Markov Decision Processes

Younes EL BOUZEKRAOUI
ybouzekraoui3@gatech.edu

***Abstract*—**The Purpose of this project is to explore Markov Decision Processes, create two interesting MDPs, solve them using value iteration , policy iteration as well as Q-learning, compare them and analyse how each method converges, how it performs and how their parameters affects the solution.

1 TOOLS AND ENVIRONMENT

The programming language used in this assignment is Python with Numpy, Matplotlib, Pandas, Seaborn, time libraries.

We used also gym library of OpenAi and The mdptoolbox library that provides classes and functions for the resolution of descrete-time Markov Decision Processes.

The programming environment is Jupyter Notebook, and the code is available in a jupyter-notebook format.

2 ENVIRONMENTS AND PROBLEMS

2.1 MDP1 : Frozen Lake

The first Markov Decision Problem I choose is a grid problem called Frozen Lake. It's a 4x4 grid containing a beginning cell (S), some frozen cells (F) indicating a safe position, some holes (H) to avoid and a goal cell (G) to achieve. Any agent must begin at S and navigate to G through F while avoiding H. The agent's actions are : UP DOWN LEFT and RIGHT and generates deterministic transitions that perform as expected. If the agent gets to G, he will get 1 point; else, he will get 0.

Moreover we are in a stochastic world (Slippery lake) so the agent will move in intended direction with probability of $1/3$ else he will move in either perpendicular direction with equal probability of $1/3$ in both directions.
For example, if action is left then:
- P(move left)=$1/3$
- P(move up)=$1/3$
- P(move down)=$1/3$

The grid world is an excellent abstraction of various real-world applications; the agent must make a series of decisions to arrive to his objective quicker, and the obstacles (holes) may be viewed as real-world constraints.

For example, while using GPS in a car to plan a route, we pick the beginning location and target and wish to figure out what is the optimal path taking into consideration all the obstacles (closed roads, traffic jam) and expenses.

As a result, grid world challenges are interesting to investigate since they are a simulation of a real-world situation.

2.2 MDP2 : Forest management

The second Markov decision problem is Forest Management environment, which is a non grid problem

3 ALGORITHMS DESCRIPTION

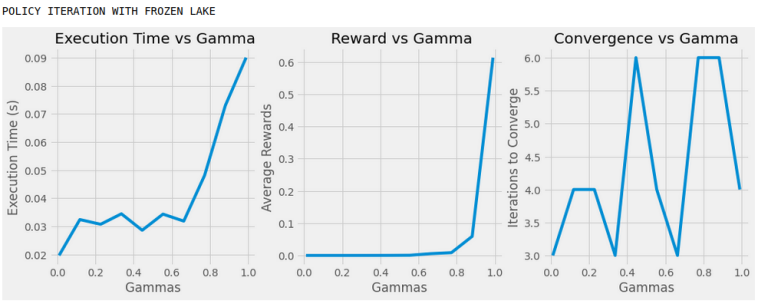
- **Value Iteration :** Evaluates state-by-state until the answer converges using the Bellman equation. Except for the immediate reward utility, the utility is unknown at first. The utility of the closest states is determined based on discounted reward using the immediate reward at the terminal state, and this iterates until all states' utility is known.
- **Policy Iteration :** Starting with a random policy, Policy Iteration involves two steps: evaluating the policy and then improving it after some iterations. We start with some random values for all the states in the MDPs in the policy evaluation step, then update the value of each state by taking the expected value across all potential actions if the agent follows the provided policy, weighted by the probability of ending up in each future state. This method also converges, however requires more calculation as it solves a set of linear equations when determining the expected value of all states.
- **Q-learning :** Unlike the two previous methods where we use domain knowledge to calculate the transition function , Q-learning is model free . after assigning initial q values for the states, The agent keeps visiting again these states and updating the q values based on reward (immediate and delayed). Q-learning can use randomness to select policies at the beginning by exploring more but as it progresses, the agent will exploit more and the q values will converge.

4 FROZEN LAKE

4.1 Policy Iteration :

We applied the Policy Iteration algorithms described above to our MDP.

The discount factor is a parameter that tells how important future rewards are to the current state. In order to see how this parameter affects the things , we tried to hyper-tune it and compare the results in regards to the Execution Time , Average Reward and Convergence.



