CS7641 Assignment 4

Sahil Soni -ssoni41@gatech.edu

Markov Decision Processes

CS7641 Machine learning, Assignment 4 , MDPs, Q -Learning, Reinforcement learning

Abstract: MDPs, Q -Learning

Index Terms: MDPs, Q -Learning,reinforcement learning,Pymdptoolbox, Frozen Lake,Gym , Forest Management

1. Introduction

This is the fourth and final task for Spring 2020 as part of OMSCS CS7641(Machine Learning). We will discuss reinforcement learning algorithms in this phase. We will use Mdps and Value Iteration, Policy Iteration and Q learning algorithms. In this topic we will study two MDPs problems, one Grid World and one Non Grid World. Larger States do have to be one of the issues.

2. Data Sets

I have used Frozen Lake environment as Grid world problem which has 16 states (4X4) states . Figure 1 shows states values for Frozen Lake. Frozen lake problem is very interesting because it can be both Deterministic and sophisticated in nature. Also we can customized it to increased its states to 64 (8*8) or more even by using bigger Grid States 16*16(256) .There, S is the starting point, G is the target, F is the stable ice where the agent may stand and H is the space where if the agent goes down, it falls back. The agent has 4 potential motions that are described in the atmosphere as 0, 1, 2, 3 respectively for left, right, back, up.The water is mostly frozen, but there are a few holes where the ice has melted. When you walk into one of those places, you may plunge into the frozen cold. As the ice is unstable, and though, for example, we want to go right, we can slip and go up instead.Much of the lake remains solid but there are a few gaps where the ice has dissolved. When you walk into one of those places, you may plunge into the frozen cold. As the ice is unstable, and though, for example, we want to go right, we can slip and go up instead

For Second problem , i have chosen Forest management . Forest management is non-Grid World Problem . The problem of forest management is created from a finite state. Every condition consists of the same aging trees. The value of a stand being harvested and cut relies on the condition of the plants, that is, the age of the plants. Forest is governed through two 'Wait' and 'Strike' acts.An action is decided each year with first the objective to maintain an old forest for wildlife and second to make money selling cut wood. There is a chance p per year that a fire would destroy the forest. Let 'Pause' be action 0 and 'Cut' be action 1. The forest is at the youngest state after a storm, that is state 0.The incentive is achieved while the forest is in its oldest condition, and 'Wait' operation. Default: 4.Although when we described it when initializing the MDP, we may

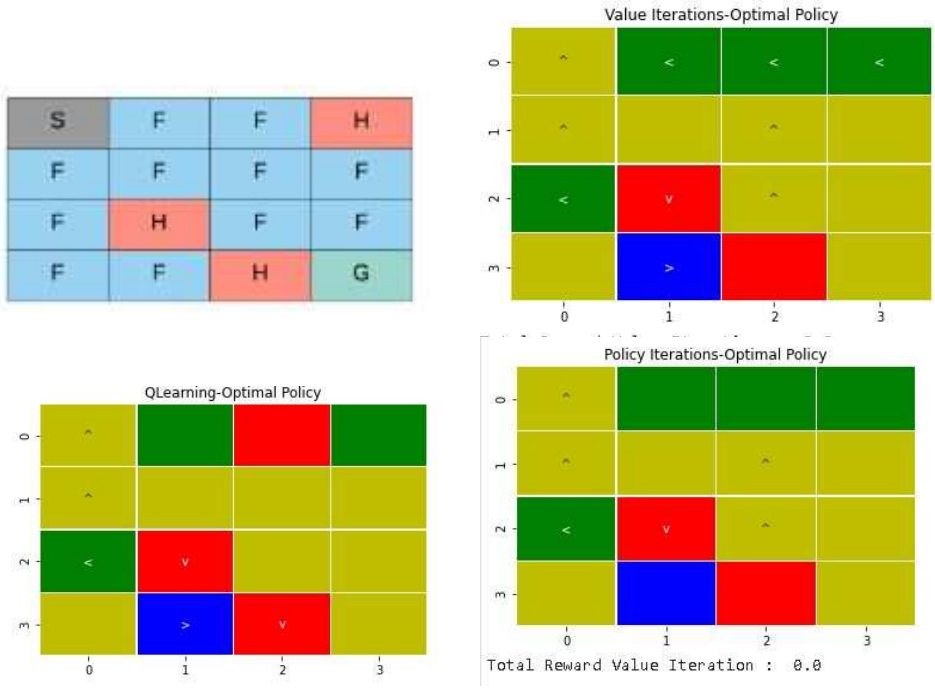

Fig. 1. Frozen Lake-Q

modify it. Below is example of such an mdp.

