November 24, 2019 (F19)
Md Khaled Hassan (ID: mhassan49)
Georgia Institute of Technology

Markov Decision Processes

CS7641: Machine Learning (Assignment 04)

Abstract:

The objective of this assignment is to explore two interesting Markov Decision Processes (MDP) and solve them using three different reinforcement learning algorithms. I picked up two grid world problems from OpenAI Gym (https://gym.openai.com/): 1) Frozen Lake (16 states) and 2) Taxi (500 states). I have solved these two problems with three different algorithms: 1) Value Iteration, 2) Policy Iteration, and 3) Q-Learning (ϵ-greedy). In the Introduction section, I have briefly explained MDP, the two MDPs that I explored here, and the three algorithms that I applied to these MDP problems. In section II and III, I have discussed the simulations results of FrozenLake and Taxi MDPs, respectively. The conclusion section summarizes all the key learnings from this assignment.

I. Introduction

Markov Decision Process (MDP):

In a Markov Decision Process (MDP), an agent can perceive the distinct states of its environment (a set S) and has a set of actions (A) that it can perform. The agent senses the current state s_t (at a discrete time step t), chooses and performs a current action a_t. The environment responds by giving the agent a reward $r_t = r(s_t, a_t)$ and produces the succeeding state $S_{t+1} = \delta(s_t, a_t)$. Here, the functions δ and r are part of the environment, not necessarily known to the agent, and depend only on the current state and action. The task of the agent is to learn a policy π (the solution to the MDP), which is a mapping from S to A ($\pi : S -> A$). A discount factor γ controls the importance of future rewards. A value function $V^{\pi}(s_t)$) determines how good is a state for an agent to be in[1-2].

Two Interesting MDPs:

The first MDP problem that we explored is *Frozen Lake-v0* [3]. This is an easy problem with 4 actions and 16 states. The problem is interesting since it represents a real life problem of finding a safe path in a potential hazardous location. In this problem, an agent controls the movement of a character in a grid world where some tiles are walkable (ice) while others can lead to falling into the water and the agent is rewarded for successfully finding the destination (goal

tile) [3]. The second problem that I picked up and which is much harder than the first one is *Taxi-v3* [4] with 6 actions and 500 states. This problem is also interesting since it also involves a real world problem where a taxi driver need to pick up and drop off a passenger as fast as possible. The agent also gets penalized for any illegal pick-up and drop-off actions [4].

Three Reinforcement Learning Algorithms:

Three different algorithms are applied to solve the two above mentioned MDP problems. They are briefly explained below.

1. Value Iteration: In value iteration, the value function is iteratively improved using *Bellman equation* until the algorithm converges to the optimal value. Then the corresponding optimal policy is extracted. The simulations starts with a random value function and the iteration is guaranteed to converge to the optimal value [2, 9].
2. Policy Integration: In this algorithm, the policy is redefined in each iteration (evaluates and improves the policy) and compute the value according to the new policy until the policy converges. This algorithm is also guaranteed to converge to the optimal policy and often take less iterations than the value iteration algorithm [2, 9].
3. Q-Learning (ϵ-greedy): Q-Learning is a model free algorithm where the agent discovers the good and bad actions by trial and error. The algorithm approximates the state-action pairs from the samples of $Q(s,a)$) that we observe during interaction with the environment.This approach is known a time-difference learning.

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha Q_{obs}(s,a)$$

Here, α is the learning rate and the $Q(s,a)$ table is initialized randomly. In the greedy approach, the optimal action is taken by simply choosing the action that is expected to provide greatest reward. In other words, the agent exploits the current knowledge about the reward structure of the given environment [13]. In the contrary, a random approach always picks up an action randomly [13]. ϵ-greedy is a combination of both greedy and random approach. ϵ is an adjustable parameter that can be as high as 1 (meaning the agent is 100% greedy) at the beginning and decays over time as the agent becomes more confident with the estimate of Q values. The agent chooses a small random probability and compares with ϵ. If the the value is greater than ϵ, the agent chooses the action from the Q-table. Otherwise, it randomly picks up an action [2, 13, 14].

