CS-7641 Machine Learning

Reinforcement Learning: Markov Decision Processes

Junle Lu
junle.lu@gatech.edu
Georgia Institute of Technology
Nov 25, 2018

Abstract – This paper investigates three reinforcement learning algorithms: value iteration, policy iteration, and Q-learning. They are applied on two Markov decision processes (MDPs): EasyGrid and HardGrid. The performance and differences are discussed among the three algorithms.

1. Introduction

A **Markov decision process (MDP)** is a discrete time stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker [1]. The process has the following components:

- **S:** a set of states of the given process
- **A:** a set of all possible action can be taken in a state
- **T(s,a,s') = Pr(s' | s, a):** a transition model representing the probability of an action to a state
- **R(s,a):** a read-valued reward function. It is the reward for being in a state and taking an action
- **Policy:** the policy that defines the actions to take in a particular state

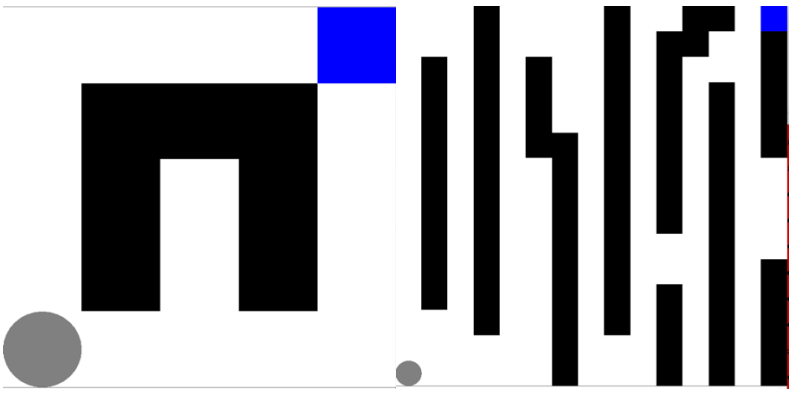
The MDPs are typically solved with either **value iteration** or **policy iteration**. The value iteration is finding the values for the states over iterations and 'learn' the optimal policy. It always tries to maximize reward. Each state is assigned with a 'reward' value and returned by a utility function. Each action will lead to the state with the maximum utility value. The drawback is that each state's 'reward' value is usually unknown. With policy iteration, it attempts to find the optimal policy directly by iterating over policies instead of states. The drawback is that it can be very slow to converge. Both algorithms are great to apply on the known MDPs, that the sufficient information is given such as find the path in a graph or solve a maze.

Reinforcement Learning (RL) is a type of Machine Learning and a branch of artificial intelligence. It does not need the knowledge about the MDP. It allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behavior; this is known as the reinforcement signal [2]. In this paper, **Q-learning (QL)** algorithm is implement on the chosen MDPs. The Q-learning algorithm is not heavily depended on the

strong assumptions about the stochastic process or the transition model. Therefore, it learns the optimal policy from the incomplete information about the MDPs, and it assigns estimated values known as Q-value to the states. Over the iterations, the Q-values are updated and may converge to some values.

2. Markov Decision Process Problems

Grid World is shown in the lecture and it is a classic example of an MDP problem. The process has three different grids or states: wall, empty space, and terminate. The actions are directional moves: up, down, left, and right. The process is simple itself, but that also makes it **interesting**. It represented by small sets of states and actions. The move actions are very intuitive, and the states are very common. Most algorithms will converge within 100 iterations or less. More importantly, this problem can be visualized to inspect and discuss on the performance of different algorithms. It is even more interesting that this can be seen (in some degree) that model the self-driving vehicle path planning implementation. The two different grids are easy and hard grids. One is 5x5 and the other is 15x15 size as shown below:



