Markov Decision Process

Assignment Week 1(RL)(Theory part)

June 8, 2025

Yash Bankar
Roll Number: 24034017

---

1. Derive optimal bellman equations for value and action value functions
   (a) What is optimality?
   (b) What is expectation and how does it relate to bellman equations?
   (c) Solve exercise 3.25- 3.29 from Sutton and Barto
      (a) Exercise 3.25 Give an equation for v_* in terms of q_*.
      (b) Exercise 3.26 Give an equation for q_* in terms of v_* and the four-argument p.
      (c) Exercise 3.27 Give an equation for π_* in terms of q_*.
      (d) Exercise 3.28 Give an equation for π_* in terms of v_* and the four-argument p.
      (e) Exercise 3.29 Rewrite the four Bellman equations for the four value functions (v_π, v_*, q_π, and q_*) in terms of the three-argument function p (3.4) and the two-argument function r (3.5).

Ans: A1.

Deriving Optimal Bellman Equations

Optimal State-Value Function

The optimal value function $v_*(s)$ represents the maximum expected return achievable from a state s under any policy. It is defined as:

$$v_*(s) = \max_a q_*(s, a)$$

Using the definition of the action-value function and the expected return, we can expand this:

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \\ &= \max_a \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a \right] \\ &= \max_a \mathbb{E} \left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a \right] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_*(s') \right] \end{aligned}$$

This is the **Bellman optimality equation** for the value function v_*.

Optimal Action-Value Function

The optimal action-value function $q_*(s, a)$ gives the maximum expected return after taking action a in state s and thereafter following the optimal policy. It can be expressed as:

$$\begin{aligned} q_*(s, a) &= \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a \right] \\ &= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right] \end{aligned}$$

This is the **Bellman optimality equation** for the action-value function q_*.

Summary

The Bellman optimality equations form the basis for many reinforcement learning algorithms. They define recursive relationships:

$$\boxed{v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_*(s') \right]}$$

$$\boxed{q_*(s, a) = \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right]}$$

These equations assert that the value of a state or state-action pair under an optimal policy equals the expected immediate reward plus the discounted optimal value of the next state.

(a) Optimality in the context of Markov Decision Processes (MDPs) refers to selecting a policy π^* that yields the highest expected return from every state. An optimal policy π^* satisfies:

$$v_{\pi^*}(s) \geq v_\pi(s) \quad \forall s \in \mathcal{S}, \forall \pi$$

This means that no other policy achieves a higher value function for any state.

(b) The expectation operator $\mathbb{E}$ computes the average of a random variable with respect to its probability distribution. In Bellman equations, we use expectations to compute the expected return — the average future cumulative reward:

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$$

This equation expresses the fact that the value of a state is the expected immediate reward plus the discounted expected value of the next state, under policy π.

(c) Exercise Solutions from Sutton & Barto (3.25 – 3.29)

Exercise 3.25: Give an equation for v_* in terms of q_*

$$v_*(s) = \max_a q_*(s, a)$$

This means the optimal state-value function is the maximum action-value obtainable from state s.

Exercise 3.26: Give an equation for q_* in terms of v_* and the four-argument p

$$q_*(s, a) = \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma v_*(s') \right]$$

Here, $q_*(s, a)$ is the expected reward plus the discounted optimal value of the next state.

Exercise 3.27: Give an equation for π_* in terms of q_*

$$\pi_*(s) = \arg\max_a q_*(s, a)$$

This means the optimal policy selects the action that maximizes q_* in each state.

Exercise 3.28: Give an equation for π_* in terms of v_* and the four-argument p

$$\pi_*(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_*(s')\right]$$

This is a formulation of the optimal policy based on evaluating the expected return for each action.

Exercise 3.29: Rewrite the Bellman Equations Using $p(s'|s, a)$ and $r(s, a)$

Assume: - $p(s'|s, a)$ is the probability of transitioning to s' from s using action a. - $r(s, a)$ is the expected reward for taking action a in state s.

- **State-value under policy:**

$$v_\pi(s) = \sum_a \pi(a|s) \left[r(s, a) + \gamma \sum_{s'} p(s'|s, a) v_\pi(s')\right]$$

- **Optimal state-value:**

$$v_*(s) = \max_a \left[r(s, a) + \gamma \sum_{s'} p(s'|s, a) v_*(s')\right]$$

- **Action-value under policy:**

$$q_\pi(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \sum_{a'} \pi(a'|s') q_\pi(s', a')$$

- **Optimal action-value:**

$$q_*(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \max_{a'} q_*(s', a')$$

These forms are useful when working with environments where reward and transitions are defined as separate deterministic or expected functions.

2. Prove convergence of policy and value iterations
   (a) Try to build some intuition on how are they working and why should they work
   (b) What actually is GPI?
   (c) What is the complete and rigorous proof of their convergence?
   (d) What is the time and memory complexity for each of the algorithms? Where are each of them particularly useful?

Ans: A2.

Convergence of Policy Iteration and Value Iteration

Overview

Both Policy Iteration (PI) and Value Iteration (VI) are classical Dynamic Programming algorithms used to compute the optimal policy for a Markov Decision Process (MDP) when a full model of the environment is known.

