Theoretical Guarantees of Value-Based Reinforcement Learning: From Bellman Equations to Algorithmic Convergence

by Taha Majlesi

July 17, 2025

Abstract

This document provides a comprehensive exploration of the theoretical foundations of value-based Reinforcement Learning (RL). We begin by establishing the mathematical framework of Markov Decision Processes (MDPs), the formal language for sequential decision-making. We then derive the Bellman equations, which form the recursive basis for evaluating policies and finding optimal ones. The core of the theoretical guarantee is presented through the lens of fixed-point theory, proving that the Bellman optimality operator is a contraction mapping. This result, via the Banach Fixed-Point Theorem, guarantees the existence and uniqueness of an optimal solution. Finally, we analyze the classical dynamic programming algorithms—Value Iteration and Policy Iteration—that leverage these principles to compute optimal policies, and conclude with a synthesis of their properties.

Contents

1 The Mathematical Framework of Sequential Decision-Making

The pursuit of artificial intelligence that can learn and make optimal decisions in complex, dynamic environments is formalized through the field of Reinforcement Learning (RL). At its core, RL is a computational approach to learning from interaction, where an agent tries to maximize a cumulative reward signal by making a sequence of decisions. To analyze and solve such problems with mathematical rigor, we must first establish a formal framework. The **Markov Decision Process (MDP)** provides this precise mathematical language, serving as the bedrock upon which the entire theory of value-based reinforcement learning is constructed.

1.1 Defining the Markov Decision Process (MDP)

An MDP is the standard formalization of the sequential decision-making problem under uncertainty. It describes the interaction between a learning agent and its environment in terms of states, actions, and rewards. The fundamental premise is that an agent observes the state of its environment, selects an action, and the environment responds by transitioning to a new state and providing a scalar reward signal. This interactive loop continues, with the agent's objective being to learn a strategy, or policy, that maximizes the total reward accumulated over time.

Definition 1.1 (Markov Decision Process)**.** *A finite MDP is formally defined by the tuple* $(\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma)$*, where:*

- ***State Space*** $(\mathcal{S})$***:*** *A finite set of all possible states the agent can be in. A state, denoted* $s \in \mathcal{S}$*, is a complete description of the environment at a particular moment.*

  Example 1.2. In a game of chess, the state is the configuration of all pieces on the board. In a robotics task, it could be the joint angles and velocities of the robot's limbs.

- ***Action Space*** $(\mathcal{A})$***:*** *A finite set of all possible actions the agent can take. An action is denoted* $a \in \mathcal{A}$*. The set of actions available in a specific state* s *is denoted* $\mathcal{A}(s)$*.*

- ***Transition Dynamics*** (P)***:*** *This defines the environment's dynamics. It is a joint probability distribution over the next state* s' *and reward* r*, conditioned on the current state* s *and action* a*:*

$$p(s', r|s, a) = \Pr(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a) \tag{1}$$

  From this, we can derive:

  - ***State-Transition Probability:*** $P(s'|s, a) = \sum_{r\in\mathcal{R}} p(s', r|s, a)$
  - ***Expected Immediate Reward:*** $r(s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} r \cdot p(s', r|s, a)$

- ***Reward Function*** $(\mathcal{R})$***:*** *The reward function defines the goal. At each time step, the environment sends a scalar reward* r *to the agent. The agent's objective is to maximize the cumulative reward.*

- ***Discount Factor*** $(\gamma \in [0, 1))$***:*** *A scalar that determines the present value of future rewards. A reward received* k *time steps in the future is worth only* γ^{k-1} *times what it would be worth if it were received immediately.*

A core assumption of the MDP framework is the **Markov Property**. This property asserts that the future is independent of the past given the present. Formally, the probability of the next state and reward depends only on the current state and action, not on the entire history.

$$\Pr(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a) = \Pr(S_{t+1} = s', R_{t+1} = r|S_t, A_t, \ldots, S_0, A_0) \tag{2}$$

This assumption is what makes the problem tractable. The state s_t is a *sufficient statistic* of the history, enabling the recursive structure of the Bellman equations.

1.2 The Agent's Goal: The Return and the Role of the Discount Factor (γ)

The agent's goal is to maximize the cumulative reward, known as the **return**, G_t.