Easy Grid 5x5 Hard Grid 15x15

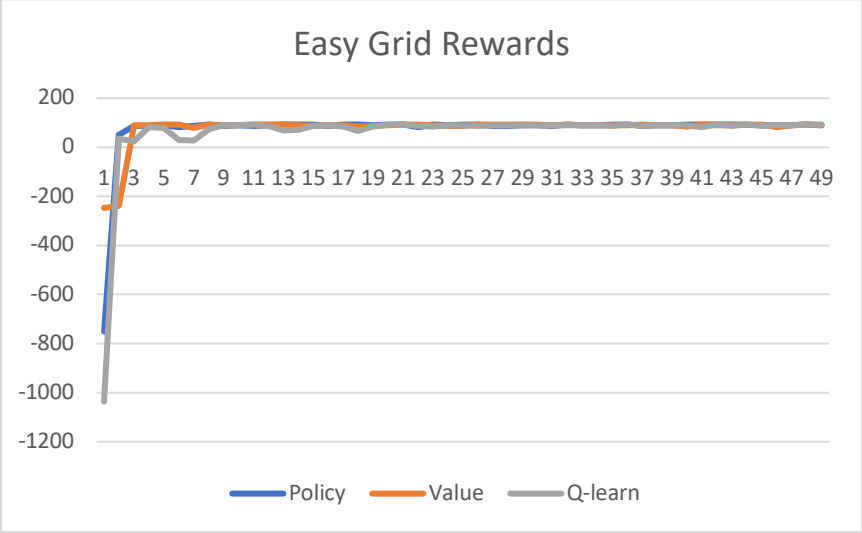
The learning algorithms find the optimal policy to reach the terminal state as in blue color. The reward is -1 for non-obstacle grids and 100 for terminal grid.

3. Implementation

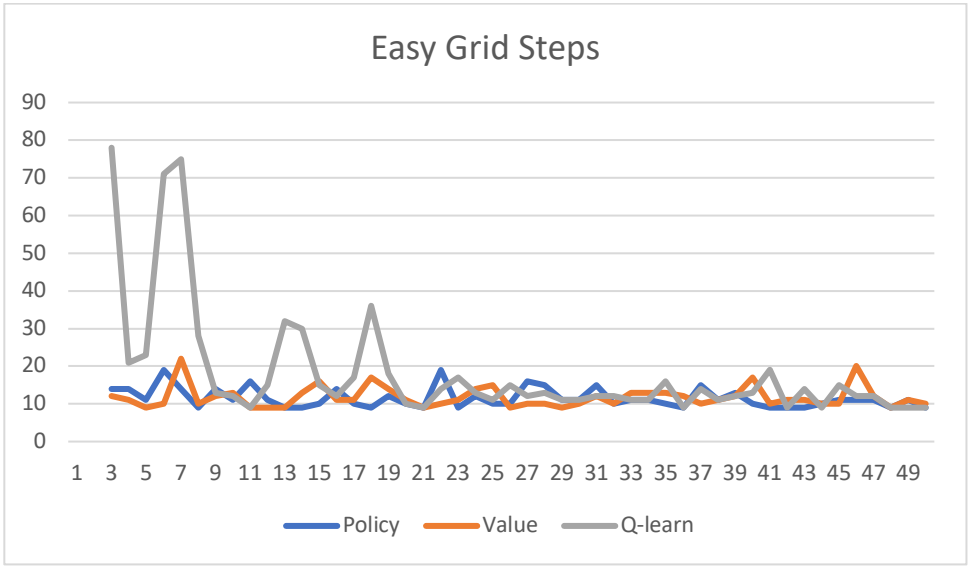
All the algorithm implementations are using BURLAP library. The default parameters are used for the value and policy iteration algorithms. The learning rate and epsilon value are experimented for the Q-learning algorithm.