Coding and Implementation:

Various online resources are explored, combined, and modified to solve these two problems using the gym library in python [5-8]. Unless otherwise specified, γ (discount factor)=0.9, ϵ=1, ϵ-decay=0.99995, and α (learning rate)=0.6 are used for the simulations. The mean iteration

numbers for Q-Learning algorithm listed in Tables 1 and 2 are approximated value to the nearest higher integer.

II. Analyses and Discussion: FrozenLake MDP

In the FrozenLake problem, the surface is described using a grid like the following: An episode ends when the agent reaches the goal or falls in a hole. A reward of 1 is received if the agent reaches the goal, and zero otherwise [3].

|S|F|F|F|	(S: starting point, safe)
|F|H|F|H|	(F: frozen surface, safe)
|F|F|F|H|	(H: hole, fall to your doom)
|H|F|F|G|	(G: goal, where the target is located)

Value Iteration:

In value iteration, the values are initialized using immediate rewards and updated in the successive iterations based on the best next state [11]. The process is repeated until convergence, meaning the difference between values in two successive iteration changes by a value less than or equal to δ. In our simulations, δ was set to 10^{-3}. Fig.1 shows the reward versus episodes for two different discount factor, γ (0.90 and 0.50) for a total number of episode of 100. The discount factor is a measure of how much the agent care about short-term or immediate rewards over long-term rewards. A lower value of γ means the agent is more interested in short term reward. We observe that the environment is more rewarding (higher average value over the total number of episodes) with larger value of γ. This makes sense because, in the Frozen Lake problem, the agent only gets rewarded for successfully reaching the goal and no point is awarded for the intermediate steps. Also as expected, the simulation converges much faster (Table 1) when γ is smaller.

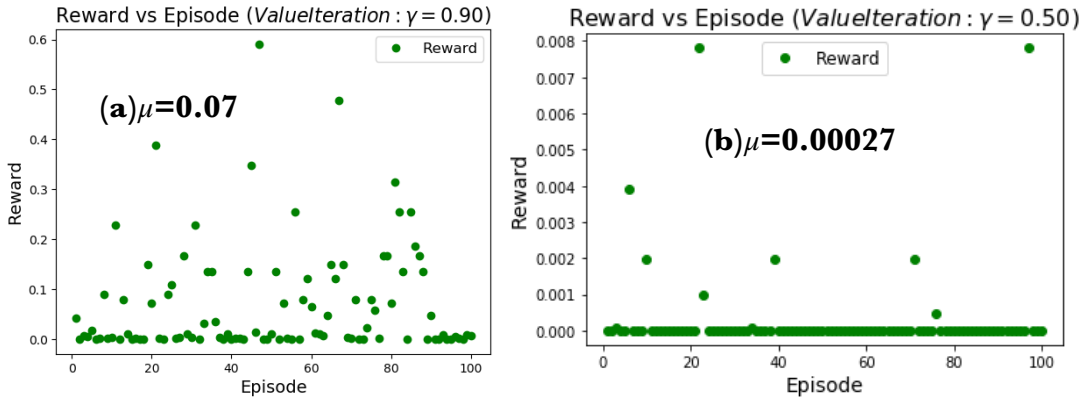


Fig.1: FrozenLake-Value Iteration (a) Discount factor=0.9, (b) Discount factor=0.50

Policy Integration:

