Machine Learning, Assignment 4 Report

Seyed Aghigh

Gatech Email: saghigh3@gatech.edu

1 Introduction

In this assignment two problems of reinforcement learning (RL) are chosen, one grid world and another one consider as non-grid world; ***Frozen Lake***, ***Forest Management*** respectively. Each problem is tested on two world sizes; ***small*** and ***large***. The following RL optimization algorithms are evaluated on each problem with the considered world size; Value Iteration (VI), Policy Iteration (PI), and Q-learning (QL).

For each problem on each world size, below outline is as followed:

(a) Game Description and Formalization as Markov Decision Process (MDP)
(b) RL Optimization Algorithm Implementations
(c) Hyper-parameters Tuning & Result Comparison
(d) Graphs and Conclusion

2 Code Accessibility

All implemented functions and Read-me file are available in the following link
https://drive.google.com/drive/folders/1RZ5m9zVrK6_LWuEss_FAh8M2E1bNQNit?usp=sharing

3 Frozen Lake

Frozen Lake (FL) is a grid world game in which, by default, the agent will be rewarded only if it reaches the Gold (G) state from Start (S) state. There are two other states in between; Hole (H) and Frozen (F). If the agent falls in H, the game will be terminated. The agent reaches G state by stepping through the F states. The action space is a set of four movement; O: Left, 1: Down, 2: Right, 3: Up. Observation state is a value representing the agent's location. In this study, two world sizes are considered for FL problem; 8×8 ***as for the small world*** and 20×20 ***as for the large world***. The FL game, at its default, has stochastic nature, with agent moving toward intended direction by probability of $\frac{1}{3}$ and moving in an unintended perpendicular direction of intended move with probability of $\frac{1}{3}$ each way.

The following is the reward distribution in this game at its default;

- +1 for reaching G state
- 0 for reaching Hole (H) state
- 0 for reaching Frozen state (F)

Figure 1 shows the two world size layout; 8×8 Figure 1a and 20×20 Figure 1b.

(a) 8×8 world size.

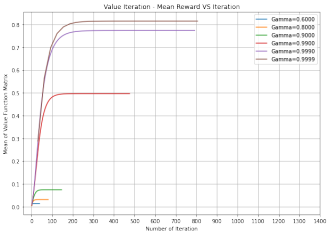
(b) 20×20 world size.

Figure 1: FL world size.

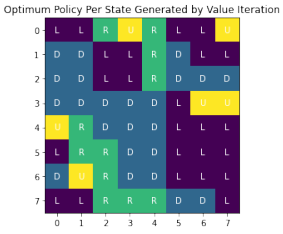
3.1 8 × 8 Frozen Lake Optimization Algorithms

3.1.1 Value Iterations (VI)

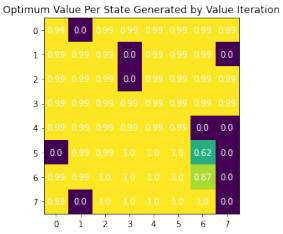
Figure 2 shows the results for VI. In Value iteration optimization technique, to reach golden state, agent steps into states which has higher state value. Figure 2a depicts the effect of Gamma hyper-parameter on number of iteration and mean of state-value matrix. Essentially Gamma determines how much agent cares about future reward in relative to immediate reward. The higher the gamma, the higher number of iterations is needed to reflect and update future rewards for all states and therefore the longer it takes to converge. Furthermore, since the only state that propagates reward is golden state (final state) by increasing the gamma, higher mean of value (reward) is achieved. Figure 2b shows the optimum per state policy generated using VI algorithm after choosing optimal Gamma (0.9999). The delta threshold which concludes the convergence criteria in Value Iteration algorithm, is equal to 1e-10. If in Nth iteration the differences of newly calculated state-value matrix with N-1 iteration state-value matrix is less than delta the convergence is concluded. Figure 2c shows the optimal state-value matrix generated by VI.



(a) VI Gamma hyper-parameter effect.



(b) VI optimal policy.

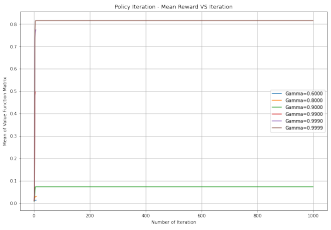


(c) VI optimal value.

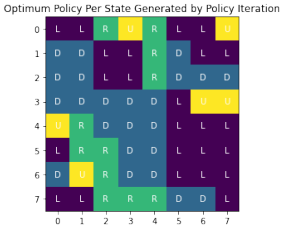
Figure 2: Results for VI on 8 × 8 Frozen Lake.

