ↑↓←←   ↑↓←←   ↑↓←←   ↑↓←←   ↑↓←←   ↑↓←←   ↑↓←←
HFFG   ←→↑←   ←→↓←   ←→↓←   ←→↓←   ←→→←   ←→↓←   ←→↑←   ←→↓←   ←→↓←   ←→↓←
```

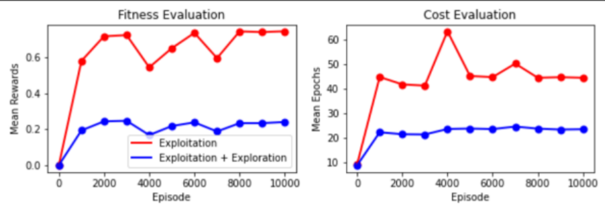

Figure 6. [Frozen Lake] Q Training History

Finally, we may assess if such a parameter works fine to problems with larger size. OpenAI Gym API generates the map with a probability to make a state frozen (i.e., non-absorbing state). Hence, we need to assess a proper value starting with 8x8 grid world. Figure 7 shows how agents receives the average reward with a running time given the probability. After 0.8, the agent starts to figure out a solution, and the chance to reach the goal increases as the hole is lesser. It might be interesting to measure the cofactor of such a probability, along with the problem size, to see the trend of reward and runtime, yet it is hard to measure and require lots of running resources. For simplicity, I fix the probability to be 0.98 in the future and vary the problem size only.

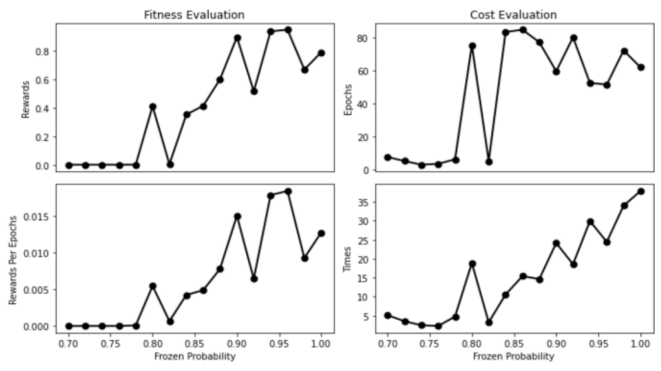

Figure 7. [Frozen Lake] Q Frozen Probability

With a fixed frozen probability 0.98, now I measure the problem size curve. The result is in Figure 8, and it shows two facts. As the problem size grows, the agent receives lower reward and be more prone to enter the holes, and it becomes ineffective to reach the goal after a size 16. It is because larger grid world has a greater number of holes given a fixed frozen probability. Agent becomes unlikely to reach the goal state with so many traps. Another fact is that as the episode goes, the number of walks is decreasing, which means agent will try to avoid hovering around.

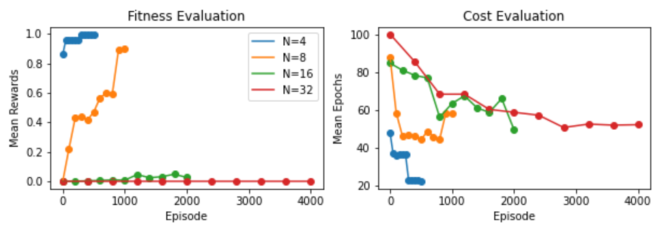

Figure 8. [Frozen Lake] Q Training Curve

Now we may evaluate how agent effectively learn a useful policy. Table 1 compares random walk performance with the agent trained via Q-Learning using the last checkpoint of previous model. As the problem size grows, the agent is making more improvement. Combined with the result in Figure 6, we know moving opposite to holes becomes a more important strategy as the problem size is bigger and bigger.

Table 1. [Frozen Lake] Q Test Performance

Problem Size	Random Walk	RL Agent	Improvement
4x4	0.8316	0.9993	1.20x
8x8	0.0833	0.9002	10.81x
16x16	0.0006	0.0482	80.33x
32x32	0.0000	0.0000	---

However, training via Q-Learning is costly. Generating the problem size curve in Figure 8 requires 21 minutes! It echoes what I mentioned in Problem Section: if we know environments and rewards beforehand, Q-Learning shows no advantages. In the upcoming section, let us see if VI and PI works faster or more accurate.