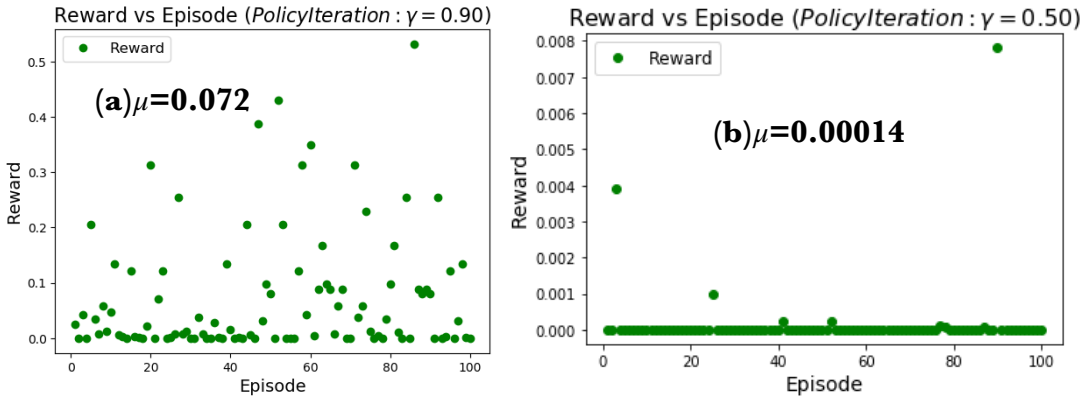


Fig.2: FrozenLake-Policy Iteration (a) Discount factor=0.9, (b) Discount factor=0.50

In policy iteration, the policy is directly modified instead of relying on finding the optimal value [12]. The corresponding value functions for policies in successive iterations are compared and a convergence is reached if the difference is less than or equal to δ. In our simulations, δ was set to 10^{-3} and the simulations were run for 100 episodes, same as the value iteration algorithm. As far as the discount factor (γ) is concerned, we observe similar trend, lower discount factor is less rewarding as shown in the Fig. 2. This is simply because the Frozen Lake environment is rewarding only if the agent reaches the final destination and therefore, higher average reward is observed for higher discount factor The convergence time is also faster for smaller γ (Table 1).

Q-Learning (ϵ-greedy):

We have explored the ϵ-greedy approach using different initial ϵ value and learning rate (α). The learning rate determines to what extent the latest information overrides the old information. A learning rate 0 means the agent only exploits the prior knowledge while a rate 1 means the agent only considers the latest acquired knowledge. I have simulated three sets of parameters (two different ϵ and α parameters) and plotted them in Fig. 3. The simulations were run for 100000 episodes and maximum iteration was set to 1000. The upper panel (a, c, and e) plots the reward (score) and the lower panel (b, d, f) plots the convergence time in each of the episodes. Column 1 and 2 compares the effect of ϵ. Higher ϵ means higher random exploration rather than selecting the best action. Therefore, the average reward over 100000 episodes is higher for lower value of ϵ. In addition, the higher ϵ value converges faster meaning it is easier for the agent

to explore better policy . Column 2 and 3 compares the effect of different learning rates (α). Since higher α means the agent is more considerate to the latest knowledge rather than the prior or best available knowledge, the average reward is also smaller. The average convergence time is higher for higher learning rate indicating that improvement in each iteration is relatively worse. However, each of the above three parameter sets learns the best option for this environment. In addition, the convergence time remains same for most of the episodes since the algorithm simply explores or exploits in each of the iterations and there are only two states in terms of rewards.