3.1.2 Policy Iterations (PI)

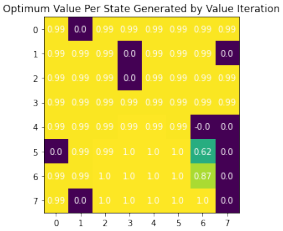
Figure 3 shows the result for PI. In Policy Iteration optimization technique, to reach golden state, agent steps into states based on an action which maximizes the state-value function. it starts with randomly generated policy. At each iteration a value function is calculated using Bellman Equation for each states. In policy improvement step, the action that maximizes the value function at state S is then compared with old action, if same then it continues to other states if not same, policy for that state is replaced with the action that maximizes the value function. Theses steps continues until there are no improvement in state-policy, which concludes convergence. Figure 3a Shows the effect of gamma in number of iteration and mean reward achieved. As far as number of iteration for convergence there is little to no difference cause this technique is policy based. But in terms of achieved mean reward, again since gamma, propagates future reward to early states, the higher the gamma the higher reward is achieved. Figure 3b and Figure 3c shows the optimum policy and optimum state-value matrix generated using Policy Iteration technique after choosing optimal Gamma (0.9999).



(a) PI Gamma hyper-parameter effect.



(b) PI optimal policy.

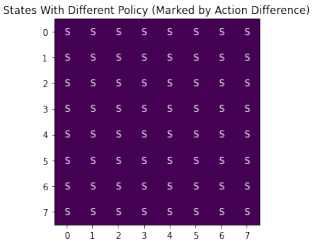


(c) PI value function.

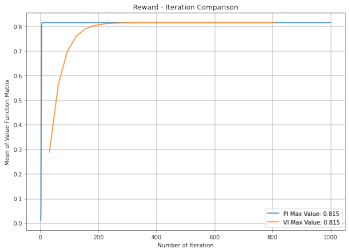
Figure 3: Results for PI on Frozen Lake 8 × 8.

3.1.3 Value Iteration vs Policy Iteration

Figure 4 compares VI and PI. Figure 4a shows the per state policy difference generated by VI and PI. Letter D, indicates states at which difference actions were taken based different policies generated by VI and PI. Letter S, indicates no action difference as per policy. As you can see VI and PI have converged to exact same policy. Figure 4b shows the maximum mean reward achieved per technique. Both algorithm have converged to same mean value. Figure 4b also shows Value Iteration technique takes significantly longer iteration to converge to an optimum policy.



(a) VI and PI policy difference. ("S": Same, "D": Different)

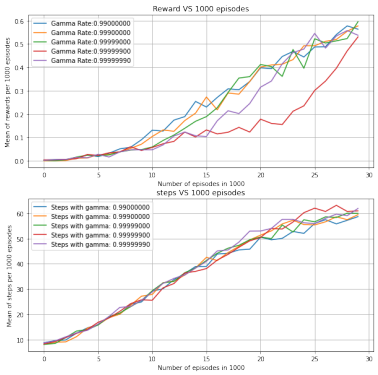


(b) Iteration Mean Reward Comparison for VI vs PI.

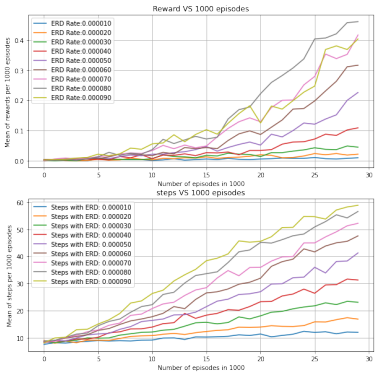
Figure 4: VI vs PI on Frozen Lake 8×8.

3.1.4 Q Learning (QL)

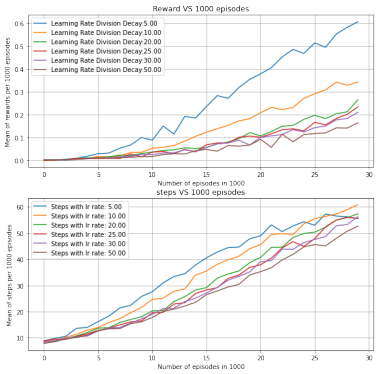
Q-Learning is a popular model free reinforcement technique. In Q-Learning, the agent actually learns the optimal policy after playing the game for thousands or millions, depending on complexity of game, episodes. For Frozen lake problem, I've developed a custom Q-Learning function using Openai toolbox. Two main target variables are watched for this method. Average reward and average number of agent steps per episodes. For smoothing the result, I keep track of mean reward and mean steps per 1000 episodes. Agent steps are tracked to make sure, given hyper-parameters value, agent explores the states just enough to reach golden state. Four hyper-parameters are considered and tuned for reaching optimum result. Figure 5 shows the result for all hyper parameters tuning. Figure 5a shows gamma value of 0.99999 reaches the highest mean reward. Figure 5b show the effect of exploration-exploitation trade off, the smaller the decay, agent has more opportunity to explore the environment (more number of steps). Figure 5c shows the effect of learning rate decay on performance and number of steps taken by agent. Learning rate dictates how much we should accept new values versus old value, or how much of future rewards, we should propagate to current state. Figure 5c shows the smaller the decay, the higher reward. Last but not least, Figure 5d shows number iterations (episodes) effect on performance, as you can see, algorithm converges at 100000 episodes, increasing the number of iteration holds no advantage.