B. *Frozen Lake – Value Iteration*

VI requires lesser parameters than previous method. It only needs to tune gamma and delta (convergence criteria). Since VI cares about the exact value function, it may need more iteration than PI, and we may assess it in the next section. With a default setting of gamma to be 0.99 and delta to be 0.01, let us tune each separately.

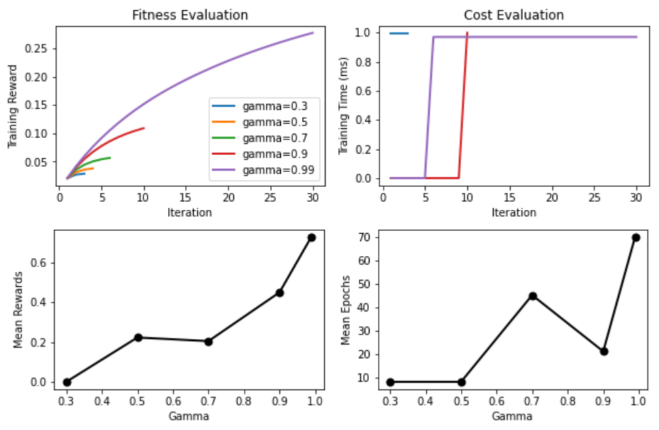

Figure 9. [Frozen Lake] VI Gamma Evaluation

First, I assess how gamma affect the model. The result is shown in Figure 9, where the upper graph measure how rewards and times change throughout iterations, and the lower one applies agent simulation across different gammas. Just as Q-Learning, large gamma has better reward. Model of 0.99 gamma needs 30 iterations and has 0.7335 reward in 10,000 simulations.

Second, I evaluate the delta convergence in Figure 10. Smaller deltas require more epochs to converge. An effective epsilon is 0.01 since a larger value requires 3 times more iterations yet only improve the reward marginally. It yields 0.7335 test reward with only 30 iterations.

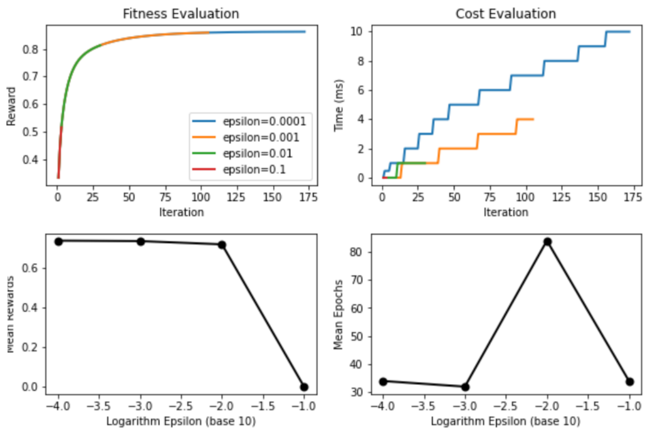

Figure 10. [Frozen Lake] VI Delta Evaluation

Finally, we may check how a different size affect the model with a fixed parameter. Curves are shown in Figure 11, where the left figure is the utility (not the actual reward) to test whether convergence only, and the right one is the time. It shows VI converges across different problem size, and training the agent requires less than a second! Notice, however, the required time is not linear or quadratic to the size, and for N=32 the time suddenly boosts from 8ms to 90.76ms. It may be due to the uncertainty of transitions.

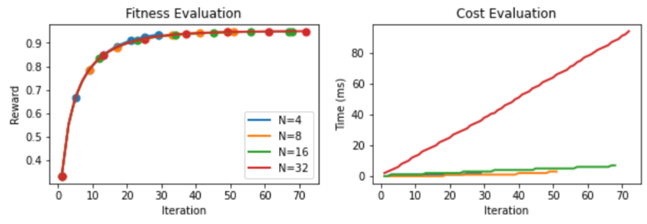

Figure 11. [Frozen Lake] VI Training Curve

The test performance is shown in Table 2. Compared with Q-Learning shown in Table 1, VI outperforms in all aspects, especially when the problem size is large!