mdptoolbox.example.forest(P, r1=4, r2=2, p=0.1)

”P” is a Vector in 3 Dimensions. The first element is that of motion. The second aspect is the condition you are in at the moment and the third is the following. The meaning is the probability of you performing an intervention in the current setting, and the possibility of you landing in the defined next location.So for each current stage, P for action[0] is .1 to return you to the original state (because of fire). The operation P value [1] is 1 to restore you to the initial location, as you have opted to burn the forest down. ”R” is a 2 dimensional matrix where the first dimension is the state and the second dimension is the action taken at that state.It offers you the opportunity to be at the state and to take the practice. ”R1” is the incentive while the forest is in its oldest age, and Wait is done in practice. Default: 4.”R2 ”is the incentive while the forest is at its oldest age and Cu is done in practice. Parameters: 2. ”P” Probability of incidence of forest fire, within range [0, 1]. Defaults to 0.1.So Once a stand is cleared,new trees grow until the next harvest and so on. The probability of losing several stands is represented by the risk of fire occurrences.The forest management yields important incomes that can be optimized: the aim is to find an optimal strategy (also called policy) that maximizes the benefits while taking into account uncertainty of fire occurrence.

I have used default rewards and probability that each year there will be Fire and after that forest will be in young state and so we have to ”wait”. I have tried to solved it with larger states of 3000 .We can increased the states and probability along with rewards so this can be very small and Big problem as we want it. Most importantly this can be continues problem as there is no terminate state and after fire we have to wait and after ”wait” we have ”cut”.At same time we can defined terminate state to be ”cut” , in case after ”cut” land is not good for growing trees. So again we can customized it to non continues states. We will see when we run our analysis with high probabilty of ”fire” on the forest.

