Policy Iteration vs Value Iteration

19520208 - Huynh Phuong Nhu

June 2021

1 Introduction

1.1 Policy Iteration

Policy Iteration is a way to find the optimal policy for given states and actions. Assuming we have a policy (π : S $\rightarrow$ A) that assigns an action to each state. Action $\pi(s)$ will be chosen each time the system is at state s.

Basic Idea

1. Evaluate a given policy (eg. initialise policy arbitrarily for all states $s \in S$) by calculating value function for all state $s \in S$ under the given policy.

$$V_\pi(s) = E[R(s, \pi(s), s') + \gamma V(s')] \tag{1}$$

   With $V_\pi(s)$ is the value function, $E[R(s, \pi(s), s') + \gamma V(s')]$ is the expected value of reward collected at the first step add with discounted value at the next state.

2. Improve policy: find a better action for state $s \in S$

$$\pi_1(s) = \arg\max_{a \in \mathcal{A}} E[R(s, \pi(s), s') + \gamma V(s')] \tag{2}$$

3. Repeat step 1,2 until value function converge to optimal value function

1.2 Value Iteration

Value iteration is a method of computing an optimal policy for an MDP and its value.

Basic Idea

Value iteration starts at the "end" and then works backward, refining an estimate of either Q^* or V^*. There is really no end, so it uses an arbitrary end point. Let V_k be the value function assuming there are k stages to go, and let Q_k be the Q-function assuming there are k stages to go. These can be defined recursively. Value iteration starts with an arbitrary function V_0. For subsequent

stages, it uses the following equations to get the functions for $k+1$ stages to go from the functions for k stages to go

$$Q_{k+1}(s,a) = R(s,a) + \gamma * \sum_{s'} P(s'|s,a) * V_k(s') \tag{3}$$

$$V_k(s) = \max_a Q_k(s,a) \tag{4}$$

1.3 Environments

We will implement Value Iteration and Policy Iteration on 3 environments:

1. FrozenLake-v0
2. FrozenLake8x8-v0
3. Taxi-v3

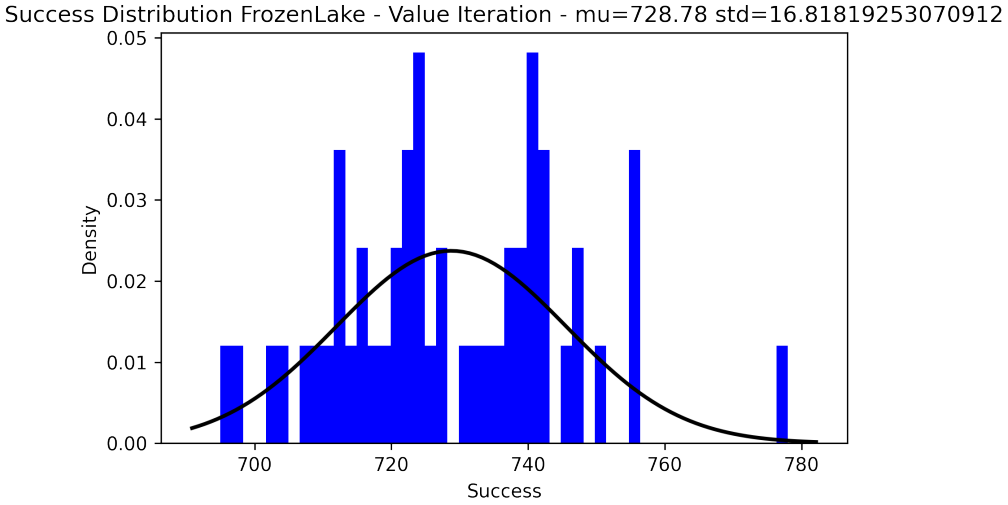
With FrozenLake, the difference between the two is its observation space whereas in the v0 we will have a 4X4 matrix and the 8x8-v0 will be an 8x8 matrix. Our target is to find a safe path across a grid of ice and water tiles while reaching the final goal for the reward and dodge holes. Each episode will be displayed as characters S, F, H, G where S is the starting point and safe; F is the frozen surface, also safe; H is the hole where you will end the journey if reached and the final character is G which is the goal, where the frisbee and reward are located. For each move you make, you will not be rewarded unless you reach the final goal with the reward equals to 1. Also, the probability of successfully executing an action is 1/3.
The game Taxi-v3 which was introduced in [1] to illustrates some issues in hierarchical reinforcement learning. There are 4 locations (labeled by different letters) and your job is to pick up the passenger at one location and drop him off in another. You receive +20 points for a successful dropoff, and lose 1 point for every timestep it takes. There is also a 10 point penalty for illegal pick-up and drop-off actions. It is different from FrozenLake that the probability of executing an action successfully is 1.0. Because of this reason, we will not judge this environment to be successful or not; we only consider the average reward of each play.