Definition 1.3 (Return)**.** *The return G_t is the sum of future rewards from time step t.*

- *For **episodic tasks** (with a terminal state at time T):*

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T \tag{3}$$

- *For **continuing tasks** (infinite horizon), we use discounting:*

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{4}$$

The discount factor γ has a profound, multifaceted role:

1. **Mathematical Necessity:** For infinite-horizon problems, $\gamma < 1$ ensures the sum of rewards remains finite if rewards are bounded. This is a prerequisite for the mathematical machinery of value functions to apply.
2. **Behavioral Preference:** The value of γ models the agent's preference for immediate versus future rewards.
   - $\gamma \to 0$: The agent is "myopic," prioritizing immediate rewards.
   - $\gamma \to 1$: The agent is "farsighted," valuing future rewards highly, enabling long-term planning.
3. **Probabilistic Interpretation:** γ can be seen as the probability of the process continuing. At each step, there is a $(1-\gamma)$ chance of termination. The agent's effective planning horizon is approximately $1/(1-\gamma)$.

1.3 Policies (π): The Agent's Blueprint for Behavior

A policy defines the agent's behavior. It is a mapping from states to actions.

Definition 1.4 (Policy)**.** *A policy π specifies the action to take in a given state.*

- ***Stochastic Policy,** $\pi(a|s)$**:** A probability distribution over actions for each state.*

$$\pi(a|s) = \Pr(A_t = a | S_t = s) \tag{5}$$

- ***Deterministic Policy,** $\pi(s)$**:** A direct mapping from a state to a single action.*

*The ultimate goal of an RL agent is to find an **optimal policy**, π^*, which maximizes the expected return from any starting state.*

2 Evaluating Policies with Value Functions and the Bellman Expectation Equation

To find an optimal policy, we must first be able to evaluate how "good" a given policy is. This is the role of **value functions**. They satisfy a fundamental recursive relationship known as the **Bellman expectation equation**.

2.1 The State-Value Function (v_π): How Good is a State?

The state-value function quantifies the long-term value of being in a state s while following policy π.

Definition 2.1 (State-Value Function)**.** *The state-value function $v_\pi(s)$ is the expected return starting from state s and then following policy π.*

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right] \tag{6}$$

2.2 The Action-Value Function (q_π): How Good is a State-Action Pair?

The action-value function (or Q-function) quantifies the value of taking a specific action a in a state s and then following policy π.

Definition 2.2 (Action-Value Function)**.** *The action-value function $q_\pi(s, a)$ is the expected return after taking action a in state s and then following policy π.*

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \tag{7}$$

The Q-function is crucial for control because it allows an agent to compare different actions within a state, which is necessary for improving its policy.

The two value functions are related:

$$v_\pi(s) = \sum_{a\in\mathcal{A}(s)} \pi(a|s) q_\pi(s, a) \tag{8}$$

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{9}$$

2.3 Derivation and Interpretation of the Bellman Expectation Equation

The Bellman expectation equation provides a recursive definition for value functions, decomposing the value of a state into the immediate reward and the discounted value of successor states.

Derivation for $v_\pi(s)$:

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

This gives the **Bellman expectation equation for v_π**:

$$v_\pi(s) = \sum_{a\in\mathcal{A}(s)} \pi(a|s) \left(r(s, a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s, a) v_\pi(s')\right) \tag{10}$$

Derivation for $q_\pi(s,a)$: A similar derivation, where the first action a is given, yields the **Bellman expectation equation for q_π**:

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \tag{11}$$
$$= \sum_{s',r} p(s', r|s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']] \tag{12}$$
$$= \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{13}$$

By substituting Eq. (8) into Eq. (13), we get a recursive form in terms of q_π:

$$q_\pi(s,a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') q_\pi(s', a') \right] \tag{14}$$

For a finite MDP, this equation defines a system of $|\mathcal{S}|$ linear equations that can be solved to find the value function for a given policy π. This process is called **policy evaluation**.

3 The Quest for Optimality: The Bellman Optimality Equation

The ultimate goal is to find the best policy. This requires the **Bellman optimality equation**, a non-linear equation that defines the value of behaving optimally.

3.1 Optimal Value Functions (v^* and q^*)

Definition 3.1 (Optimal Value Functions)**.** *The **optimal state-value function**, $v^*(s)$, is the maximum achievable expected return from state s.*

$$v^*(s) \doteq \max_\pi v_\pi(s) \quad \forall s \in \mathcal{S} \tag{15}$$

*The **optimal action-value function**, $q^*(s,a)$, is the maximum achievable expected return starting from state s, taking action a, and then following an optimal policy.*

$$q^*(s,a) \doteq \max_\pi q_\pi(s,a) \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{16}$$

The optimal value of a state is the value of taking the best action from that state:

$$v^*(s) = \max_{a \in \mathcal{A}(s)} q^*(s,a) \tag{17}$$

Once we have q^*, the optimal policy π^* is simply to act greedily with respect to it.