4. **Easy Grid**

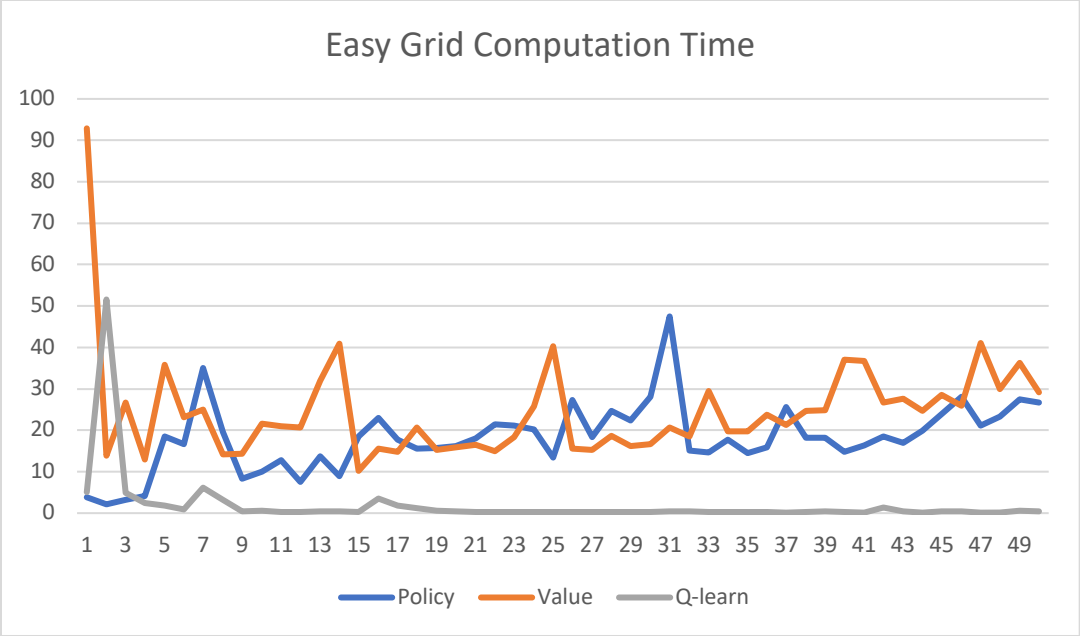
The three difference algorithms ran with 50 iterations. For Q-learning, the learning rate is set to 0.1 and the epsilon is set to 0.7



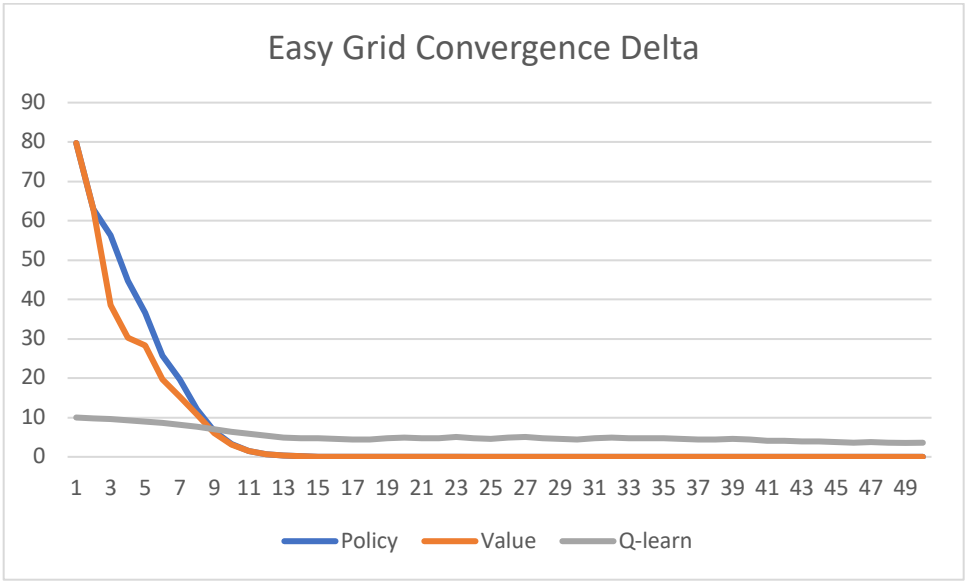
All three algorithms converge reach the desired reward value (around 100) in just 3 to 5 iterations. However, the Q-learning has oscillation and takes more iterations to stabilize the result. This is expected as the Q-learning is based on estimation of the state values.