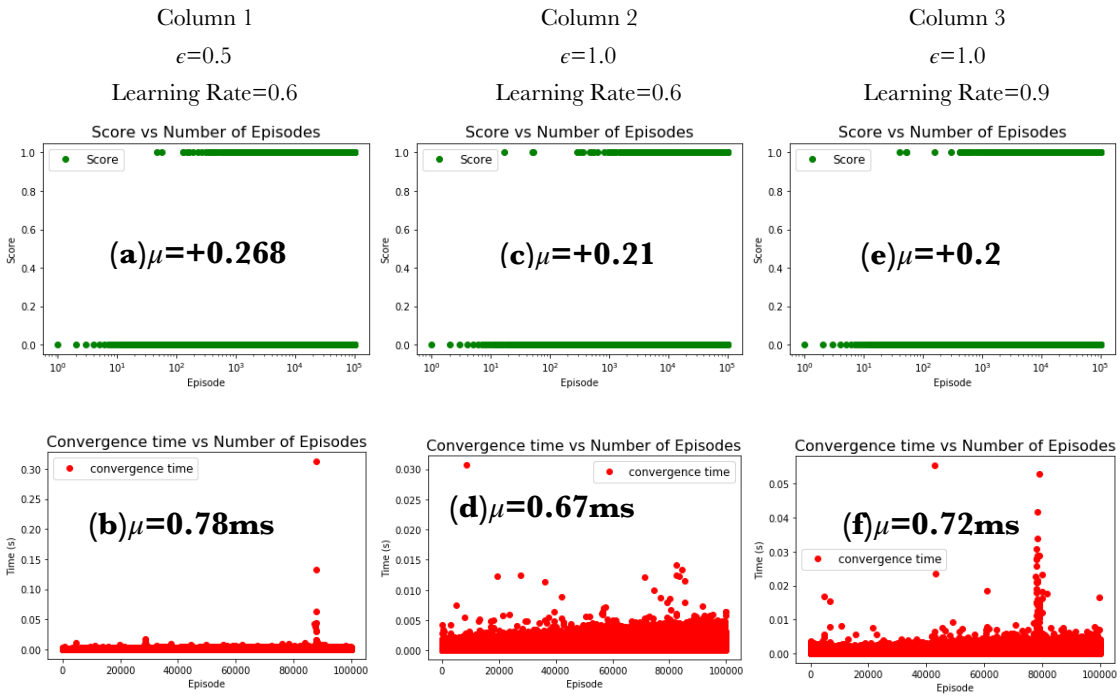


Fig.3: FrozenLake-Q-Learning : (a), (c), and (e) score vs Episode (b),(d), (f) convergence time vs Episode

Comparison:

Table 1 summarizes the performance of all three algorithms. We observe that both policy and value iteration converges to the same policy and performs worse than Q-learning in terms of rewards. Although the Q-learning takes relatively longer time to learn, convergence time in each episode is much faster compared to the value or policy iteration. Between, value and policy iteration algorithms, policy iteration achieves slightly better score (γ=0.9) with a much smaller iterations. Since the policy iteration calculates policy in each iteration algorithm is slower compared to value iteration (obvious from the results from γ=0.5). The rewards are very small for

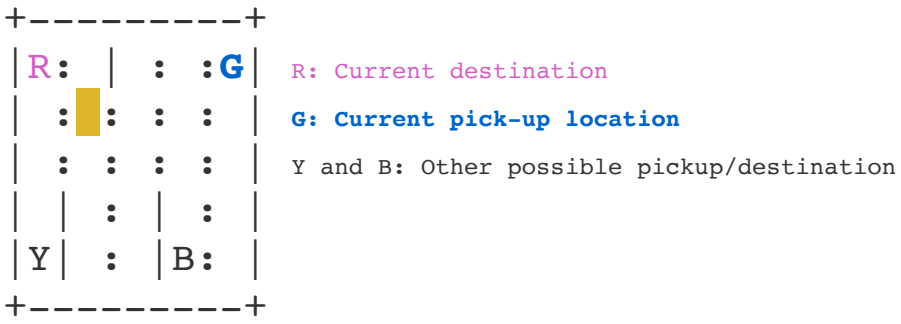
both algorithms when γ=0.5. I am not sure if a comparison in terms of rewards is justified in this case.

(Table.1: Comparison of three algorithms (FrozenLake-v0)

Algorithms	Total Episodes	Reward (mean/ episode)	Number of Iterations	Convergence Time (ms)
Value Iteration (γ=0.90)	**100**	**0.07**	**27**	**4.45**
Value Iteration (γ=0.50)	**100**	**0.00027**	**7**	**1.58**
Policy Iteration (γ=0.90)	**100**	**0.073**	**3**	**3.64**
Policy Iteration (γ=0.50)	**100**	**0.00014**	**3**	**2.61**
Q-Learning (ϵ=0.5, Learning Rate=0.6)	**100000**	**0.268**	**22 (mean)**	**0.78 (mean)**
Q-Learning (ϵ=1.0, Learning Rate=0.6)	**100000**	**0.21**	**19 (mean)**	**0.67 (mean)**
Q-Learning (ϵ=1.0, Learning Rate=0.9)	**100000**	**0.202**	**20 (mean)**	**0.72 (mean)**