Table 2. [Frozen Lake] VI Test Performance

Problem Size	Random Walk	RL Agent	Improvement
4x4	0.8316	0.9973	1.20x
8x8	0.0833	0.9925	11.91x
16x16	0.0006	0.7394	1232.33x
32x32	0.0000	0.0006	---

C. *Frozen Lake – Policy Iteration*

PI is even expected to yield a lower iteration to converge than VI since it does not consider the exact value of utilities, but rather focus on the policy change in a discrete space. The only parameter needs to be tuned is the gamma, and the plot is in Figure 11. Again, a larger gamma yields better rewards, where the model of 0.99 gamma has a final average reward 0.73 (shown in the lower figure) via 10,000 times simulation.

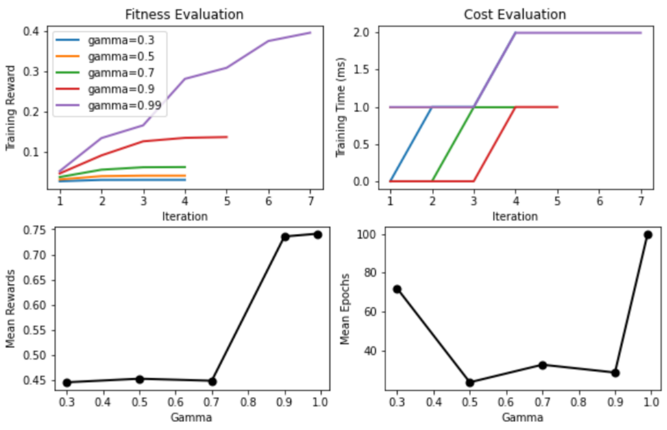

Figure 11. [Frozen Lake] PI Gamma Evaluation

Lastly, we may plot the PI training curve in Figure 12 given gamma 0.99. Compared to VI, it indeed requires less iteration to converge as shown in the x-axis of left figure: PI requires 20 iterations, whereas VI requires 72. However, PI requires longer runtime, especially for large N. This is because PI compute matrices to solve N linear equations, which is cubic to the problem size. A size of 32 requires 561.52ms, which is more than 6 times slower than VI. Whatever, it is still faster than Q-Learning.

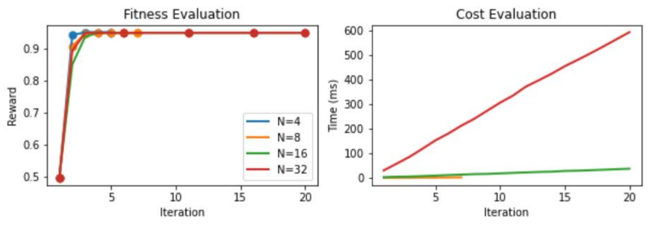

Figure 12. [Frozen Lake] PI Training Curve

The test performance is in Table 3, and it yields slightly lower rewards than VI, but still outperforms Q-Learning with large sizes. To this end, we may conclude that **VI is not only the fastest but also the most accurate, while Q-Learning is the worst regarding both.**

Table 3. [Frozen Lake] PI Test Performance

Problem Size	Random Walk	RL Agent	Improvement
4x4	0.8316	0.9973	1.20x
8x8	0.0833	0.9892	11.88x
16x16	0.0006	0.7132	1188.67x
32x32	0.0000	0.0004	---

In the last section of Frozen Lake analysis, let us see how the policies work on 3 methods. We know VI and PI significantly outperforms Q-Learning at 16x16 grid world given Table 1 to 3, and it is interesting to visualize the policies. Figure 13 is the result, where red rectangles are holes and green one the goal states. There are 6 holes in the map, and we can see that VI and PI will try to move to the opposite directions of the holes at best (except the hole lying at row 6 and column 5), and the policies looks very in order pointing to the optimal directions to goal state. In Q-Learning, however, many actions are made towards the holes unreasonably, and the overall policy look very unorder. Since Q-Learning is based on the agent experience in environment, the ultimate result is very unpredictable and has hard time reaching the goal if the maze is huge.