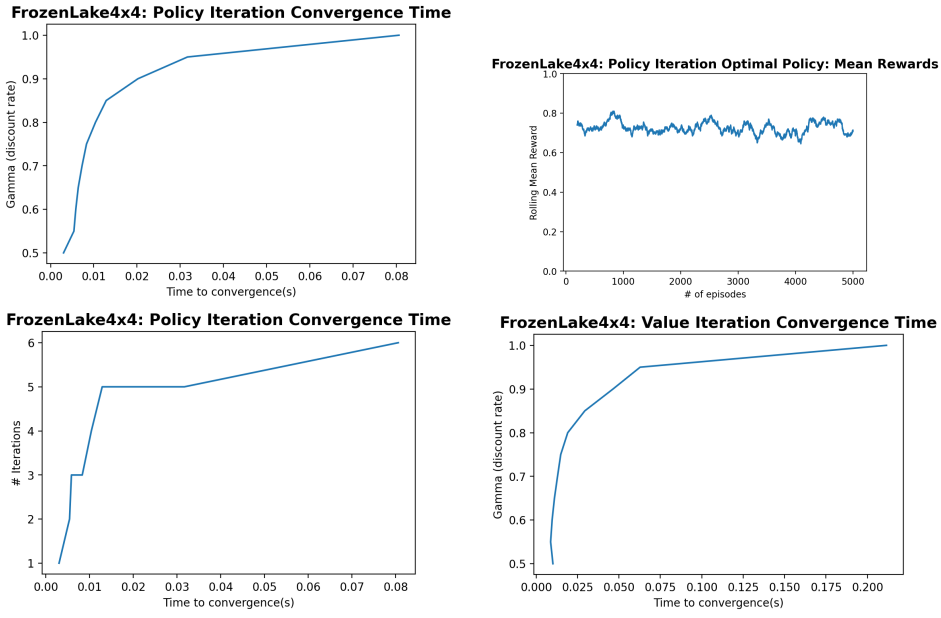

Fig. 2. Frozen Lake

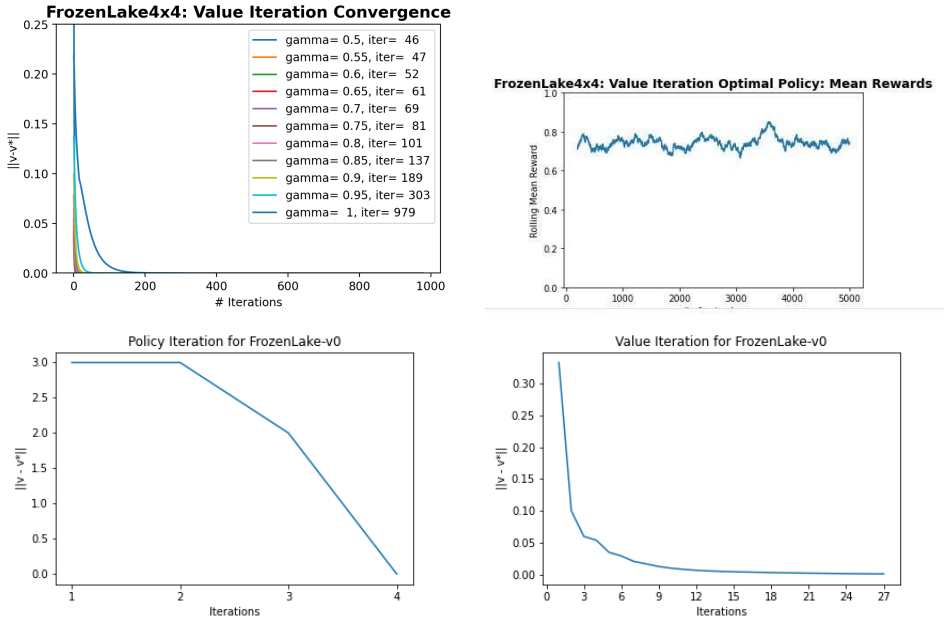

Fig. 3. Frozen Lake-Value Iterations

3. Data Acquisition and Tools used

I have used Open AI library along with Phton 3.7 . I have used Jupiter notebook and take help of Google cloud to run my experiments faster via Google colab .I have used Gym Library for Frozen Lake problem and pymdptoolbox for Forest Management problem .