III. Analyses and Discussion: Taxi MDP

In the Taxi problem, the simulation domain can be defined as a 5x5 grid with four possible locations (R, G, Y, and B) where the taxi can pick up or drop off. An agent can take six possible actions: can pick-up or drop off a passenger or move to a direction (north, south, east, or west) [10]. The agent receives +20 points for a successful drop-off, and loses 1 point for every time step it takes. In addition, a 10 point penalty is imposed for illegal pick-up and drop-off actions [4].



Value Iteration:

I did similar analysis as that of the Frozen Lake environment with the same parameter settings. We observe from Fig.4 that the environment is more rewarding (higher average value over the total number of episodes) with larger value of γ, a conclusion similar to the Frozen Lake problem. This is because, the agent only gets rewarded for successfully reaching the goal and

negative point is awarded for the intermediate steps. Also as expected, the simulation converges faster (Table 2) when γ is smaller.

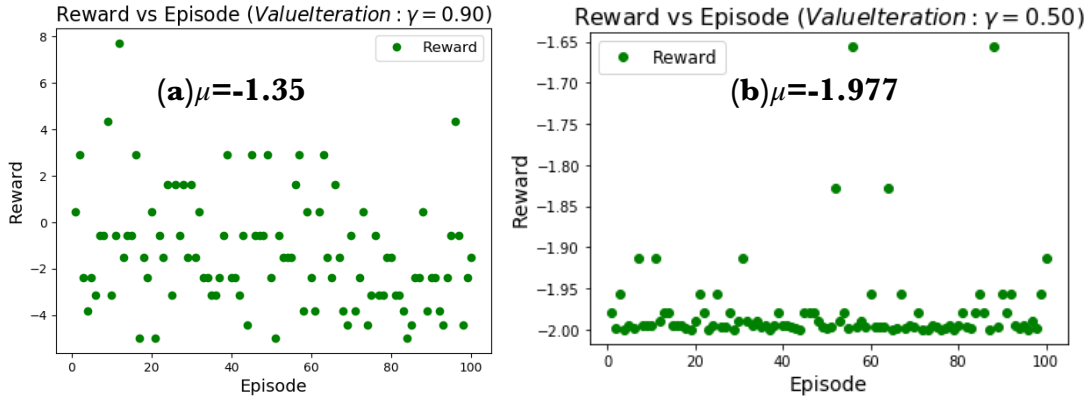


Fig.4: Taxi-Value Iteration (a) Discount factor=0.9, (b) Discount factor=0.50

Policy Iteration:

Similar analysis was carried out with same parameter settings as that of the Frozen Lake environment. We observe similar trend, lower discount factor is less rewarding as shown in the Fig. 5. This is because the Taxi environment only rewards the agent for a successful drop-off.

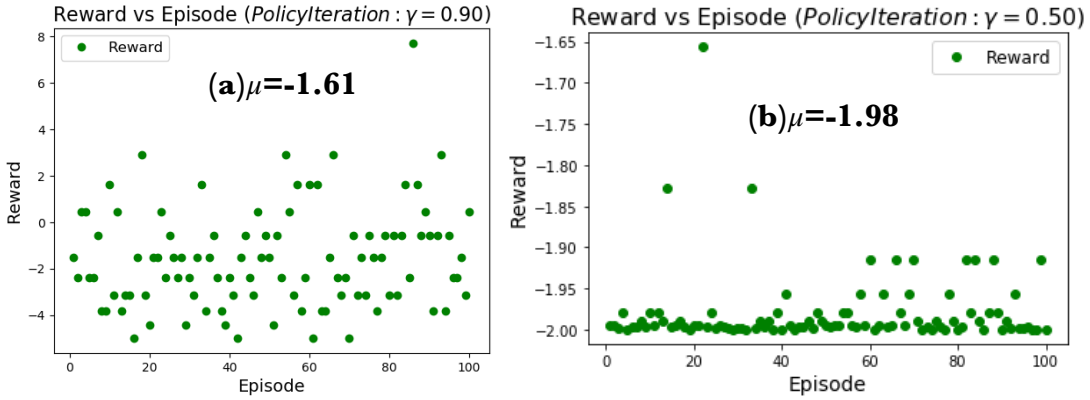


Fig.5: Taxi-Policy Iteration (a) Discount factor=0.9, (b) Discount factor=0.50

Q-Learning (ϵ-greedy):