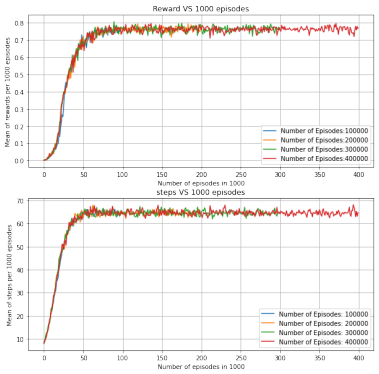
(a) Gamma hyper-parameter.



(b) Exploration Decay Rate hyper-parameter.



(c) Learning Rate decay hyper-parameter.



(d) Episode hyper-parameter.

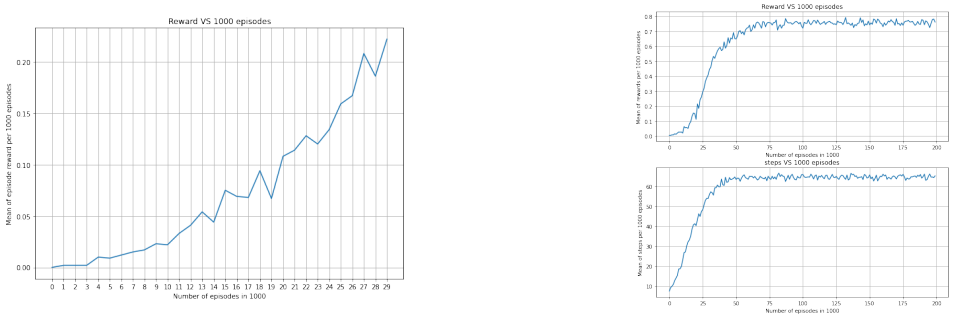
Figure 5: Results for QL hyper-parameter tuning on Frozen Lake 8×8.

Figure 6 Shows the result for optimum QL algorithm after hyper-tuning. Figure 6a depicts the optimal policy generated by Q-Learning, if we compare the QL generated policy with VI/PI Optimum policy, only 3 out of 64 states have different policy. Figure 6b Shows the optimum Q table generated by QL optimization algorithm.

(a) Optimal QL State-Policy **(b)** Optimal QL Q-Table

Figure 6: State-Policy and Q-Table for optimal QL on Frozen Lake 8×8ß.

Figure 7 Shows the performance comparison of naive vs optimal QL after hyper-tuning. The performance of QL algorithm has drastically improved after hyper-tuning.



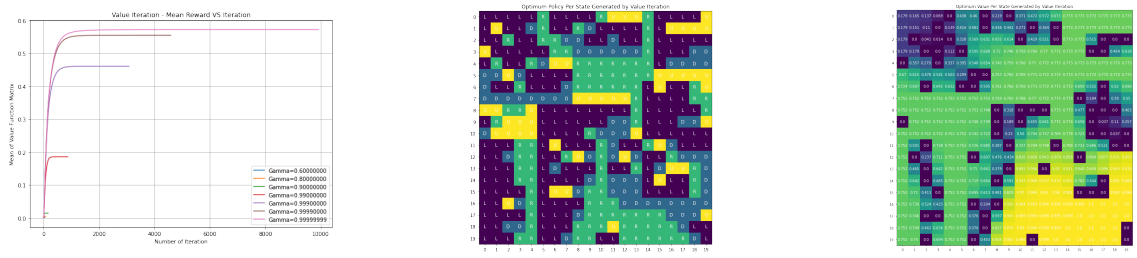
(a) Performance of Naive QL. **(b)** Performance of Optimal QL.

Figure 7: Results for Naive QL vs Optimal QL on Frozen Lake 8×8.

3.2 20×20 Frozen Lake Optimization Algorithms

3.2.1 Value Iterations (VI)

Figure 8 shows the results for VI algorithm implementation on large state size FL problem. Figure 8a depicts the effect of Gamma hyper-parameter on number of iteration and mean of state-value matrix. Comparing Figure 2a with Figure 8a, as you can see it takes drastically larger number of iterations, almost 10 times more, for large state size problem to converge, while small state converges after 150 iterations, it takes 1500 iterations for large state to converge. In addition, average reward achieved in large state is almost half the average reward achieved in smaller state (0.42 vs 0.82). This makes sense due to increase complexity (Size) of larger state the agent can not perform as well as smaller state size. Figure 8b shows the optimum per state policy generated using VI algorithm after choosing optimal Gamma (0.99999999). As we have higher number states, gamma value needs to be relatively bigger to properly propagate golden state reward to early states. Figure 8c shows the optimal state-value matrix generated by VI.