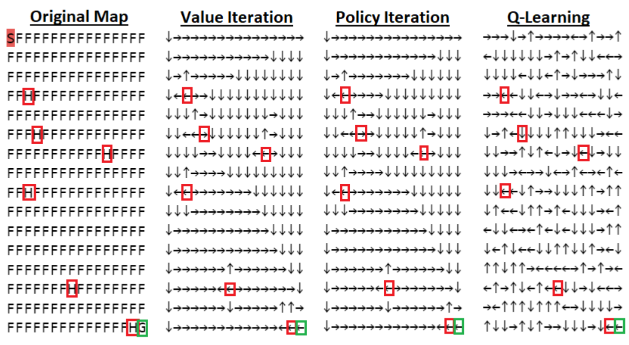

Figure 13. [Frozen Lake] Policies Visualization

D. *Forest Management – Q-Learning*

Now we change our focuses to a larger size problem. Throughout the experiment of Forest Management, I set the maximal tree age to be 2,000. Afterall, rare trees in real life can live longer than such a period of years.

As before, I will use the default parameters at first and tune them sequentially. Since a fixed alpha and epsilon tends to diverge, I introduce alpha decay and epsilon decay (to be 1e-5) here. They are the decay factors during the training to ensure convergence. The rest of the parameters have the same values as Frozen Lake. Note we do not need to tune non-absorbing state decay since there is no goal state in this problem.

First, I tune the alpha (learning rate). In the Frozen Lake case, too large alpha causes the instability to converge while too small alpha requires too many episodes. The default alpha is 0.2 here, and it is worth to know if it is still the optimal choice. The result is shown in Figure 14. Notice that the x-axis is no longer the number of episodes as there is no way to define reaching a goal as one episode. Rather, **it is the number of iterations (transitions).** Also, notice the criteria of convergence is measured by the shape of the training curve and will be increased if there is an obvious divergence. Besides, rewards on the y-axis of fitness evaluation are not the simulated rewards but **the utilities (value functions)**. It is nothing to do with the model performance but is only used to judge convergence.

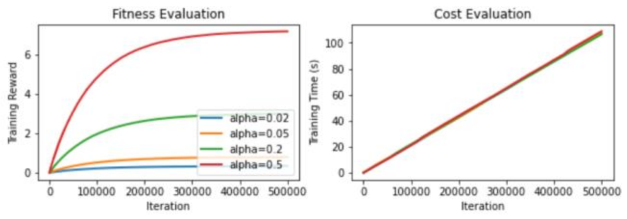

Figure 14. [Forest] Q Alpha Evaluation

There are two facts. First, the training times are the same across different alpha because the number of iterations is fixed. Second, and more interesting, it turns out that large alpha does not affect the stability of training curve. Perhaps in deterministic problems, a larger alpha is more suitable since the agent is not worrying about the uncertain transition. Thus, I will change the alpha from 0.2 to be 0.5 in future sections.

Second, I measure the effect of gamma. We know from previous result that a larger gamma is essential. The plot is in Figure 15.

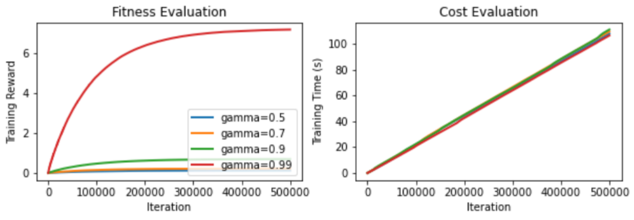

Figure 15. [Forest] Q Gamma Evaluation

Actually, we cannot tell an optimal gamma given the above figure, since value functions do not represent actual rewards. Evaluation shows that 0.99 gamma has marginally better rewards than the other (47.123 average rewards among 1,000 trees during 100 episodes). Thus, I will still use the gamma with a value of 0.99.