3.2 Deriving the Bellman Optimality Equation

The Bellman optimality equation is derived by incorporating a maximization over actions, embodying Bellman's **Principle of Optimality**.

Principle 3.2 (Principle of Optimality)**.** *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

Derivation for $v^*(s)$: Starting with $v^*(s) = \max_a q^*(s,a)$ and substituting the expression for $q^*(s,a)$ (which is Eq. (9) with an optimal policy, so $v_\pi \to v^*$):

$$v^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')] \tag{18}$$

Derivation for $q^*(s,a)$: Similarly, the value of taking action a and then behaving optimally is the expected reward plus the discounted *optimal* value of the next state.

$$q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a] \tag{19}$$

$$= \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')] \tag{20}$$

Substituting $v^*(s') = \max_{a'} q^*(s', a')$, we get the **Bellman optimality equation for q^***:

$$q^*(s,a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a' \in \mathcal{A}(s')} q^*(s', a') \right] \tag{21}$$

The 'max' operator makes this a system of non-linear equations, necessitating iterative solution methods.

3.3 The Bellman Optimality Operator ($\mathcal{T}^*$)

To analyze convergence, we define the Bellman optimality operator, $\mathcal{T}^*$, which maps a q-function to a new q-function.

Definition 3.3 (Bellman Optimality Operator)**.** *The operator $\mathcal{T}^*$ is defined as:*

$$(\mathcal{T}^* q)(s,a) \doteq \sum_{s',r} p(s', r|s, a) \left(r + \gamma \max_{a' \in \mathcal{A}(s')} q(s', a') \right) \tag{22}$$

The Bellman optimality equation can now be expressed as a **fixed-point problem**:

$$q^* = \mathcal{T}^* q^* \tag{23}$$

This states that the optimal action-value function q^* is a fixed point of the Bellman optimality operator. This reframing is crucial for proving the existence and uniqueness of a solution.

4 The Bedrock of Convergence: Contraction Mappings and Fixed-Point Theory

To guarantee that an optimal solution exists, is unique, and is findable, we turn to fixed-point theory. The **Banach Fixed-Point Theorem** provides the formal justification for why value-based RL algorithms work.

4.1 Preliminaries: Metric Spaces and the L-infinity Norm

The set of all bounded real-valued q-functions over the finite state-action space, $\mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$, forms a complete metric space. The distance metric used is the **L-infinity norm ($\|\cdot\|_\infty$)**.

Definition 4.1 (L-infinity Norm Distance)**.** *The distance between two q-functions, q_1 and q_2, is the maximum absolute difference between them over all state-action pairs:*

$$d(q_1, q_2) = \|q_1 - q_2\|_\infty \doteq \max_{s \in \mathcal{S}, a \in \mathcal{A}(s)} |q_1(s,a) - q_2(s,a)| \tag{24}$$

4.2 The Banach Fixed-Point Theorem

This theorem is the cornerstone of our theoretical analysis.

Theorem 4.2 (Banach Fixed-Point Theorem)**.** *Let (X, d) be a non-empty complete metric space and let $\mathcal{T} : X \to X$ be a* ***contraction mapping*** *on X. Then $\mathcal{T}$ has a unique fixed point $x^* \in X$ (i.e., $\mathcal{T}x^* = x^*$). Furthermore, the sequence $x, \mathcal{T}x, \mathcal{T}^2x, \dots$ generated by starting with an arbitrary point $x \in X$ and iteratively applying $\mathcal{T}$ converges to this unique fixed point x^*.*

Definition 4.3 (Contraction Mapping)**.** *A mapping $\mathcal{T} : X \to X$ is a contraction if there exists a constant $\alpha \in [0, 1)$ such that for all $x, y \in X$:*

$$d(\mathcal{T}x, \mathcal{T}y) \leq \alpha \cdot d(x, y) \tag{25}$$

If we can prove the Bellman optimality operator $\mathcal{T}^*$ is a contraction, we guarantee that a unique optimal solution q^* exists and that iterative algorithms will converge to it.

4.3 Proving the Bellman Optimality Operator is a Contraction Mapping

This proof is the linchpin of the entire theoretical argument. We will show that $\mathcal{T}^*$ is a γ-contraction with respect to the L-infinity norm.