I repeated the same simulations as before for the Taxi environment and results are plotted in Fig. 6. Smaller value of ϵ with same learning rate (column 1 vs 2) reaches best solution faster and therefore has higher average reward over 100000 episodes. This is because higher ϵ means higher random exploration rather than selecting the best action. Contrary to the Frozen Lake environment, average convergence time is smaller for smaller ϵ in this case. Since the environment has large number of states, the convergence time is much slower when the agent learns the environment at the beginning (lower panel plots in Fig. 6). Since larger ϵ value takes larger number of episodes to learn, average convergence time is expected to be higher in this environment. As for the effect of changing learning rate (column 2 vs 3), we observe same trend as before, higher learning rate is slower and less rewarding. The agent is more considerate to the most recent knowledge rather than the prior or best available knowledge and improvement in each iteration is relatively worse.

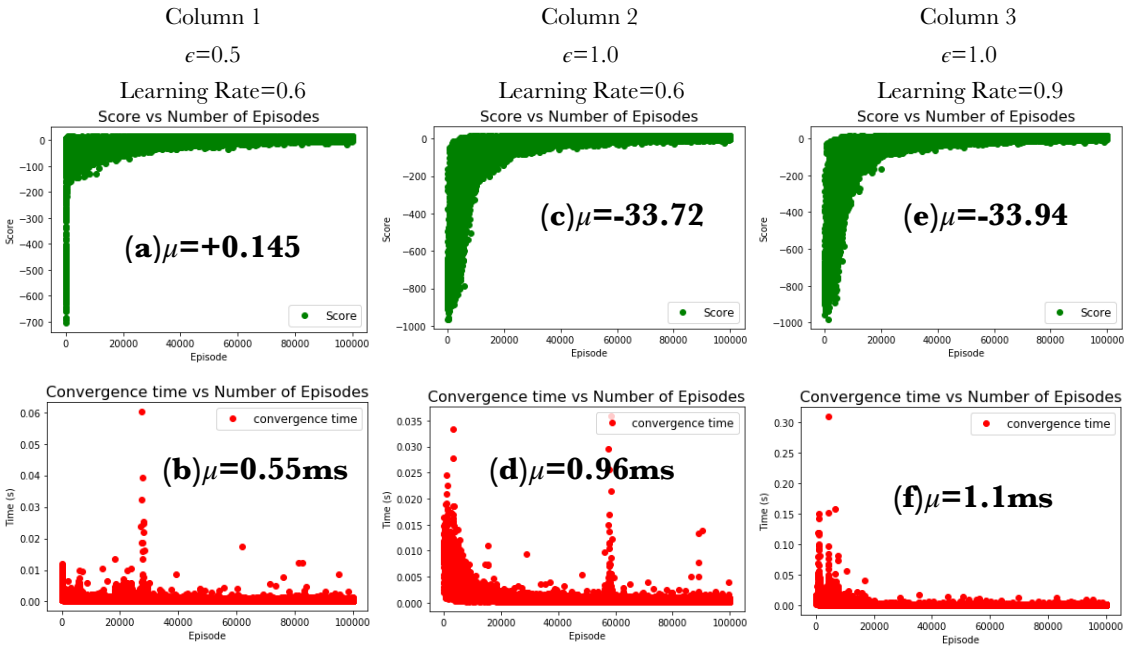


Fig.6: Taxi-Q-Learning : (a), (c), and (e) score vs Episode (b),(d), (f) convergence time vs Episode

Comparison:

Table 2 lists the performance parameters (rewards and convergence times) for the three algorithms (value iteration, policy iteration, and Q-Learning). The policy iteration is slower than

the value iteration algorithm although it is more computationally efficient (takes mush lesser number of iterations). Both algorithms converge to the same optimal policy. In terms of reward, value iteration performs better. Q-learning is much slower at the beginning when the agent has almost no or little knowledge about the more complicated environment (large number of states). In terms of parameter variations (ϵ and α), we observe similar trend as that of the FrozenLake environment. Regardless of the parameter settings (ϵ and α) that we considered here, the Q-learning always converges to the best policy (although a large number of episode is required to train the algorithm).