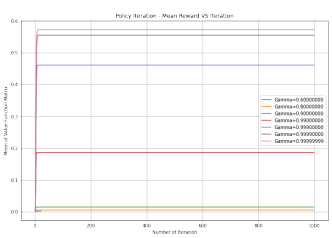
(a) VI Gamma hyper-parameter effect. **(b)** VI optimal policy. **(c)** VI value function.

Figure 8: Results for VI on Frozen Lake 20×20.

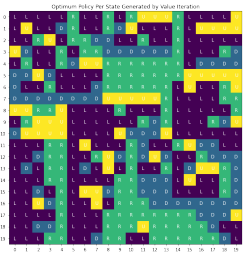
3.2.2 Policy Iterations (PI)

Figure 9 shows the result for PI. Figure 9a Shows the effect of gamma in number of iteration and mean reward achieved. There is little to no difference in number of iterations needed per gamma, cause this technique is policy based. But in terms of achieved mean reward, again since gamma, propagates future reward to early states, the higher the gamma the higher reward is achieved. Comparing Figure 9a and Figure 3a, one can see it takes relatively longer number of iteration to converge to optimum policy in large state size problem. This is

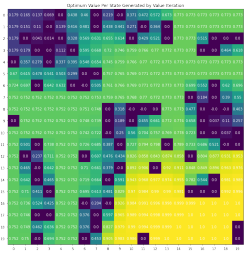
due to larger pool of available policies in larger state size problem. Figure 9b and Figure 9c shows the optimum policy and optimum state-value matrix respectively generated using Policy Iteration technique after choosing optimal Gamma (0.99999999)



(a) PI Gamma hyper-parameter effect.



(b) PI optimal policy

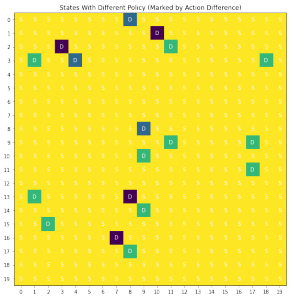


(c) PI value function.

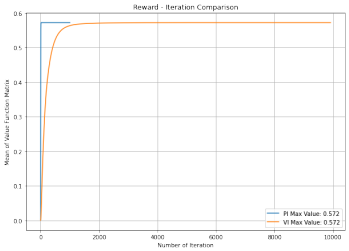
Figure 9: Results for PI on Frozen Lake 20 × 20.

3.2.3 VI vs PI

Figure 10 compares VI and PI. Figure 10a shows the policy difference. As you can see VI and PI algorithm have converged to different policies. 18 out of 400 states have different polices, for instance at state 54 for VI policy , agent has decided to move Left, instead of Upward dictate by PI. Figure 10b shows the maximum mean reward achieved per technique. Despite of policies differences, both techniques have converged to same maximum mean reward, which could indicate the possibility of multiple optimal policies. Figure 10b also shows Value Iteration technique takes significantly longer iteration to converge to an optimum policy.



(a) VI and PI policy difference. ("S": Same, "D": Different)



(b) Iteration Mean Reward Comparison VI (Orange) vs PI (Blue).

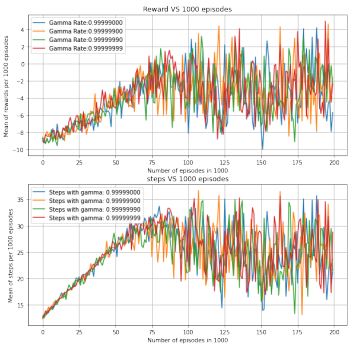
Figure 10: VI vs PI on Frozen Lake 20 × 20.

3.2.4 Q Learning (QL)

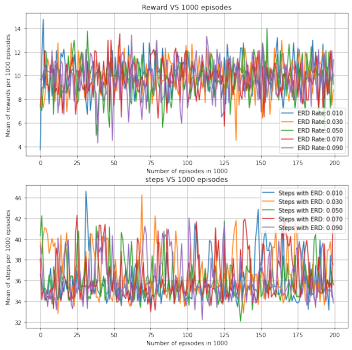
Again, In this problem size, two main target variables are watched. Average reward and average number of agent steps per episodes. For smoothing the result, I keep track of mean reward and mean steps per 1000 episodes. Figure 11 Shows the hyper-tuning parameters effect on two target variables, In this problem, as we are dealing with bigger environment, It's really hard for the agent to find it's way to Gold state without knowing how to avoid holes, considering the stochastic nature and 0 rewards for all Holes and regular steps. I found it really challenging to train algorithm. So I decided to train my agent how to avoid holes through Reward Shaping. In an essence, in reward shaping we change the distribution of reward per terminal states, now reward schedule is as follow:

- +100 for reaching Golden state
- -10 for reaching Hole (H) state
- 0 for reaching Frozen state (F)

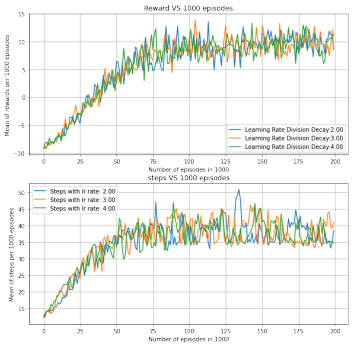
as Figure 11c shows, learning rate decay had the highest impact on convergence. Figure 11a and Figure 11b depict, gamma and exploration-exploitation decay rate did not have much of impact on algorithm performance, and in fact the default values were already optimized.



(a) Gamma hyper-parameter tuning.



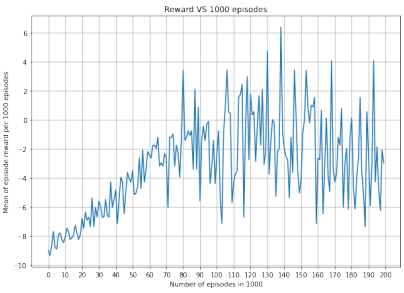
(b) ERD hyper-parameter tuning.



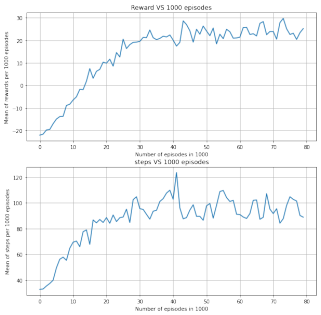
(c) Learning Rate decay hyper-parameter tuning

Figure 11: Results for QL hyper-parameter tuning on Frozen Lake 20 × 20.

Figure 12 Shows the performance comparison of naive vs optimal QL after hyper-tuning. The performance of QL algorithm has drastically improved after hyper-tuning.



(a) Performance of Naive QL.



(b) Performance of Optimal QL.

Figure 12: Results for Naive QL vs Optimal QL on Frozen Lake 20 × 20.

4 Forest Management (FM)

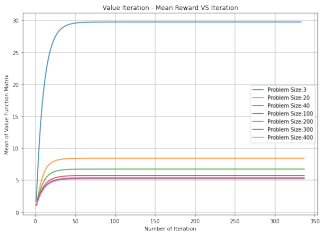


Figure 13: Problem size effect on Forest Management.

Forest Management (FM) is a non-grid world game. The forest in managed by two actions 0: Wait and 1:Cut, the main objective is to keep forest (Use Wait Action) for wildlife and second objective to make money by selling the lumbers generated by Cut action. The agent decides on an action each year towards maximizing the objectives. There are two rewards schedule in this game, agent receives r1 reward when the forest is in its oldest state and action 'Wait' is performed. And agent receives r2 reward when the forest is in its oldest state and action 'Cut' is performed. Forest Management, at its default, has stochastic nature, each year with probability of 0.1, at default, the forest burns and goes back to its youngest state (s = 0). The following is the reward distribution chosen in this game;

- +4 for waiting long enough to reach oldest state
- +2 for cutting at oldest state
- 0.2 Probability of burning forest at each state

Figure 13 investigates problem size effect on Value Iteration technique. One can observe as problem size increases, the mean collected reward decreases, this is due to stochastic nature of the problem. Essentially there is only $(1-p)^s$,with s being the problem size, chance of reaching oldest state and receive the highest reward. Therefore as the problem size increases the odds of reaching oldest state decreases and so as the mean reward. I believe the same analogy applies to number of iteration, as problem size increases the action of cutting becomes more attractive and common, therefore it would take less number of iterations to converge to optimum policy. Whereas in smaller problem size, if actions of cutting or waiting is equally weighted, then the model takes longer iteration to find optimum combination and converge.

4.1 20 States Forest Management Optimization Algorithms

4.1.1 Value Iterations (VI)

Figure 14 depicts the effect of Gamma hyper-parameter on number of iteration and mean of state-value matrix. Again, higher the gamma, the higher number of iterations is needed to reflect and update future rewards for all states and therefore the longer it takes to converge. In Figure 14 one can observe for sufficiently large gammas ($\geq$ 0.999) the default max number of iteration (1000) is not large enough to allow the model to converge. Figure 15 shows the optimal policy derived from Value Iteration technique, Letter W is representing Wait action, whereas letter C, represent Cut action. Figure 16 shows the optimal state-value generated from VI algorithm.