2 Comparison

In order to deduce an conclusion, I am using data generated by source code in this folder.

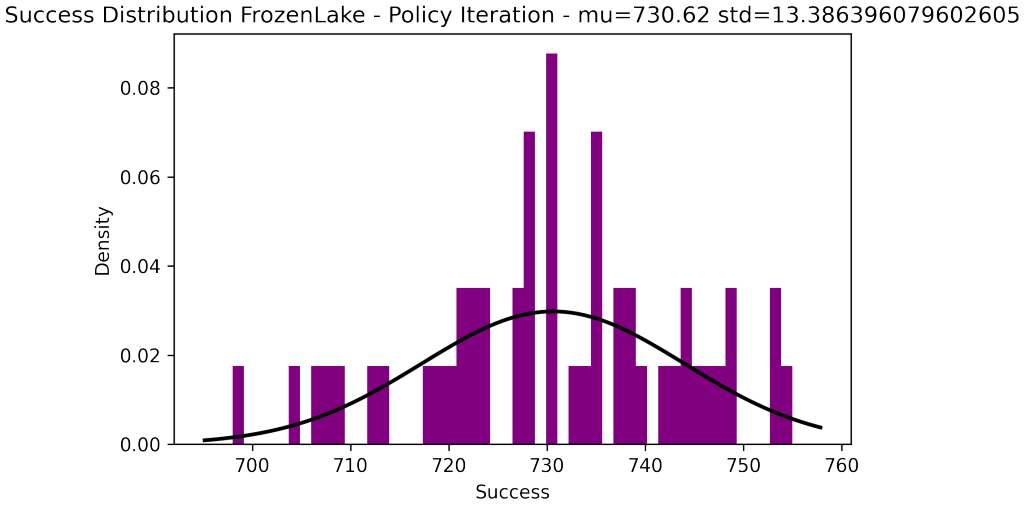
2.1 FrozenLake-v0

2.1.1 Value Iteration



Looking at the distribution above, we can see that mean of success over 1000 times of applying policy derived from Value Iteration is around 728.78 and most of the time, the success rate over 1000 episode 's iterations is concentrated between roughly 71% to around 74.5%.