The reward curves in regards to Gamma shows a Null reward for values of gamma less than 0.7 however the reward reaches a high value for higher gammas (>0.8) this means that for lower discount factors the cannot find a good policy so this results in small reward however a higher discount rate gives a good reward 0.6 thus a better policy

For the execution time , a higher gamma results in a high execution time so the discount rate highly affects the execution time, in our case the state space is very low (16) so this should not be a problem but in other cases the execution time might be a real struggle

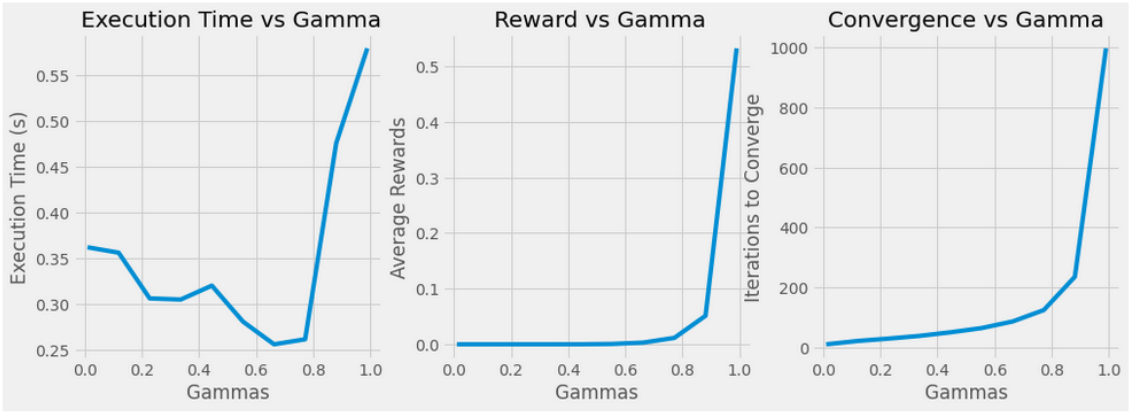
In terms of Iteration of convergence Policy Iteration is always converging and we can see that the curve is not showing a specific trend, maybe this is because of the small state space that we are in.

4.2 Value Iteration :

We applied the Value Iteration algorithm described above to our MDP.

For the analysis we apply the same methodology as previously, we hyper-tune the discount factor and see how it affects the Execution Time, The Rewards and The number of iterations needed to converge.

VALUE ITERATION WITH FROZEN LAKE



As we can see in the figure above , the reward curve have the same shape as in Policy iteration, smaller discount rates cannot assign the optimal values an thus an optimal policy, however a higher discount rate results in an average reward of 0.52 which is less the reward we got in Policy iteration for the same gamma .

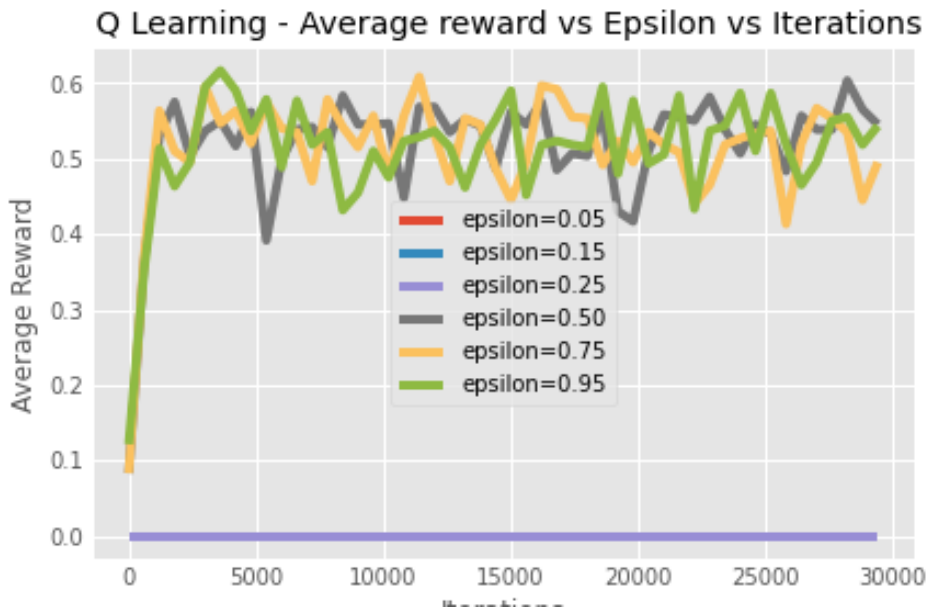
Convergence time is following a decaying trend for discount rates from 0 to 0.7 an then starts increasing to high values.

It seems logic to use higher discount rates in this case, considering it's positive effect on the reward and the policy even if it results in higher execution time because as mentioned before we are still in a small state space and 0.55 is still a very good execution time.

In terms of convergence Value Iteration is always converging and we can see from the figure the exponential trend of the number of iteration to converge against Gamma , higher values of gamma makes the algorithm in need of more iterations to converge because it will be taking more into account the delayed reward and not only the imediate rewards.