Third, I tune epsilons. Previous experiment shows epsilon is not very influential on model performance. Figure 16 again suggests with a large enough epsilon, the overall utilities and the runtimes are nearly identical. Without loss of performance, I again use the value of 0.2.

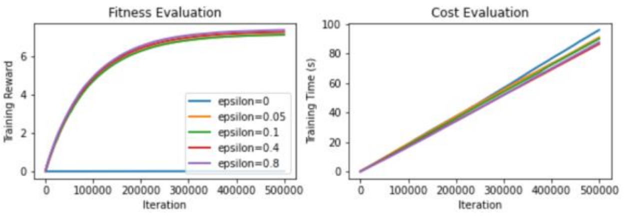

Figure 16. [Forest] Q Epsilon Evaluation

To this end, we know the optimal setting for Q-Learning (though not obvious unfortunately). Let us assess how different problem size affects the reward via simulations. Result is shown in Figure 17, and it turns out that no matter how large the problem size is (i.e., maximal tree age), the curve always become stable after sufficient episodes (about 40). It is because as time goes, most trees will eventually transform into state 0 by either begin fired or cut. Note that the burnt probability here is 0.1, so the survival chance after 40 episodes is less than 1.5%. After trees die, it is reasonable to wait them grow for only one episode, and then immediately cut it to get an instant reward 1. Future analysis will show how the policy look like. In the later episodes, therefore, the policy will form a **cut-wait cycle**, making the average reward unchangeable.

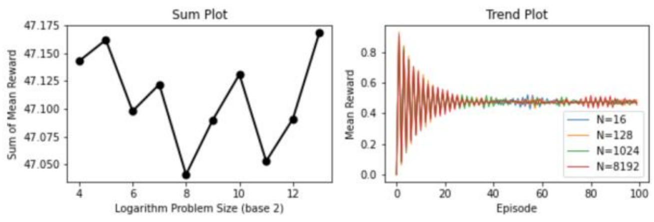

Figure 17. [Forest] Q Problem Size Curve

E. *Forest Management – Value Iteration*

As before, I evaluate different parameters to figure out the optimal setting. First, the gamma plot is shown in Figure 18. Again, large gamma is always more effective than smaller ones: as the iteration goes, the utilities also vary.

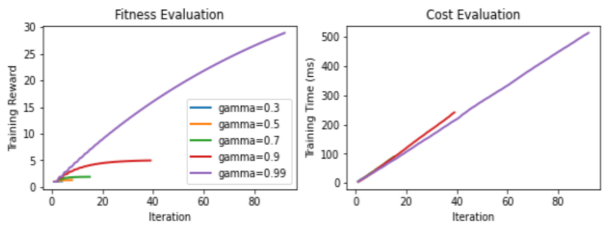

Figure 18. [Forest] VI Gamma Evaluation

Actually, the policy of VI is very simple: **decide to cut trees in young ages and wait if old enough**. Regardless of state 0 since it must be waited, the policy makes a threshold (which will be discussed in more detailed afterward).

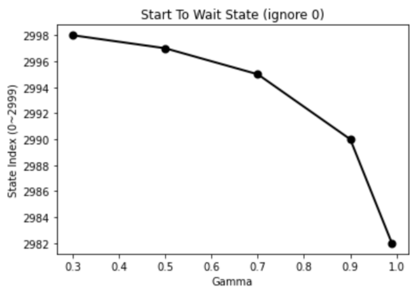

Figure 19. [Forest] VI Gamma Actual Affect

Given this knowledge, it may be easier to evaluate how gamma actually affect the policy. It turns out that as gamma becomes larger, the agent tends to wait for more younger old trees.

The fact is shown in Figure 19, where the y-axis represents the threshold of state to be regarded as "old enough". Small gamma makes model unable to hope for the future reward, so models will be "short-sighted" to cut the tree instantly.

Second, we can check how small the delta should be. The plot is shown in Figure 20, and we can see the default epsilon 0.01 has already yield non-trivial utilities in small iterations.