2.1.2 Policy Iteration

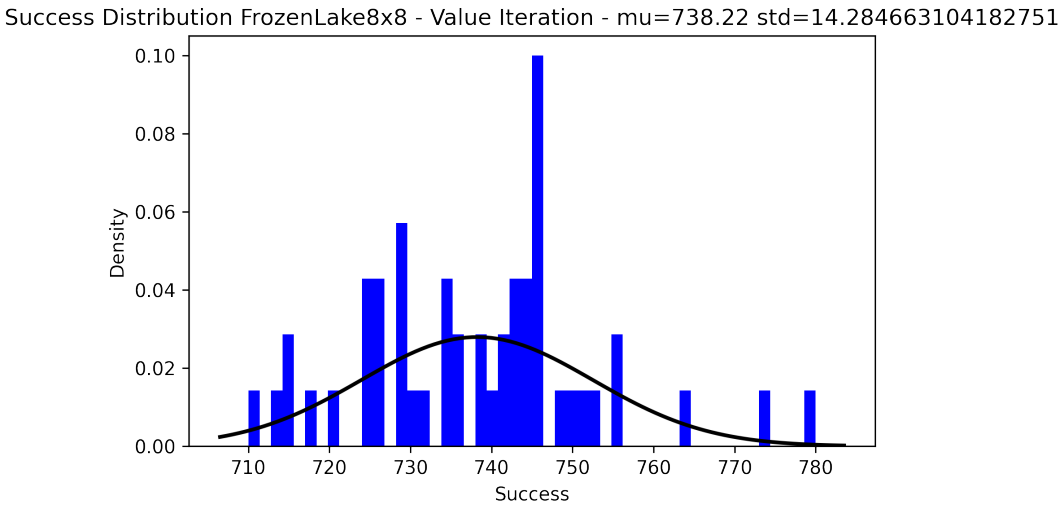


The graphic above is displaying the number of time the agent success going to the goal over 1000 episodes when applying a policy from Policy Iteration. The statistics obtained from applying the policy are a bit higher than the Value

Iteration one with its average is 730.62. The time takes to run Policy Iteration is also smaller than running Value Iteration.

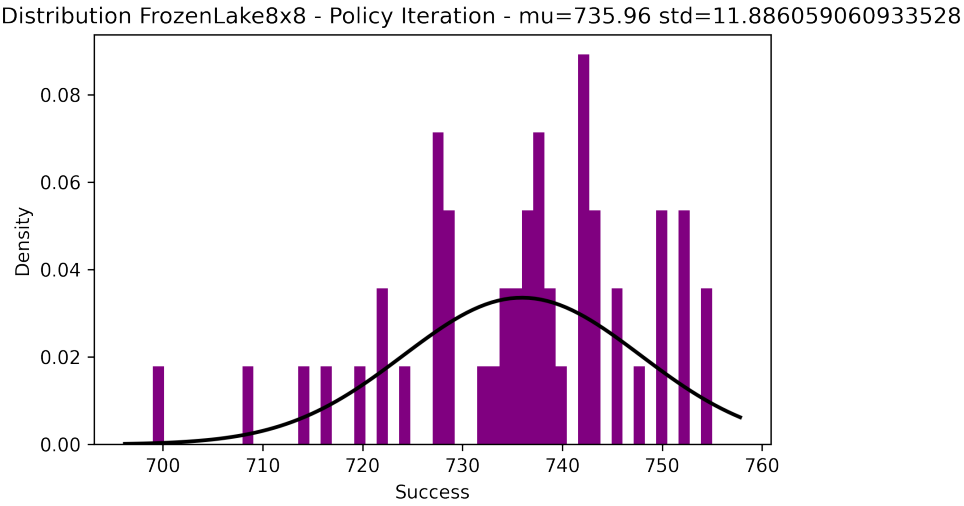
2.2 FrozenLake8x8-v0

2.2.1 Value Iteration



It can be seen that average successful episodes is 738.22 and its usual success rate is around 72.4% to 75.2%.

2.2.2 Policy Iteration



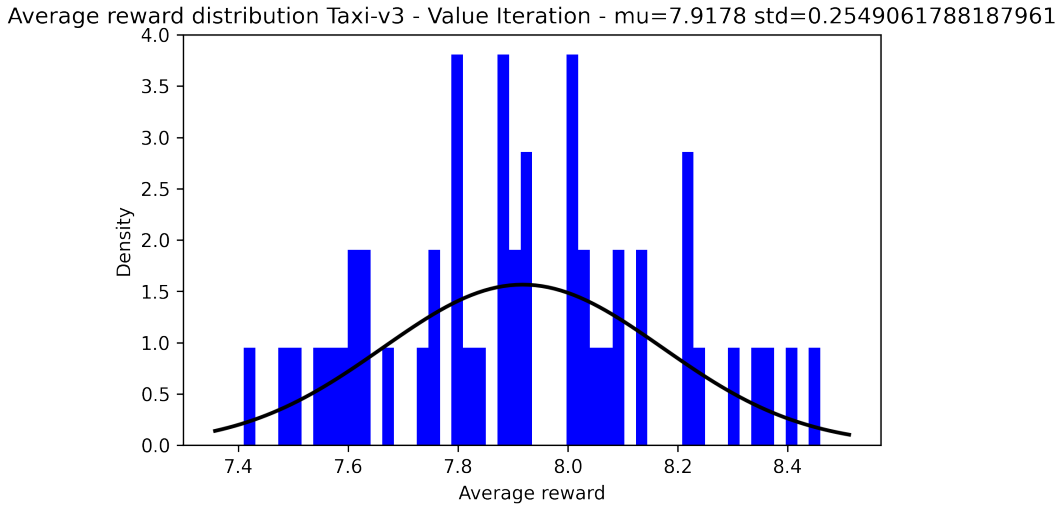
Comparing to Value Iteration, the time it takes is about the same but the policy