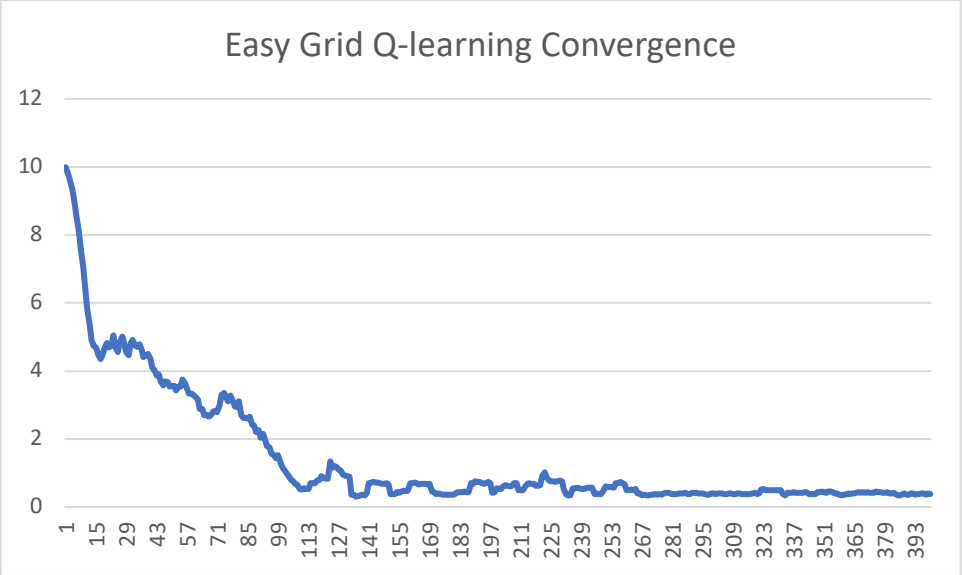
For the steps, the first couple iterations are pretty large steps for all algorithms. I removed them from the plot to make more presentable to inspect the convergence. Both value and policy algorithms are having the similar performance, which converge to 10 steps around 3-5 iterations. As expected, the Q-learning oscillates and takes a few more iterations to stabilize.



Q-learning clearly has advantage in computation time over the other two algorithms. It runs at constant time for each iteration because it just needs to update the Q-values until a policy is computed. For policy and value algorithms, their performance are similar in this case.

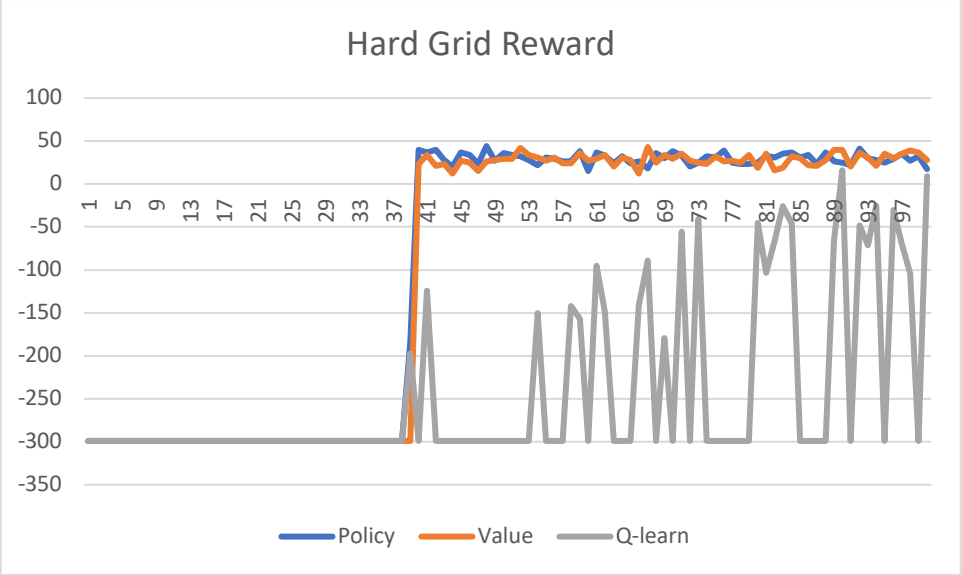


The value and policy algorithms converge very fast at around 12 iterations. It's expected as the MDP model is known to both algorithms. In contrary, the Q-learning takes much longer time to converge as shown in graph below for 500 iterations:

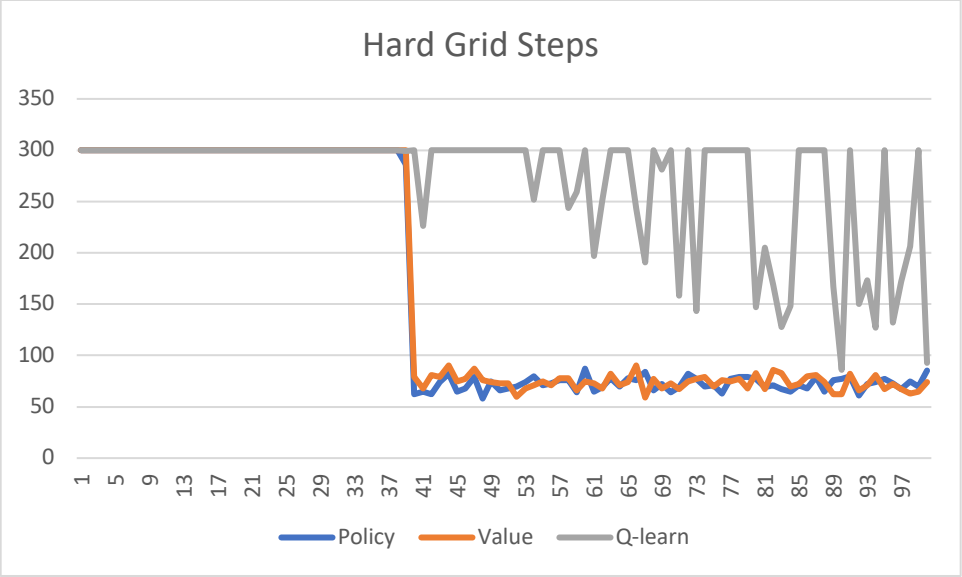


5. Hard Grid

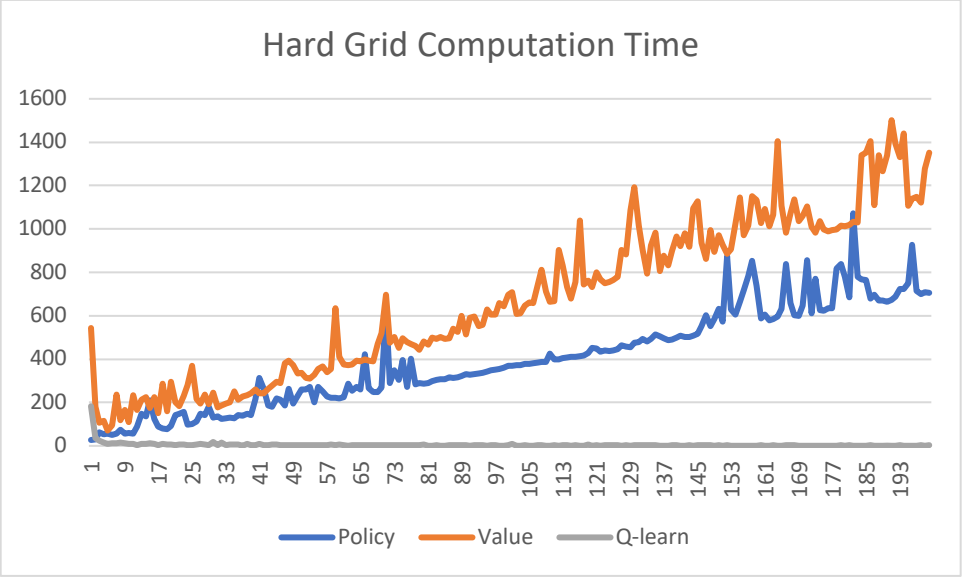
The hard grid ran with algorithms with 100 iterations for value an policy algorithms. For Q-learn, the learning rate is 0.1 and epsilon is 0.7.