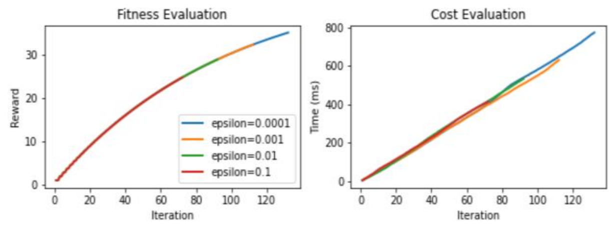

Figure 20. [Forest] VI Delta Evaluation

Last but not least, we may now assess how the problem size affect the model. The result is shown in Figure 21. It turns out while large size problem yields stable result with large enough episodes, there is a distinct pattern for the N=16 case. As described in Figure 19, VI policy will cut trees in younger ages. However, as the range of age (problem size) decreases, the agent starts to expect the possibility to reach the oldest state. It turns out that N=16 setting makes the policy becomes always awaiting. As the chance to get the oldest state increases, of course the overall reward is larger. It is however not applicable to larger size problems, since waiting the trees to grow for too long will at risk catching fires and eventually receive a zero reward.

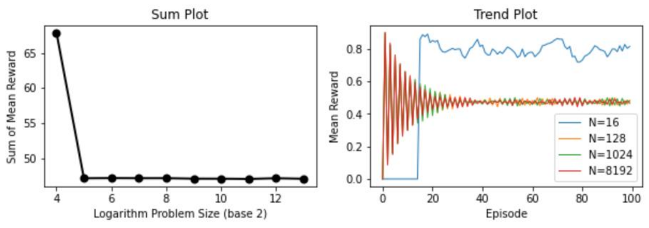

Figure 21. [Forest] VI Problem Size Curve

F. *Forest Management – Policy Iteration*

PI needs to tune the gamma only. Figure 22 shows there is no distinct pattern to choose an optimal gamma.

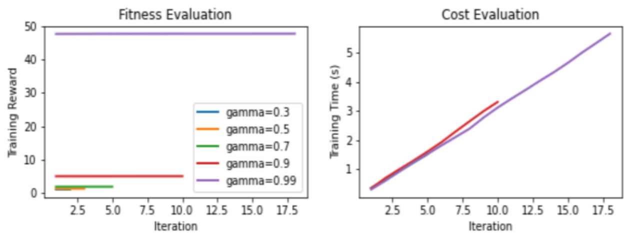

Figure 22. [Forest] PI Gamma Evaluation (1)

Another measurement is by looking at the training errors. Figure 23 shows the gamma of 0.99 is indeed learning something, as errors are monotonically decreasing. Though utilities are not changing, variation of errors indicates the policy is not unchanged. Detailed result shows that PI policy is actually identical to VI: it also makes decision based on a specific threshold!

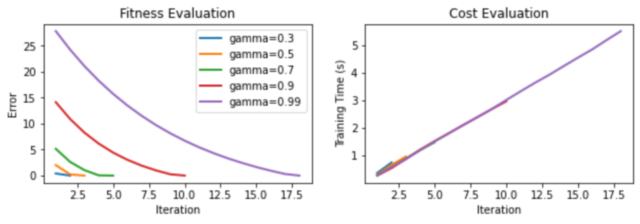

Figure 23. [Forest] PI Gamma Evaluation (2)

Finally, we may assess how problem size is affecting the reward in simulation. Figure 24 is the result, and its pattern is actually very similar to VI. As described earlier, PI sometimes yields the same policy as VI. The main reason to yield the difference is caused by randomness.

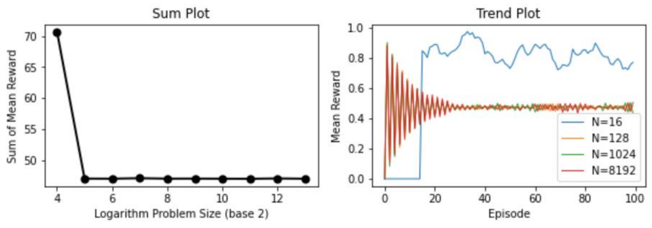

Figure 24. [Forest] PI Problem Size Curve

In the last section of Forest Management analysis, we may measure how the policy look like in practice. Figure 25 shows the policy plot for Q-Learning, VI and PI. Notice that action 0 represent to wait the tree to grow, whereas 1 to cut the tree. VI and PI yields identical result in the problem size of 3,000, so I merge them into a single plot on the right figure.