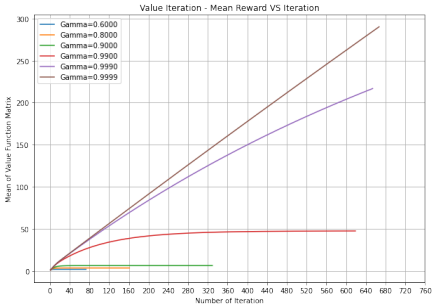


Figure 14: VI Gamma hyper-parameter effect for Forest Management 20.

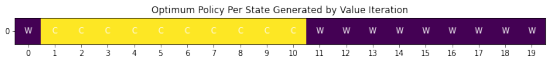


Figure 15: VI optimal policy for Forest Management 20 (W:Wait & C: Cut)

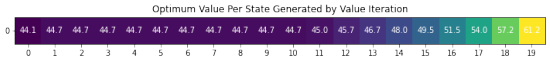


Figure 16: VI value function for Forest Management 20.

4.1.2 Policy Iterations (PI)

Figure 17 Shows the effect of gamma in number of iteration and mean reward achieved. There is little to no difference in number of iterations needed per gamma, cause this technique is policy based. But in terms of achieved mean reward, again since gamma, propagates future reward to early states, the higher the gamma the higher reward is achieved. Figure 18 and Figure 19 shows the optimal per state policy and state-value achieved using Policy Iteration technique respectively.

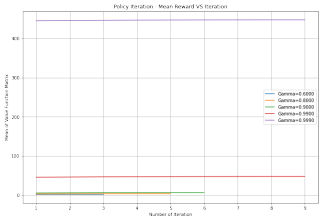


Figure 17: PI Gamma hyper-parameter effect for Forest Management 20.

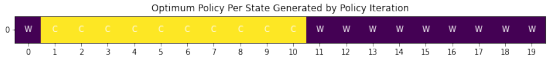


Figure 18: PI optimal policy for Forest Management 20 (W:Wait & C: Cut)

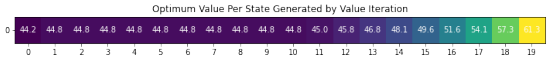
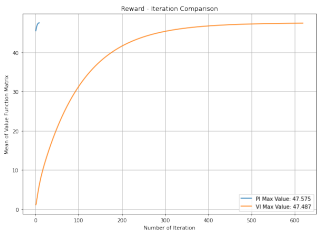


Figure 19: PI value function for Forest Management 20.

4.1.3 VI vs PI

Figure 20 compares VI and PI on FM 20. Figure 20a shows the per state policy difference generated by VI and PI. As you can see VI and PI have converged to exact same policy. Figure 20b shows the maximum mean reward achieved per technique. Both algorithm have converged to **almost** same maximum mean value. The policy iteration value is few decimals higher than value iteration, the reason is that, the delta scale is not as small as it needs to be and I haven't run the VI algorithm for a longer than default max iteration (1000), otherwise if the delta was sufficiently small and number of iteration was large enough, it would converge to same value. Figure 20b also shows Value Iteration technique takes significantly longer iteration to converge to an optimum policy.

(a) VI and PI policy difference.

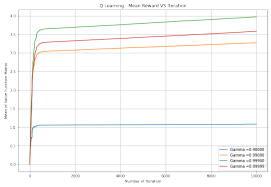


(b) Iteration Mean Reward comparison for VI vs PI.

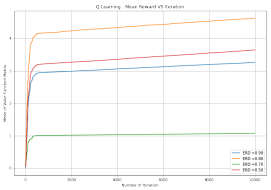
Figure 20: VI vs PI Forest Management 20.

4.1.4 Q Learning (QL)

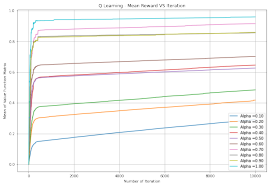
For Q-Learning algorithm in Forest Management (FM) setting, hiive mdptoolbox library is used. In this problem effect of four different hyper-parameters on Q-Learning algorithm performance is studied and showed in Figure 21. Figure 21a shows the gamma effect on collected rewards, as gamma increases, model achieves higher mean rewards. From Figure 21b optimum decay rate for exploration-exploitation parameter is obtained (0.8). Figure 21c reveals the optimal learning rate value and Figure 21d shows the learning rate decay value effect on model performance. Last but not least Figure 21e shows the high impact of number iterations on model performance, as the number of iteration increases the performance of model almost linearly improves.



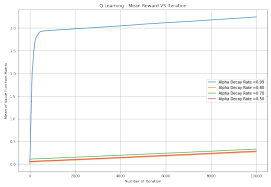
(a) Gamma hyper-parameter.