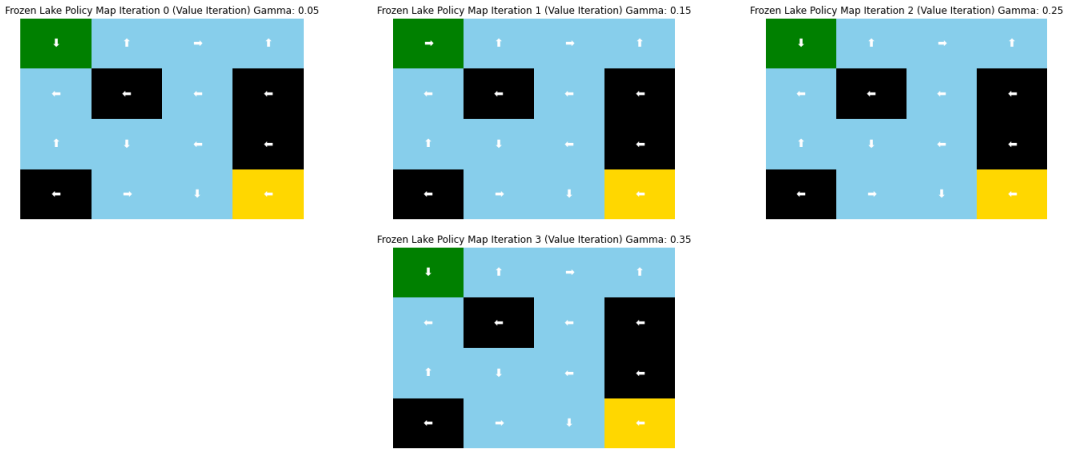

Fig. 4. Frozen Lake-Value Iterations

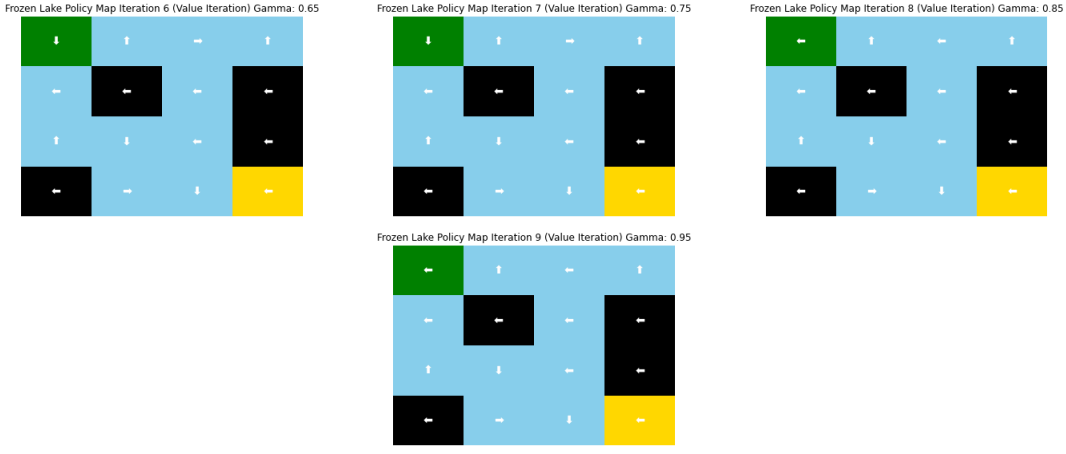

Fig. 5. Frozen Lake-Value Iterations

4. Policy Iteration

The policy iteration method explicitly manipulates the regulation rather than discovering it implicitly with the desired utility function. Regulation iteration requires policy assessment accompanied by policy enhancement.In policy assessment, a policy's value feature is only the estimated eternal discounted incentive that would be received by enforcing the policy at each level. Through solving a series of linear equations, it can be calculated. Here is a policy assessment equation.

$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} T\left(s, \pi_k(s), s'\right)\left[R\left(s, \pi_k(s), s'\right) + \gamma V_i^{\pi_k}\left(s'\right)\right]$

And on policy change, we decide how the interest may be enhanced by modifying the first step taken until we learn the importance of every state under the new policy. If it is necessary, we must change the protocol and take the new action if it is in it case. This move is intended to purely boost policy efficiency.If no changes are necessary, then the policy is assumed to be optimal.so To predict the evolution of the formula, the policy can only be modified if the new behavior for any state increases the predicted benefit.

Below is Equation of Policy improvement.

$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T\left(s, a, s'\right)\left[R\left(s, a, s'\right) + \gamma V^{\pi_k}\left(s'\right)\right]$

And simply government iteration, iterates through Policy Analysis and Policy Implementation until

it reaches the optimum policy iteration. Policy iteration is expected to converge and at equilibrium, the new policy and its value mechanism becomes the ideal policy and value mechanism!

5. Value Iteration

Where S is a finite set of states, A is a finite set of acts, P is the likelihood that behavior an in state s at the time t will lead to state s 'at the time t+1, R is the immediate reward (or predicted immediate reward) obtained following the change from state s to state s', due to behavior a.One downside of policy iteration is that each of the iterations requires policy evaluation, which may itself be a lengthy iterative calculation involving several sweeps across the state collection.In value iteration, then, we must truncate policy iteration measures to avoid policy evaluation after only one sweep (one backup for each state). It often converges even as the value function varies in a sweep by just a minimal number. Below is the value iteration convergence.

$\forall S \in S: \quad V^*(s) = \max_A \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$

Now run above value iterations until it converges. This produces V*, which in tells us optimal policy .

$\pi^*(s) = \arg\max_{a \in A} \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$

So as per theorem , Value iteration converges. At convergence, we have found the optimal value function V* for the discounted infinite horizon problem, which satisfies the Bellman equations

5.1. Policy/Value Iteration-Frozen Lake analysis

To understand the policy iteration and problem in detail , i have run experiments with deterministic and non-deterministic frozen lake on both (4X4) and (8X8) environment on both deterministic and non-deterministic environment. But for sake of simplicity , i have only shown graphs for (4X4) sophisticating environment. Here on figures we have shown results with (4*4) non-deterministic environment and where necessary to draw comparison also shown respective figures.

Gamma, Gamma is the discount element and is the most significant parameter here. It quantifies how relevant we are putting on potential incentives. Approximating the noise is often useful in potential incentives. Gamma ranges between 0 and 1 If Gamma is equal to 0, so the agent appears to only seek immediate bonuses. If Gamma is equal to one, the agent would find more weight for potential bonuses,, willing to delay the reward.