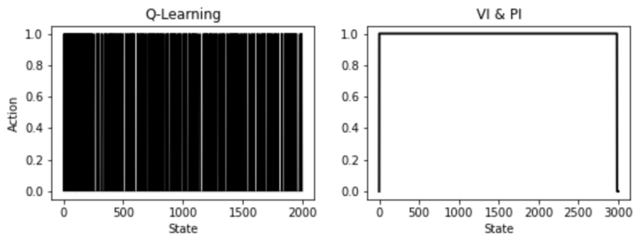

Figure 25. [Forest] Policies Visualization

We can see that Q-Learning has an irregular pattern of policy: waiting and cutting actions are crossing. It is very unreasonable. For example, if we observe a 20-aged tree and a 30-aged tree to be cur, there is no sense to wait for the tree to grow within the age 20~30, as it is impossible for these trees to reach to the oldest state. VI and PI both, on the other hand, follows a simple but reasonable policy: always cut the young one and keep the old one. We know young trees are hard to reach the oldest state since they may be burned during the episode. Thus, it is worthless to invest them waiting yet at risk receiving no reward in the end. Again, Q-Learning failed to make a reasonable policy in this problem.

5. **Discussion**

According to the previous result in Forest Management, we know PI and VI yield almost the same result. However, gamma evaluation plots in Figure 18 and 22 shows that they have different patterns of growing the utilities. I still do not figure out the exact reason behind such a fact, so I will further analyze it in the future.

Second, we know how Q-Learning, VI and PI relative performs in Frozen Lake shown in Table 1~3. However, it is hard to measure in the Forest Management case, since the simulation always run into a steady state with long enough episodes, forming a wait-cut cycle. Perhaps one way to judge their relative performances is by looking at the training records. The result shows in Figure 26, where the left figure is final utility for each different size, and the right one is the corresponded training time. Again, utilities do not reflect the goodness of models. Rather, I use them to check if the policy does not vary among different sizes too much. We know beforehand that the optimal policy is to cut young trees and keep old ones, so the expected long-term reward for most of the states do not vary a lot.

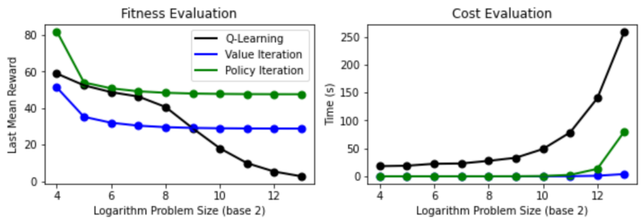

Figure 26. [Forest] PI Problem Size Curve

Considering the utility part, the mean utility for VI and PI converges as the problem size is large enough. Yet, Q-Learning decays in larger sizes. It implies a pessimistic idea for the agent to get ideal reward values in a long run if the range of tree age is large. Combined with results in Table 1~3 and Figure 25, I hypothesize that experience based RL agent cannot effectively figure out good policies in large-scale problems.

As for the runtime, the relative performance is more distinct: VI is always the fastest, and PI is fast in small size problems but eventually be slower with a large enough size. Unfortunately, Q-Learning is the slowest.

6. Conclusions

Here are some brief conclusions draw from the previous experimental results:

<u>In General</u>

- Gamma should be high for all RL methods

<u>Value Iteration</u>

- Best reward and fastest running time
- Suitable for large-scale problems

<u>Policy Iteration</u>

- Fewest iterations
- Suitable for middle-scale problems
- Unsuitable for large-scale problems because manipulating matrices is too costly

<u>Q-Learning</u>

- Poorest reward and longest running time
- Unsuitable for problems with known reward and environment; use VI or PI instead
- Exploration rate is not very influential
- Too small alpha converges slower, while too high alpha tends to diverge