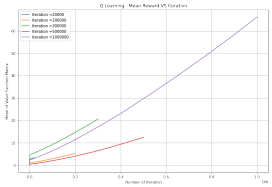
(b) ERD hyper-parameter.



(c) Learning Rate hyper-parameter.



(d) Learning Rate decay hyper-parameter.



(e) Episodes hyper-parameter.

Figure 21: : Results for QL hyper-parameter tuning on Forest Management 20.

4.1.5 Naive vs Optimal QL

Figure 22 and Figure 21e Shows the performance comparison of naive vs optimal QL after hyper-tuning. The performance of QL algorithm has drastically improved after hyper-tuning. Figure 23 and Figure 24 highlight the policy difference, before and after tuning.

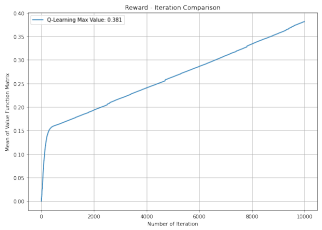


Figure 22: Performance of Naive QL Forest Management 20.

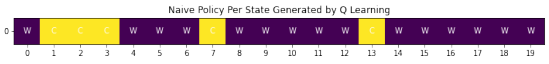


Figure 23: Naive QL policy for Forest Management 20.

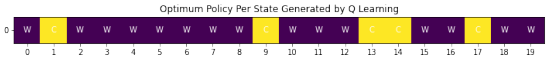


Figure 24: Optimal QL policy for Forest Management 20.

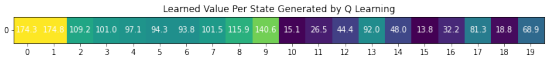
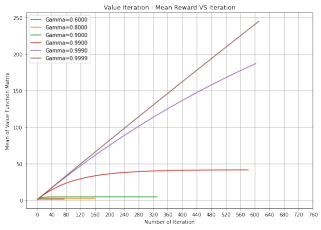


Figure 25: Optimal QL value function for Forest Management 20.

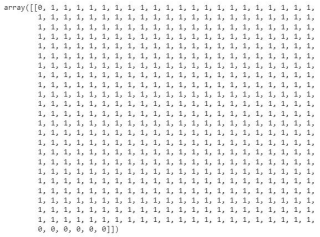
4.2 512 States Forest Management Optimization Algorithms

4.2.1 Value Iterations (VI)

Figure 26 shows the result gained from VI method on large state FM problem. Figure 26a depicts the effect of Gamma hyper-parameter on number of iteration and mean of state-value matrix. Same rule applies, the higher the gamma, the higher mean rewards would be. In this problem, I had to decrease the delta to 1e-90 so that mean reward value converges to a value obtained from PI method. For sufficiently large gammas ($\geq$ 0.999) the predefined max number of iteration (2000) is not large enough to allow the model to converge. Figure 26b shows the optimal per state policy, in form of matrix, generated from VI algorithm for large states.



(a) Gamma hyper-parameter effect.

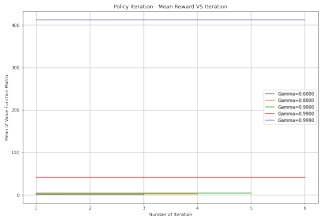


(b) VI optimal policy.

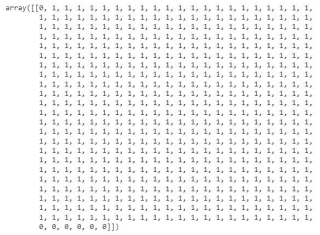
Figure 26: Results for VI on Forest Management 512.

4.2.2 Policy Iterations (PI)

Figure 27 shows the result obtained from PI method. Figure 27a Shows the effect of gamma in number of iteration and mean reward achieved. The very same effect was spotted. Figure 27b shows the optimal per state policy using Policy Iteration technique.



(a) Gamma hyper-parameter effect.

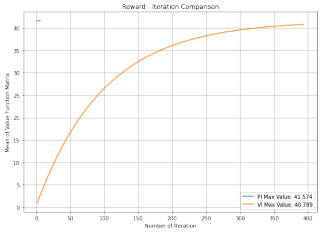


(b) PI optimal policy.

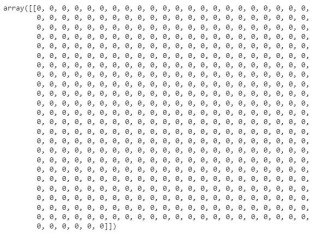
Figure 27: Results for PI on Forest Management 512.