They are guaranteed to converge under standard assumptions, such as a finite state and action space and bounded rewards.

(a) **Intuition and Working of Policy and Value Iteration**

Policy Iteration: It consists of two main steps:

- *Policy Evaluation:* Compute v_π for the current policy π.
- *Policy Improvement:* Improve π greedily by selecting actions that maximize expected returns.

This process repeats until the policy stops changing.

Value Iteration: Instead of full evaluation, VI performs a single Bellman update per state:

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right]$$

It continues updating the value function until convergence and then derives the optimal policy.

(b) **What is Generalized Policy Iteration (GPI)?**

GPI refers to the general interaction of policy evaluation and improvement, not necessarily to the strict alternation seen in policy iteration. It allows for:

- Partial evaluation
- Infrequent or asynchronous improvement

Regardless of the schedule, GPI converges to the optimal policy under the assumption that evaluation and improvement continue.

(c) **Proof of Convergence**

Value Iteration Convergence:

The Bellman optimality operator T defined as

$$(Tv)(s) = \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma v(s')\right]$$

is a contraction mapping in the space of value functions under the supremum norm, with contraction modulus $\gamma < 1$. By the Banach Fixed Point Theorem, iterating T:

$$v_{k+1} = Tv_k$$

converges to the unique fixed point v_*. That is,

$$\lim_{k\to\infty} v_k = v_*$$

Policy Iteration Convergence:

Each iteration strictly improves the policy unless it is already optimal. Because there are only finitely many deterministic policies in a finite MDP, the process must terminate in a finite number of steps with the optimal policy π_*.

(d) Time and Memory Complexity of Policy and Value Iteration

Complexity Analysis

The time complexity for each iteration of both value iteration and policy iteration is generally upper bounded by:

$$O(|\mathcal{S}|^2|\mathcal{A}|)$$

This is because, during each iteration, we must update the value of every state $s \in \mathcal{S}$. To do so, we evaluate all possible actions $a \in \mathcal{A}$ and look at all possible transitions to next states $s' \in \mathcal{S}$ via $p(s', r \mid s, a)$. This gives a complexity of $O(|\mathcal{S}||\mathcal{A}||\mathcal{S}|) = O(|\mathcal{S}|^2|\mathcal{A}|)$.

However, this is a worst-case bound. In structured environments like grid-worlds or mazes, an agent typically transitions only to a few neighboring states (e.g., 4 or 8), reducing the effective cost significantly in practice.

Convergence Behavior

While both algorithms converge to the optimal policy, the number of iterations differs:

- **Value Iteration** performs many cheap iterations using a Bellman backup:
$$v_{k+1}(s) = \max_a \sum_{s'} p(s'|s, a) \left[r(s, a) + \gamma v_k(s')\right]$$
- **Policy Iteration** performs fewer, but more expensive iterations consisting of:
  - Full *policy evaluation*, often via matrix inversion or iteration until convergence.
  - *Policy improvement*:
$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} p(s'|s, a) \left[r(s, a) + \gamma v_\pi(s')\right]$$

Convergence Rate and Number of Iterations

Since the Bellman optimality operator is a γ-contraction, we can bound the error at iteration k of value iteration by:

$$\|v_k - v_*\| \leq \gamma^k \|v_0 - v_*\|$$

To guarantee an error $\leq \epsilon$, we require:

$$\gamma^k \|v_0 - v_*\| \leq \epsilon$$

Taking logarithms:

$$k \geq \frac{\log(\epsilon / \|v_0 - v_*\|)}{\log(\gamma)} = \frac{\log(1/\epsilon) + \log(\|v_0 - v_*\|)}{\log(1/\gamma)}$$

Hence, the number of iterations is:

$$k = O\left(\frac{\log(1/\epsilon)}{\log(1/\gamma)}\right)$$

Use Cases and Practical Comparison

- **Value Iteration:**
  - Preferred in large state spaces.
  - Allows approximate solutions.
  - Faster per iteration but needs more iterations.
- **Policy Iteration:**
  - Fewer iterations to converge.
  - Expensive policy evaluation step.
  - Best suited for small-to-medium sized MDPs where exact convergence is feasible.

Summary

In summary, although exact convergence theoretically takes infinite time, we often settle for a good approximation. Once the value function is within a small ϵ of v_*, greedy policy extraction still yields a near-optimal policy. Also, note how the choice of γ affects convergence:

- Lower γ: fewer iterations, faster convergence.
- Higher γ: slower convergence, more emphasis on distant future rewards.

This tradeoff influences the design and efficiency of reinforcement learning systems.

References

[1] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.

[2] Farama Foundation. Gymnasium documentation. https://gymnasium.farama.org/, 2023. Maintained fork of OpenAI Gym.

[3] David Silver. Reinforcement learning course - deepmind ucl. https://deepmind.com/learning-resources, 2015. https://www.youtube.com/playlist?list=PLqYmG7hTraZDVH599EItlEWsUOsJbAodm.

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2020.