To determine convergence in my experiments on both value and policy converge i have compared the Old Policy to New policy with small Delta value (1e-40) . If There is less delta difference then we break the look and confirmed policy has converged.

From the figure 2, in terms of time analysis we can see that policy iteration took 0.08 seconds to converge at gamma 0.95. Compared to it with value iteration took more time around 0.20 seconds to converge at same gamma value of 0.95. So yes Value iterations is slow to converged compared to Policy iterations in terms of time analysis. So Now in terms of iterations , where policy iterations took 6 steps to converge with fixed gamma value of 0.95 , value iterations took 18 iterations to converge with gamma value of 0.1. Policy iterations took less iterations to converge as expected with optimized gamma values.

From Figure 3, we can see that in value iterations it took around 46 iterations to converge at 0.01 seconds at gamma 0.5. As gamma keep on increasing we can see that total no. of iterations also increased . At gamma 1 , it took around 979 iterations to converged at with time of 0.22 seconds.

Now Once we have optimal policy corresponding to optimized gamma value , now we will play the game with optimal policy to see how our agent behaves once it has trained. To show this , we plot optimal policy with rolling means rewards basically error when we run n iterations over the game , we can see that value iterations has average rewards of 0.75 and maximum it has touch sometime to 0.80 .Compared it to Policy iterations it has similar rewards but lesser. When we compare the raw data we found, we conclude although Both value and policy iterations converge to same policy but value iteration has a better average reward and higher number of wins compare

to policy iteration . So yeah , that's what we say on definition above , Value iterations overcame the drawbacks from Policy Iteration.

Also when we finally print , optimal policy for both Value and Policy iterations is same , as we can see on the figure 1. But there are big difference in times and no. of wins , time convergence .

When i run the same experiments on bigger states of frozen lake results were almost same but both algorithms took more time and more iterations to converge . This is expected as no. of states will increased agent has to go though more states although no. actions are fixed but total actions has been increased. So with increasing no. of states times and iterations for policy will increased . Again this is Grid world problem with sophistic environment.

In policy iterations , initially when gamma is low value 0.1 , agent need to consider immediate rewards . As it need to explore the environment and need to find where are the most rewards compare on the environment states. As agent keep on learning about the agent , we delayed the rewards and now we more worried about future that agent need to win the game based upon best state it has explore. This is where agent stop exploring the environment.As in policy iterations gamma keep on changing and ultimately it leads to optimal policy , we have shown on figure 4 and 5 how the gamma impact the policy mapping as no. of iterations keep on increasing .

Compare to it , but when we see similar gamma value impact on value iterations its totally different. As gamma value increased , it took more iterations and time to converge . Reason behind is that ,it starts with only with a random policy and it iteratively improved it. So in this case, agent need to explore more with in respect to same policy and it need to update its Q value . So it worry more about the immediate rewards and updates them to Q table with different states. But as gamma value increased , agent do not update Q values as much and do not explore the environment so it needs more iterations to converge . Even though , with different gamma values it converged to optimal policy but policy itself is different. So there are different optimal policies but different values. s

5.2. Policy/Value Iteration Analysis-Forest Management

I have run the experiments on Forest management with different states and probability of cutting the Trees.If the discount factor exceeds 1 to assess convergence, otherwise convergence can not be believed and a alert would be shown. So when we run our test we see our strategy converging after 12 iterations as gamma reaches 1.As we can see in the figure 5, with 3000 states and probability of 0.1 Policy iteration took 10 secs. In terms of rewards, when discount factor is less we have only rewards of 2 until gamma 0.7 and it increased after that to 10 when gamma is more than 0.85.In case of value iterations , it took around only 0.6 seconds to converge but it took around 50 iterations .Rewards to converge is around max 8.

So based upon this data we can conclude that , although policy and value iterations both converged to same policy both value iterations took more iterations to converged but took less time compared to Policy iterations . Also when no. of states has increased only 3000 to 4000 , time to converged has been jumped from 4 times for both value and policy iterations. Also when i increased states to 100000 my 12TB ram crashed. Also There are no changes observed in terms of rewards and iterations it took to converged. So i think if forest is big it will have millions of States and it will be iterative process , so someone need more powerful Machine and more time so this policy can be optimized. Also as long as we have probability in advanced when there will be ”fire” on forest on tree and we have to ”wait” there will be more rewards. But challenged come when there are unprecedented fire then we have to run our model again to get the new policy . In this case , states and rewards will changed .