(Table.2: Comparison of three algorithms (Taxi-v3)

Algorithms	Total Episodes	Reward (mean/ episode)	Number of Iterations	Convergence Time (ms)
Value Iteration (γ=0.90)	**100**	**-1.35**	**95**	**764**
Value Iteration (γ=0.50)	**100**	**-1.977**	**16**	**165**
Policy Iteration (γ=0.90)	**100**	**-1.61**	**17**	**2878**
Policy Iteration (γ=0.50)	**100**	**-1.98**	**17**	**940**
Q-Learning (ϵ=0.5, Learning Rate=0.6)	**100000**	**0.145**	**15 (mean)**	**0.55 (mean)**
Q-Learning (ϵ=1.0, Learning Rate=0.6)	**100000**	**-33.72**	**24 (mean)**	**0.96 (mean)**
Q-Learning (ϵ=1.0, Learning Rate=0.9)	**100000**	**-33.94**	**24 (mean)**	**1.1 (mean)**

IV. Conclusion

We have explored the performance of policy iteration, value iteration, and Q-Learning (ϵ -greedy) algorithms on two grid world problems. The Frozen Lake environment is a simple MDP with only 16 states while the Taxi environment is much harder with 500 states. The key learnings from this assignment are as follows:

- Value iteration takes larger iteration (although not necessarily slower in terms of time) compared to the policy iteration. Policy iteration extracts policy in every single iteration and therefore each iteration takes longer time than value iteration.

• Both value and policy iteration converges to the same policy for both environments. The convergence criteria for both the algorithms were set to 10^{-3} (difference in value between two successive iterations)

• Both value and policy iteration converges much faster in a simple environment (FrozenLake). Computation efficiency significantly degrades in the more complicated Taxi-environment.

• Regardless of the starting conditions (ϵ and α), the Q-learning (ϵ-greedy) reaches the same best policy over time. However, since the algorithm doesn't have any domain knowledge at the beginning, it takes longer time compared to the value and policy iteration. Lower ϵ (less greedy algorithm) converges to best policy faster. Higher learning rate (α) slows down the learning process and takes longer iteration time to reach the best policy.

• Learning process in the Q-Learning (ϵ-greedy) algorithm is much slower for an environment with larger number of states (Taxi). However, the average convergence time does not degrade much compared to the significantly simpler (FrozenLake) environment. This is because, once the algorithm determines the best policy, the computation time does not depend much on the complexity of the environment.

• Value and policy iteration algorithms can be used for a simple environment. However, for an environment with large number of states, Q-Learning is the best performer among these three.

References:

1. Tom M. Mitchell, "*Machine Learning*"
2. https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa
3. https://gym.openai.com/envs/FrozenLake-v0/
4. https://gym.openai.com/envs/Taxi-v3/
5. https://github.com/allanbreyes/gym-solutions
6. https://github.com/openai/gym/tree/master/gym
7. https://gist.github.com/malzantot
8. https://www.novatec-gmbh.de/en/blog/introduction-to-q-learning/
9. https://www.quora.com/How-is-policy-iteration-different-from-value-iteration
10. https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/
11. https://www.cs.swarthmore.edu/~bryce/cs63/s17/slides/2-20_value_iteration.pdf
12. https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/node20.html
13. https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-7-action-selection-strategies-for-exploration-d3a97b7cceaf
14. https://en.wikipedia.org/wiki/Q-learning
15. Udacity lectures of CS7641