4.3 Q-learning :

In order to Visualize the convergence using the Q-Learning algorithms and see how epsilon value affects the results, I plotted the Average Reward in regards to the number of iterations for multiple values of epsilon (0.05 , 0.15 , 0.25 , 0.5 , 0.75 , 0.95) we used a fixed value for the discount rate 0.95. Epsilon represents the exploration - exploitation rate , a lower epsilon value means more exploration and vice versa the following figure shows the resulting curves:

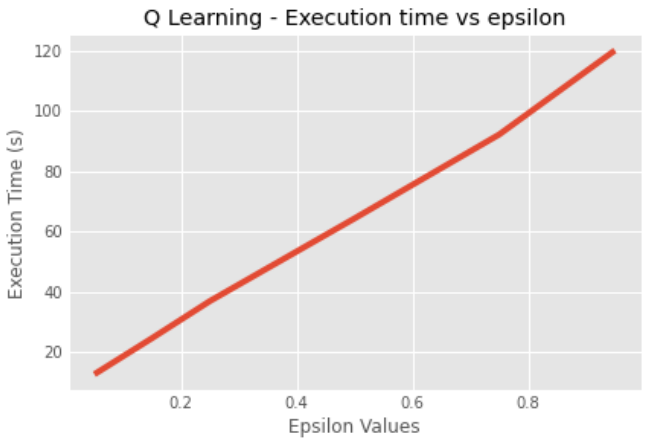


The resulting curves shows that q learning does not converge , maybe a stabilisation and fluctuation around a mean of 0.55 can be seen after about 2000 iterations only for higher values of epsilon however we can see than for lower epsilon which means less exploration the curves are at 0 .

The fluctuation can be resulting from the decay rate of epsilon , we need to use a higher decay to tell the algorithms to do less exploring when it starts learning a policy

This shows the effect of the exploration rate on finding a good policy, if we select pure greedy method (epsilon = 0) then we are always selecting the highest q value among the all the q values for a specific state. This causes issues in exploration as we can get stuck easily at a local optima.

To evaluate the Q-learning algorithm in terms of Time, I plotted the execution Time for multiple values of epsilon.



The Figure shows a growing curve for the execution Time against Epsilon values, the curve tends to be linear and we can see that higher values of epsilon results in a higher execution time, this makes sense because for a high epsilon we are performing a lot of exploration so it makes time but in return we are more likely to find the optimal policy.

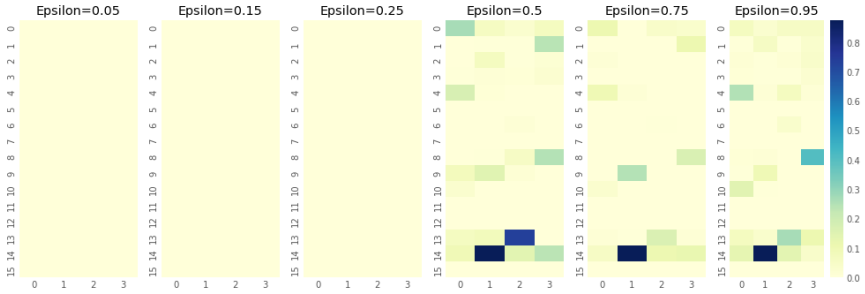


Figure 1—Final Q Table heat map vs Epsilon

The Figure above shows the final Q table as a heat map for different values of epsilon, as we can see for lower epsilon we were not able at all to learn a policy however the higher is the epsilon the more we tend to learn the optimal policy , which the most right heat map.

5 FOREST MANAGEMENT

The Forest Management environment is a non-grid world with a forest that grows. Two actions are used to manage a forest: 'Wait' and 'Cut.' Each year, an action is chosen with the goals of preserving an old forest for animals and making money by selling

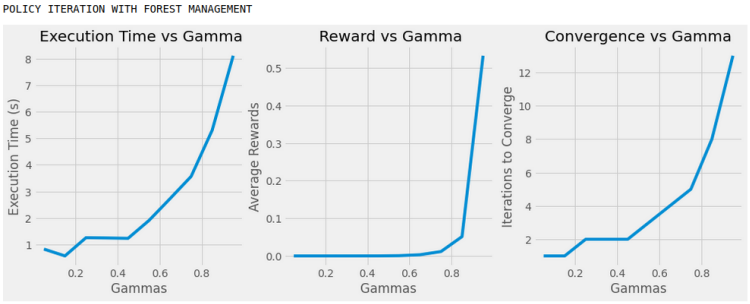
chopped wood. Every year, there is a p chance that a forest fire may burn (and 1-p that the forest grows).

The agent has two actions in each step: CUT or WAIT; if CUT, it receives one reward and the forest is moved to state 0, where it cannot be CUT. Because of a forest fire, if the agent uses the action WAIT, it moves to the next-state with probability p and returns to state 0 with (1-p). As a result, the agent must decide whether it wants to risk travelling to the end stage in order to receive greater payouts or cut the forest earlier to receive a single reward.