iteration converges quite quickly around 4-9 iterations whereas Value Iteration takes around 117 iterations.

2.3 Taxi-v3

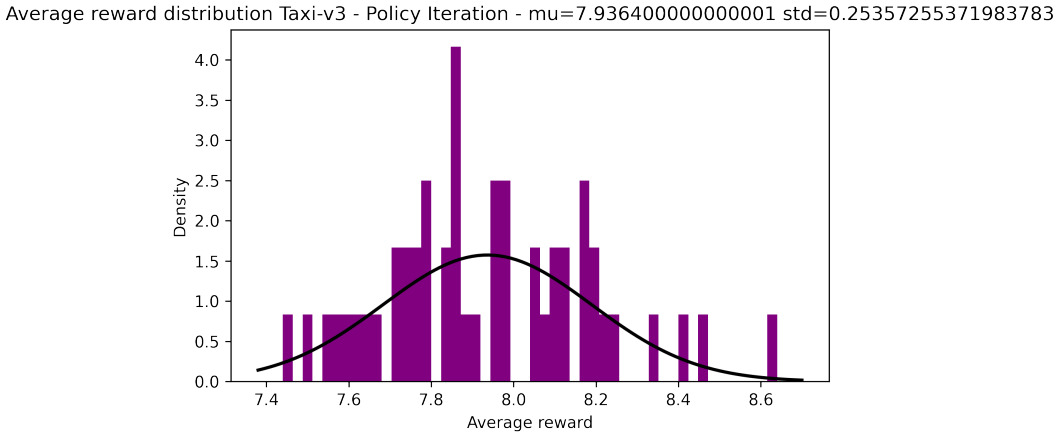
For this environment, I will only run 100 episodes per iterations.

2.3.1 Value Iteration



Value Iteration gave us an average reward of each episode to be around 7.9178. And each episode takes around 0.8-1.3 seconds.

2.3.2 Policy Iteration



Policy Iteration gave us an average reward of each episode to be around 7.9364

which is not far the given result of Value Iteration. And each episode takes around 1.6-2.5 seconds, which is around 1.8-1.9 times.

2.4 Conclusion

After running both algorithms, we can deduce that the 2 algorithms share the same working principle. However, because value iteration is based on optimality Bellman operator which contains a max operator (not linear). Therefore, it has some different features.
Talking about the time it takes to derive a solution depend on the environment state; for smaller state space like FrozenLake, Policy Iteration performs much better than Value Iteration as a policy converges more quickly than a value function. But for larger state space like Taxi-v3, Value Iteration will perform better than Policy Iteration and in fact, it is used more in real life situations where there are lots of state in a problem [2].
Talking about the final solution that both algorithms gave us, Policy Iteration is assured to have an optimal solution while the Value Iteration is not.

References

[1] Thomas G. Dietterich. "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition". In: *CoRR* cs.LG/9905014 (1999). URL: https://arxiv.org/abs/cs/9905014.

[2] Andrew Ng. "MDPs Value/Policy Iteration". https://www.youtube.com/watch?v=d5gaWTo6kDM&t=3636s. 2018.