Project Tasks Solutions

Somesh Patro

June 2025

1 Optimal Bellman Equations for Value and Action Value Functions

1.1 What is Optimality?

Optimality in the context of Markov Decision Processes (MDP) refers to the condition in which a policy maximizes the expected cumulative reward over time. A policy π is considered optimal if it produces the highest expected return of any state s.

1.2 What is Expectation and How Does it Relate to Bellman Equations?

Solution: Expectation is a statistical measure that calculates the average of a set of values, weighted by their probabilities. In the context of Bellman equations, the expectation is used to compute the expected value of future rewards given a state and action. The Bellman equation for the value function $V(s)$ is given by:

$$V(s) = \sum_{a \in A(s)} \pi(a|s) \sum_{s' \in S} P(s'|s,a) \left[R(s,a,s') + \gamma V(s') \right]$$

where:

- $V(s)$ is the state value of s
- $\pi(a|s)$ is the policy
- $P(s'|s,a)$ is the transition probability
- $R(s,a,s')$ is the reward
- γ is the discount factor.

1.3 Exercises 3.25 to 3.29

Exercise 3.25: Equation for v_* in terms of q_*

Solution: The optimal value function v_* and the optimal action value function q_* are related to each other. Their relation is given below as follows:

$$v_*(s) = \max_{a \in A} q_*(s,a)$$

Exercise 3.26: Equation for q_* in terms of v_* and the Four-Argument p

Solution: The optimal action-value function q_* can be expressed in terms of the optimal value function v_* and the three-argument function p as follows:

$$q_*(s,a) = \sum_{s' \in S} p(s'|s,a)\,[r(s,a,s') + \gamma v_*(s')] \tag{1}$$

where $r(s,a,s')$ is the reward function.

Exercise 3.27: Equation for π_* in terms of q_*

Solution: The optimal policy π_* can be derived from the optimal action-value function q_* as follows:

$$\pi_*(s) = \arg\max_{a \in A} q_*(s,a)$$

Exercise 3.28: Equation for π_* in terms of v_* and the Four-Argument p

Solution: The optimal policy π_* can also be expressed in terms of the optimal value function v_* and the four-argument function p as follows:

$$\pi_*(s) = \arg\max_{a \in A} \sum_{s' \in S} p(s'|s,a)\,[r(s,a,s') + \gamma v_*(s')]$$

Exercise 3.29: Rewrite the Four Bellman Equations

Solution: The four Bellman equations for the value functions v^π, v_*, q^π, and q_* can be rewritten in terms of the three-argument function p and the two-argument function r as follows:

- **For v^π:**
  $v^\pi(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} p(s'|s,a)\,[r(s,a,s') + \gamma v^\pi(s')]$
- **For v_*:**
  $v_*(s) = \max_{a \in A} \sum_{s' \in S} p(s'|s,a)\,[r(s,a,s') + \gamma v_*(s')]$
- **For q^π:**
  $q^\pi(s,a) = \sum_{s' \in S} p(s'|s,a)\,[r(s,a,s') + \gamma v^\pi(s')]$
- **For q_*:**
  $q_*(s,a) = \sum_{s' \in S} p(s'|s,a)\,[r(s,a,s') + \gamma \max_{a' \in A} q_*(s',a')]$

2 Prove Convergence of Policy and Value Iterations

Policy Iteration: Policy iteration consists of two main steps: policy evaluation and policy improvement. Each policy improvement step guarantees that the new policy is at least as good as the previous one, leading to a sequence of monotonically improving policies. In a finite Markov Decision Process (MDP), there are a finite number of possible policies. Since each iteration produces a strictly better policy (unless the optimal policy is reached), the process must converge in a finite number of iterations. The value function for the current policy is computed using the Bellman Expectation Equation, which ensures that the value function converges to the true value of the policy.

Value Iteration: Value iteration updates the value function for each state using the Bellman Optimality Equation until convergence is achieved. The iterative updates are based on the maximum expected return from each state, ensuring that the value function approaches the optimal value function. The updates in value iteration can be viewed as a contraction mapping, which guarantees convergence to a unique fixed point (the optimal value function) under certain conditions. Once the value function converges, the optimal policy can be derived by selecting actions that maximize the expected return based on the converged value function.