4.2.3 VI vs PI

Figure 28 compares VI and PI results on FM 512. Figure 28a shows the maximum mean reward achieved per technique. Both algorithm have converged to **almost** same maximum mean value. The policy iteration value is one point and few decimals higher than value iteration. due to very same reason explained in section 4.1.3 Figure 28b shows that both methods have converged to same policy (all elements of policy-difference matrix is equal to zeros). **Comparing** Figure 28a and Figure 20b one can see it took longer number of iterations for small size FM versus large size FM to converge (620 vs 390) again this could be due to reason explained in problem size effect, Figure 13. When it comes to reward, the mean reward collected in smaller state size is higher than large state size, again due to the fact, as the number of state increases chances of wining get smaller and smaller $(1-p)^s$.



(a) VI and PI policy difference.

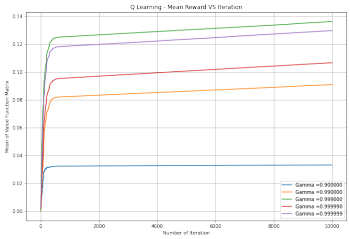


(b) Iteration Mean Reward comparison for VI vs PI

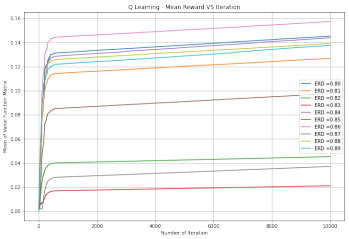
Figure 28: VI vs PI on Forest Management 512.

4.2.4 Q Learning (QL)

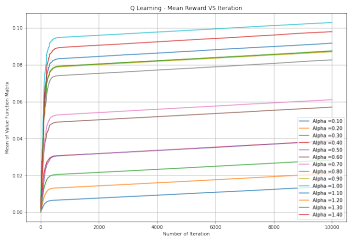
For Q-Learning algorithm in Forest Management (FM) setting, hiive mdptoolbox library is used. In this problem effect of four different hyper-parameters on Q-Learning algorithm performance is studied and showed in Figure 29. Figure 29a, Figure 29b, Figure 29c and Figure 29d show the effect of each hyper-parameter on model performance. With number of iteration being the most influential factor in improving mean reward.



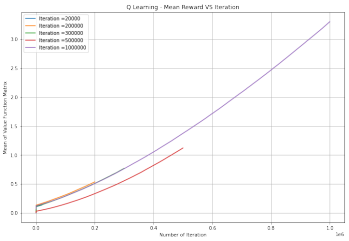
(a) Gamma hyper-parameter.



(b) ERD hyper-parameter.



(c) Learning Rate decay hyper-parameter.

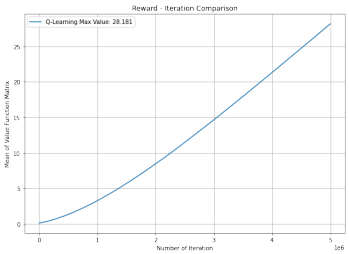


(d) Episode hyper-parameter.

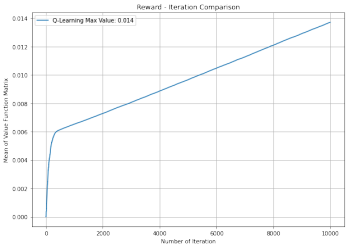
Figure 29: Results for QL hyper-parameter tuning on Forest Management 512

4.2.5 Naive vs Optimal QL

Figure 30 Shows the performance comparison of naive vs optimal QL after hyper-tuning. The performance of QL algorithm has drastically improved after hyper-tuning. **Comparing** Figure 30a and Figure 21e one can see it took significantly smaller number of iterations for QL algorithm on smaller state size in compare to larger state size (1 million vs 5 millions) to achieve rewards score of 65 versus 28. This signify the complexity of larger state size (512 states) versus smaller state size.



(a) Performance of Optimal QL.



(b) Performance of Naive QL.

Figure 30: Results for Naive vs Optimal QL on Forest Management 512.

4.3 Conclusion

In this assignment two types of Reinforcement Learning problems were selected. Grid World and Non-Grid World. Two model based and one model free RL algorithms applied and result were presented and discussed. In grid world problem, we observed for Value Iteration in Figure 10b and Figure 4b as the problem size increased, the number of iterations needed for convergence also increased significantly (800 vs 8000). In case of Policy iteration no significant difference were spotted. For QL method, it was highly depended on how well the model was tuned. In addition, as the complexity of environment increased the average collected reward by agent also decreased in all three algorithms.

In Non-grid world problem type (Forest Management) we saw a some what different behaviour. According to Figure 28a and Figure 20b as the problem size increased the number of iterations needed for convergence decreased. This could be due to reason that, when state size increases, Cutting action becomes more attractive since the chance of reaching to oldest state decreases exponentially based on $(1-p)^s$ formula, therefore obtaining the optimum policy is easier and faster. As far as reward, the same analogy applies. As complexity increases, the average collected rewards by agent decreases.