Now i change the probability parameter from 0.1 to 0.5 and run the experiment with probability of fire increased from 0.01 to 0.05. I noticed it took more time to converged and it converged at gamma 0.4 compared to gamma around 0.9 with probability 0.01. So basically , as probability of fire increased forest will be on ”fire” little soon and we we have to quickly decide where to ”wait” or ”cut” . So actions will be increased and so the execution time . It might increased and decreased

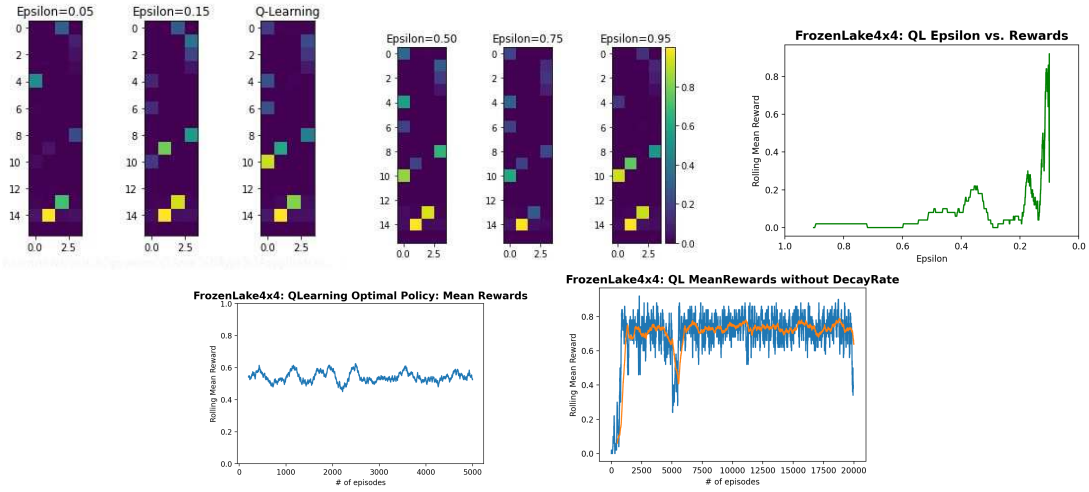

Fig. 6. Frozen Lake-Q

the rewards depends upon states and rewards and of course probability of fire.

Interesting , when we changed probability of fire in to high closed to 1, then there will be less rewards and it will converged sooner. . This confirmed when we see straight line for convergence analysis with probability of closed to 1. In terms of rewards , basically rewards decreased as probability of fire increased .

6. Q-Learning

Q-Learning is a simple method of reinforcement learning that utilizes Q-values (also known as behavior values) to iteratively develop the learning agent's behaviour. The free model algorithm to learn optimum strategy, asking agents what action to take on situations. Its is simplest form it can be defined as below :-

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

The Q-values for the states and acts are described. Q(S, A) is an approximation of how effective it is for behavior A to be taken in state S. This approximation of Q(S, A) will be iteratively determined using the TD- Update method that we can see in the following sections. An agent begins from a starting state over the span of his lifespan and allows a series of changes from his current state to a next state depending on his / her own choice of action and also the environment the agent is interacting in.ϵ-greedy strategy of is a very easy method of selecting behavior using existing Q-value estimates. Evaluation :- With likelihood (1-ϵ) select the action that has the lowest Q-value. With chance (ϵ) pick every action at random. What is important regarding Q-learning is that it converges to an optimum solution .But to figure out when the research starts, where the parameter of discovery does not rise slowly, Q-Learning converges prematurely.

6.1. Q Learning Analysis-Frozen lake

Q learning is offline strategy. In this case, we use the epsilon-greedy method to determine what action to take and simply choose the maximum action when updating the Q value. In this scenario, we start with Q learning algorithms with 20000 iterations and begin to train the agent with a constant value of ϵ. Finally when we plot rewards with iterations and we see consistent rewards after 18000 iterations we confirmed it has convergence. So although Q learning will sure find the optimal policy but how to determine convergence in this case is difficult. In our case we plot the graph to determine this. Please note compared to value/policy iterations we can not compare it to delta value because in Q-learning algorithms its action and Value pair of agents which update

the Q table and we determines best action and value value. Once we have Q table we determine optimal policy from this using Td updated value function .