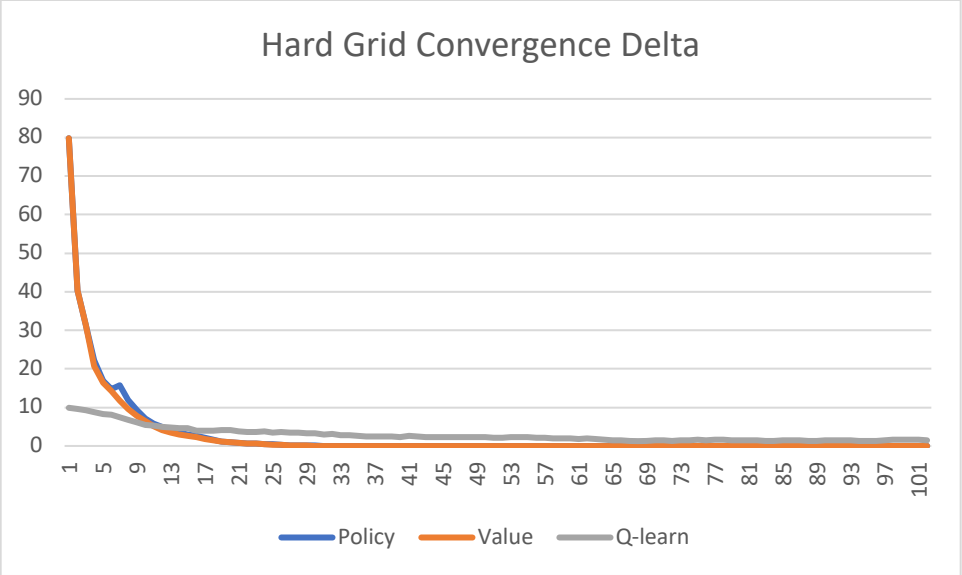
The value and policy algorithms have the consistence performance as in the Easy Grid problem. They both converge at 40 iterations. As the problem get more complicated, Q-learn is struggling to converge at the beginning and expected to take more iterations to converge.



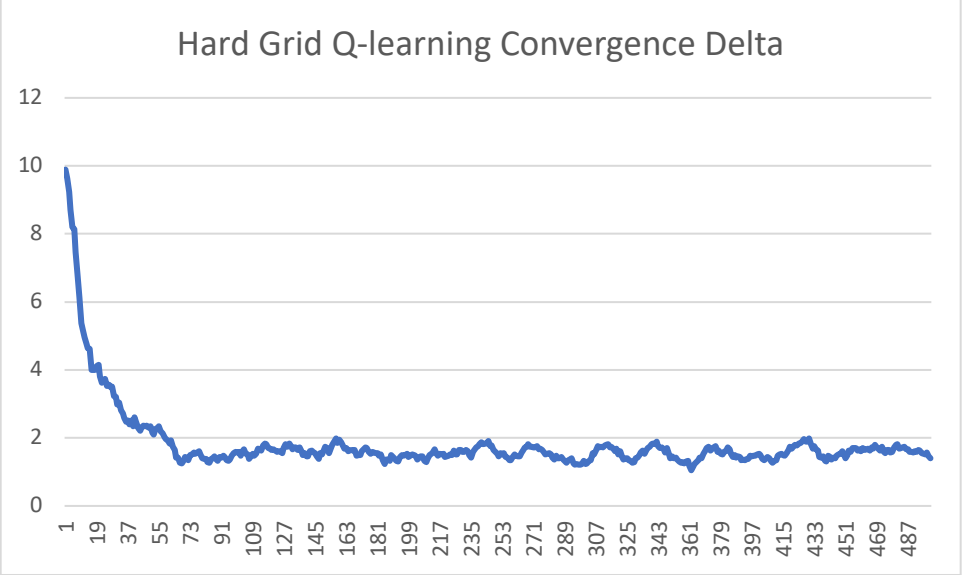
The steps are having the similar performance as in rewards for all three algorithms.



As stated in the Easy Grid problem, the Q-learn takes constant time in for the computation in every iteration. The value iteration takes more time than the policy iteration algorithm in this problem as it gets more complicated.