We used for this problem a number of states of 5000 to demonstrate and explore a large state space.

5.1 Policy Iteration :

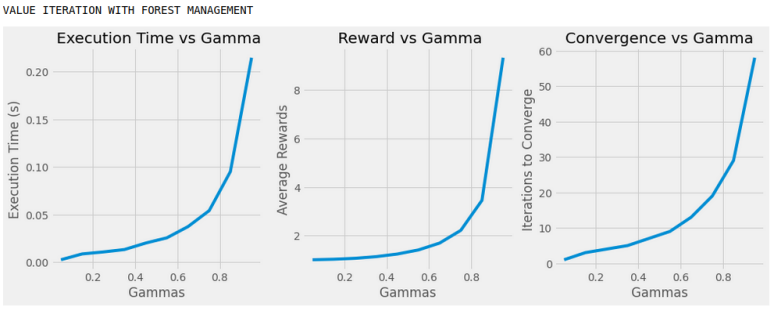
Using the mdptoolbox package we applied the Policy Iteration algorithms to our MPD. In order to compare the results with the Frozen lake MPD, we plotted to Execution Time, The Average reward and the number of iteration to Converge against Gamma the discount factor, to see if there are some changes when we are in a large state space.



From the figures above we can see relatively the same curve shapes as in the Frozen Lake, The Execution time is growing exponentially as the discount factor grows however in this MDP with larger state space the time is about 8 second for discount factor 0.95 , so we need to start taking into account the time factor.

The discount factor also affect the rewards in this case , for smaller discount rates the algorithms was not able to learn and find a good policy however for a discount rate of 0.95 we got a good reward, this makes sense because also in this MDP we need to take into account the delayed reward (what will happen in the future) to create a good policy

5.2 Value Iteration :



The curves are also having the same shape as before , however this time with value iteration we are taking less Execution time and getting bigger reward in comparison with policy iteration.

In terms of convergence both Policy Iteration and Value Iteration are converging however Value Iteration is taking more iterations but in less time and with higher return.

5.3 Q-learning :

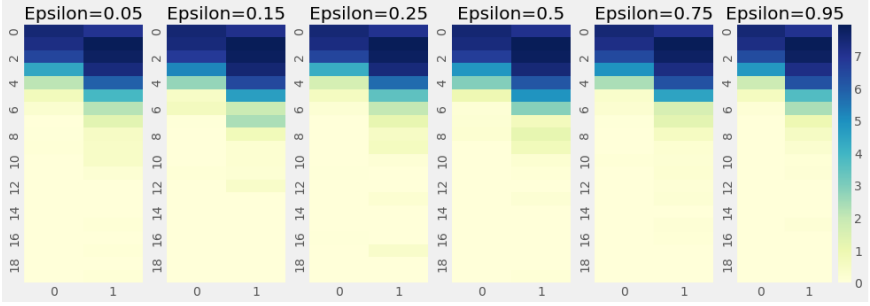


Figure 2—Final Q Table heat map vs Epsilon

As we can see from the figure above the Q learning algorithm is not converging even with high epsilon and a max iteration of 10 000, this is due to the dimension of the state space (5000 states) so Q-learning was not able to learn the optimal policy in this configuration. Maybe with more iterations (Millions), that will potentially take more time (hours) the algorithm might learn something good and might converge but in general it is not practical in term of time execution.

6 CONCLUSION

Sum up, when the reward and transition functions are available, both Policy Iteration and Value Iteration work very well on simple environement (<100 of states). For big problems (more than thousands of states), Policy Iteration seems to converge in less iterations than Value Iteration, but it takes longer time for execution.

Because of the Forest Management environment's chained-state structure, we can observe that the grid world is simpler to converge for Policy Iteration when compared to non-grid world situations.

In terms of time and iterations to converge, Value and Policy iteration are considerably superior to Q Learning. However, the reward and transition functions are unlikely to be provided in most applications.

As a result, in the vast majority of circumstances, Q Learning (and other Reinforcement Learning algorithms) is the only viable alternative. Rather of being provided these functions, Q Learning explores the environment to obtain a sense of the rewards and transitions, continually updating the values it estimates for each policy.

As I've demonstrated, there are several hyperparameters that I required to fine-tune using Q Learning. These hyperparameters are frequently domain-specific, which implies that values that might work well in the the first MDP Frozen Lake may not work in the Forest Management MDP, and vice versa.

"Artificial Intelligence, deep learning,machine learning, whatever you're doing if you don't understand it, learn it. Because otherwise you're going to be a dinosaur within 3 years. "

–Mark Cuban–