As we can see in figure 6, when initially (ϵ) is high we have less rewards but when it keep on decreasing and agent has enough learnt about the environment it has maximum rewards around 0.1 .This is what we called exploration and exploitation . When ϵ is very high agent is exploring and it has high chance of falling on hole and slipper action . So rewards are less initially . But once agent has learn enough about the environment we decreased the ϵ and rewards keep on increasing . And as ϵ approaches to 0 we have highest rewards. That's where we stop the exploration and exploit the environment for maximum reward.

When we have decay plot rewards against without decay rate which means agent is not selecting random action and it always take action based upon epsilon . So out agents is not exploring the environment randomly . Sometime we have very high rewards and sometime it goes to low. Rewards are fluctuating. Compare when we see optimal policy rewards with decay scheduled it has consistent rewards . It basically concludes, if agent is not randomly exploring the environment it will only some "known" state, so our policy might be over fitting . Which conclude over above argument. But once we only explore the environment with decay rate and randomly explore the environment and we can see the agent is learning well . As we are learning off-policy, then increasing the exploration rate may help find the better states, at the expense of slower overall learning. Also, methods that explore more widely than randomly on each action could be better -e.g. action selection methods that consistently pick unexplored state/action pairs such as Upper Confidence Bound.

In figure 6, we see Q tables heat map for different epsilon value . We can see with epsilon rate 0.5 and 0.15 there is less learning but as epsilon rate increase agent learned most of environment. so basically when agent randomly explore more it start learning when higher epsilon value. But as we have higher epsilon agent has more chance of slippery and finding a goal so we need to run many episodes to learn about optimal policy .The explanation why decay is recommended in learning speeds is that initially when we are at a totally random point in solution space we need to take huge strides towards the answer and then as we get close to it we make tiny hops and hence minor changes to eventually achieve the target.But there is finally no generalized plan that would extend to all environments and be equally successful in them. You will need to conduct a check for potential decline schedules for an appropriate solution, so the most effective decline in the learning rate will only refer to the setting you have studied.It will actually extend well enough to related situations to realize that in the future. So Q learning Off policy is easier than On Learning (Sarsa) without the risk of loss as On Policy is likely to overfit but has large incentives.

When i run my experiment with same epsilon decay schedule , but different learning parameters i found , when learning rate is too small my algorithm converges too slow but when learning rate is too high it converges high . So i think let's only used learning rate 1. But later i found when high learning rate only good for deterministic environment of Frozen lake. Reason it was replacing each new value with a new latest values and it works just fine because states are deterministic and out goal is just to predict higher accuracy . So high learning rate good for smaller deterministic values where our goal is more higher accurate predictions. But in case of Stochastic , no state has higher accuracy so high learning rate will replace all current values new Guess but if i have too low learning rate then my agent will always trust the current value. So i think low learning rate matter more on case of neutral network with Q learning. In our case, learning rate to be just low enough that inaccuracies due to over/undershooting the correct value don't prevent or delay differentiating between actions for whatever the interim policy is.

Ultimately, there is no general method, learning rate and epsilon values that all depend on the environment. You will need to check for alternative decline plans and an effective solution, so the most productive decline in the learning rate will only refer to the setting you evaluated.