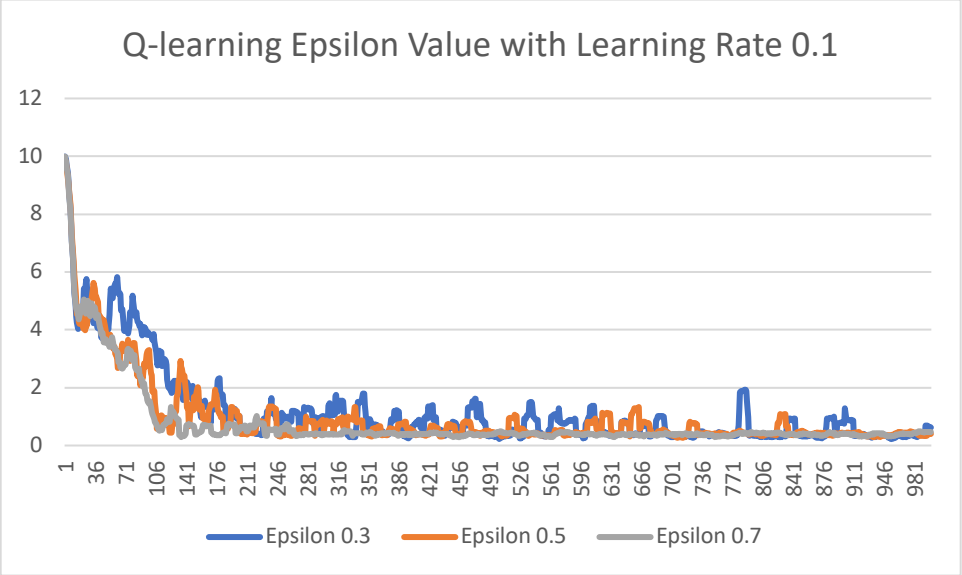


As expected, the delta value converges at 0 quickly for value and policy iterations (at 23 iterations). It takes much longer for the Q-learn algorithm as shown in graph below for 500 iterations:

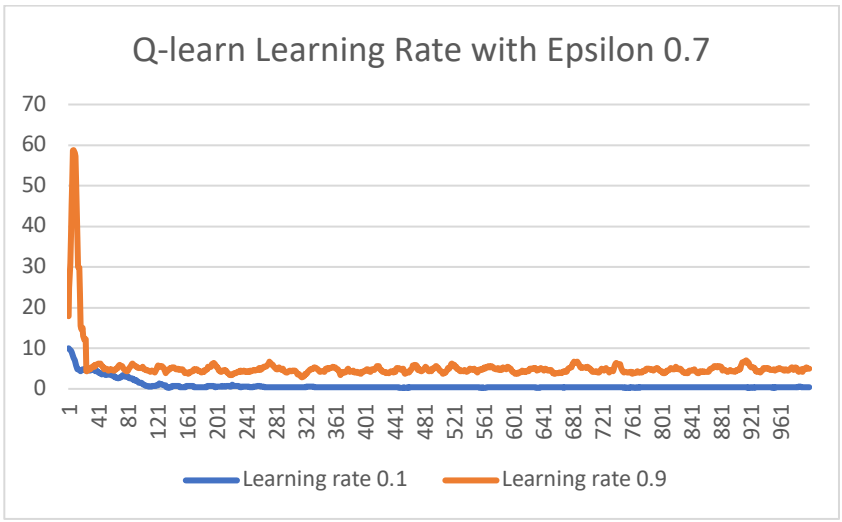


6. Q-learning: learning rate and epsilon value

The learning rate and epsilon value are two key parameters for the Q-learn algorithm. The learning rate represent the importance of each reward from the utility function. The epsilon controls if the agent randomly perform an action without looking at the utility value. The experiment is conducted on the Easy Grid problem with learning rate fixed to 0.1. Different epsilon values are tested as shown in the graph below. It clearly shows that high epsilon value converges faster.



In the second experiment, the epsilon value is set to high (0.7) to investigate the difference in learning rates. As expected, the lower learning rate results in faster convergence. It takes more iteration to learn the MDP with higher learning rate.



7. Conclusion

The value and policy iteration converge much faster than Q-learn as expected. But Q-learn takes less computation time, also it does not need the complete information about the MDP problem. For Q-learn, we learned that higher epsilon and lower learning rate result in faster convergence. Therefore, if the MDP problem is well given, then value or policy iteration is desired. Otherwise, Q-learn is the preferred choice of algorithm to solve the MDP problem.

8. Reference

[1] https://en.wikipedia.org/wiki/Markov_decision_process
[2] http://reinforcementlearning.ai-depot.com/