2.1 Intuition on Policy and Value Iterations

Policy iteration method calculates the value function on the initialized policy. It then updates the policy to optimise the value function. Whereas, value function iterates to optimise the randomly initialized value function until an optimal policy is reached.

2.2 What is GPI?

Generalized Policy Improvement (GPI) refers to the process of improving a policy based on the value function. It states that if a policy is improved in a way that it is greedy with respect to the value function, then the new policy will have a value that is at least as good as the old policy.

2.3 Proof of Convergence

Both policy iteration and value iteration converge to the optimal value and policy function under certain conditions. The convergence can be shown using the contraction mapping theorem.

2.4 Time and Memory Complexity

Policy Iteration:

- Time Complexity: $O\left(\frac{|S|\cdot|A|}{\epsilon}\right)$ per iteration, where S is the number of states,A is the number of possible actions, and ϵ is the epsilon/theta value.
- Memory complexity: $O(|S|)$ to store the value function and the policy.

Value Iteration:

- Time complexity: $O(|S|^2)$ per iteration, where S is the number of states.
- Memory complexity: $O(|S|)$ to store the value function and the policy.

3 Implement DP Algorithms on Frozen Lake from Gymnasium

The Frozen Lake environment has been trained using the Value Iteration method. We will use the metrics *convergence time* and *mean time per episode* for our analysis.

3.1 Built-in Gymnasium environment

The Built-in gymnasium methods allow us to train the environment without worrying about writing any tedious code. The results are as follows.

- Convergence Time = 0.8461 ms
- Mean Time Per Episode = 0.0275 ms

3.2 Custom Environment

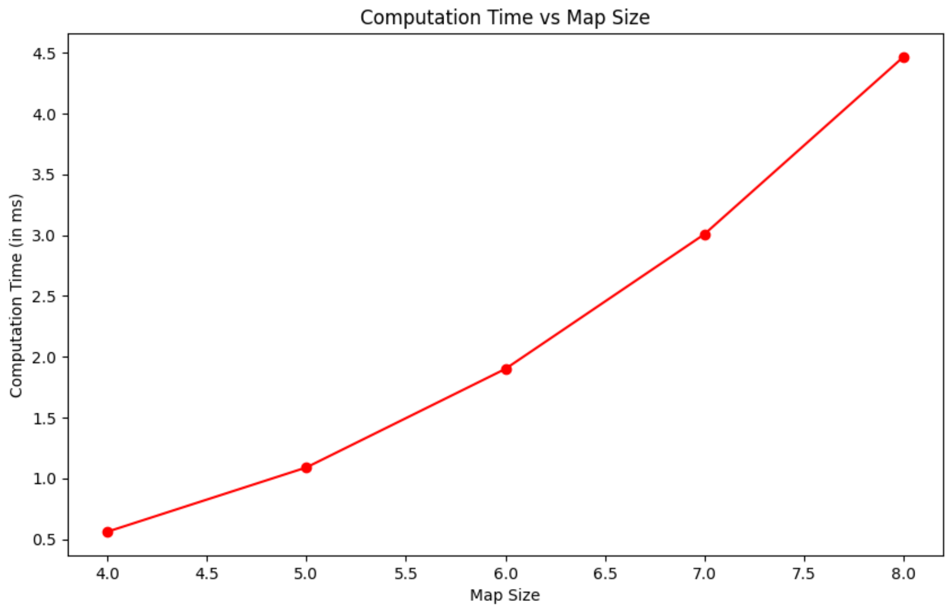
Although we have to write all the code by ourselves, the custom environment converges faster than the gymnasium environment. This is because of the additional packages that come with the gymnasium module that slightly slow it down. The results:

- Convergence Time = 0.6143 ms
- Mean Time Per Episode = 0.0057 ms

3.3 Expanded Custom Environment

In this expanded custom environment, the map size can be varied for sizes greater than 4. Also, there are 8 possible actions instead of 4 giving the agent more flexibility. Maps are randomly generated keeping in check that a valid path exists. Analysis over different map sizes yields:

- Slope of convergence time v/s map size = 0.976



*****