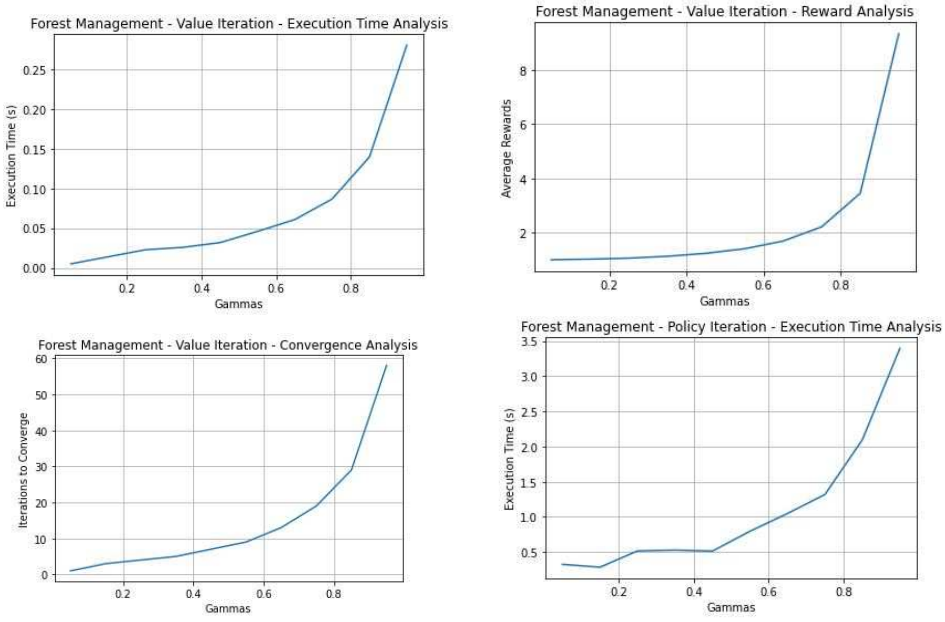

Fig. 7. Forest Management

6.2. Q Learning Analysis-Forest Management

In q learning, consistency is thus calculated when the maximal value feature shift is compared to epsilon at each iteration. If the shift falls below this point, then it is assumed that the point function has transformed to the optimum value function. As we can see in the figure of difference epsilon values nearly 8000 iterations, it begins converging and almost almost at 10000 iterations. We have received more rewards with epsilon 0.25 . Figure 8 shows show selected Q-Tables for the greedy approach (rows are states, columns are actions).As we see when we do less exploration with epsilon value 0.15 we have less rewards. Even with more exploration on 0.95 epsilon we rewards although bigger . But on ideal epsilon value 0.25 we have maximum rewards.

When we increased probability of ”fire” slowly then agent need to explore more states and so it will took more time. I think rewards will be increasing to start increased up to some probability and when we have higher probability of fire then rewards will be decreasing for same states because then there will be less ”Wait” and ”cut” and hence less rewards.

7. Conclusion

So essentially value and regulation variations converge to the same regulation except with various gamma values there may be several optimal policies. We see decreased execution time for grid problems and non-grid problems as described. Random exploration of the world is necessary in Q-Learning because if we do not explore the world we might be over-fitting. For Q, several ideal Q tables can be found based upon learning rate and epsilon decay gamma scheduled.

References

http://incompleteideas.net/book/bookdraft2017nov5.pdf
https://people.eecs.berkeley.edu/ pabbeel/cs287-fa12/slides/mdps-exact-methods.pdf
https://stackoverflow.com/questions/33011825/learning-rate-of-a-q-learning-agent

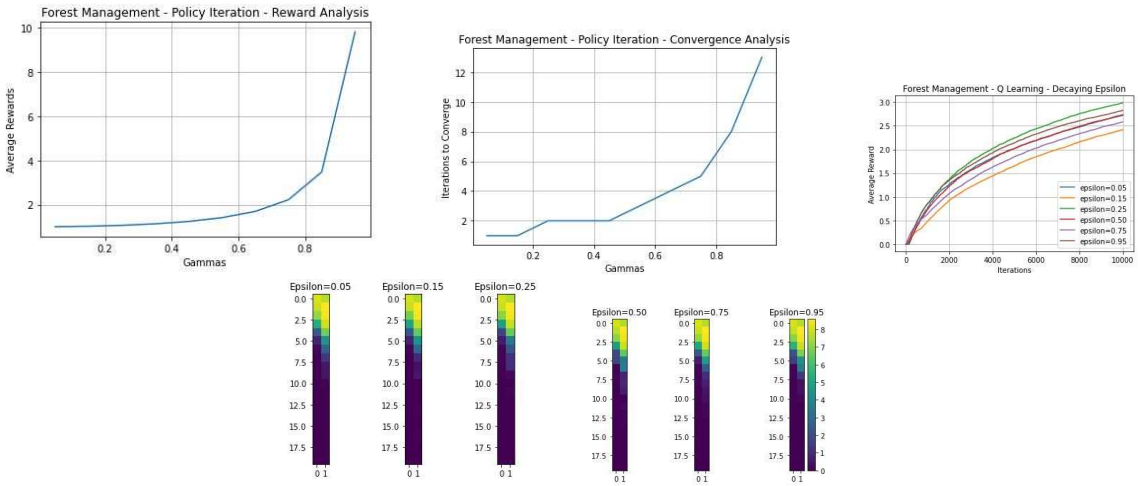

Fig. 8. Forest Management