$\mathcal{T}^*$ is a γ-Contraction. We want to prove that for any two bounded q-functions q_1 and q_2:

$$\|\mathcal{T}^*q_1 - \mathcal{T}^*q_2\|_\infty \leq \gamma \|q_1 - q_2\|_\infty$$

Let's start with the left-hand side:

$$\begin{aligned}
\|\mathcal{T}^*q_1 - \mathcal{T}^*q_2\|_\infty &= \max_{s,a} |(\mathcal{T}^*q_1)(s,a) - (\mathcal{T}^*q_2)(s,a)| \\
&= \max_{s,a} \left| \sum_{s',r} p(s', r|s, a) \left(r + \gamma \max_{a'} q_1(s', a') \right) - \sum_{s',r} p(s', r|s, a) \left(r + \gamma \max_{a'} q_2(s', a') \right) \right| \\
&= \max_{s,a} \left| \sum_{s',r} p(s', r|s, a) \left[\gamma \max_{a'} q_1(s', a') - \gamma \max_{a'} q_2(s', a') \right] \right| \\
&\leq \max_{s,a} \sum_{s',r} p(s', r|s, a) \left| \gamma \left(\max_{a'} q_1(s', a') - \max_{a'} q_2(s', a') \right) \right| \\
&= \gamma \max_{s,a} \sum_{s',r} p(s', r|s, a) \left| \max_{a'} q_1(s', a') - \max_{a'} q_2(s', a') \right|
\end{aligned}$$

Using the property that $|\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)|$:

$$\begin{aligned}
&\leq \gamma \max_{s,a} \sum_{s',r} p(s', r|s, a) \max_{a'} |q_1(s', a') - q_2(s', a')| \\
&\leq \gamma \max_{s,a} \sum_{s',r} p(s', r|s, a) \left(\max_{s'',a''} |q_1(s'', a'') - q_2(s'', a'')| \right) \\
&= \gamma \max_{s,a} \sum_{s',r} p(s', r|s, a) \|q_1 - q_2\|_\infty \\
&= \gamma \|q_1 - q_2\|_\infty \left(\max_{s,a} \sum_{s',r} p(s', r|s, a) \right) \\
&= \gamma \|q_1 - q_2\|_\infty
\end{aligned}$$

Since $\gamma < 1$, the Bellman optimality operator $\mathcal{T}^*$ is a contraction mapping with contraction constant γ. □

4.4 The Consequence: A Unique Optimal Value Function Exists

With the proof complete, the conclusion is direct and powerful. By the Banach Fixed-Point Theorem:

1. There exists a fixed point q^* such that $q^* = \mathcal{T}^* q^*$. This is the optimal action-value function.
2. This fixed point q^* is unique.

This result is the theoretical bedrock upon which value-based RL is built.

5 Dynamic Programming Algorithms for Finding the Optimal Policy

Dynamic Programming (DP) refers to algorithms that compute optimal policies given a perfect model of the MDP. The two primary DP methods are Value Iteration and Policy Iteration.

5.1 Value Iteration: Iteratively Applying the Bellman Optimality Backup

Value Iteration finds the optimal value function by iteratively applying the Bellman optimality equation as an update rule. It is a direct implementation of the iterative procedure guaranteed by the Banach Fixed-Point Theorem.

5.1.1 The Algorithm Step-by-Step

1. **Initialization:** Start with an arbitrary state-value function $V_0(s)$ for all $s \in \mathcal{S}$. (e.g., $V_0(s) = 0$ for all s).
2. **Iteration:** For each iteration $k = 0, 1, 2, \dots$, compute the next value function V_{k+1} from V_k for all states $s \in \mathcal{S}$:

$$V_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}(s)} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V_k(s') \right) \tag{26}$$

3. **Termination:** Stop when the value function converges, i.e., when the maximum change is smaller than a threshold ϵ: $\|V_{k+1} - V_k\|_\infty < \epsilon$.
4. **Policy Extraction:** Once the algorithm terminates with V^*, extract the optimal policy π^* by acting greedily:

$$\pi^*(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right) \tag{27}$$

5.1.2 Proof of Convergence

The convergence of Value Iteration is a direct consequence of the Contraction Mapping Theorem. The update rule is simply $V_{k+1} = \mathcal{T}^* V_k$. Since $\mathcal{T}^*$ is a γ-contraction, the sequence $\{V_0, V_1, V_2, \dots\}$ is guaranteed to converge to the unique optimal state-value function V^*.

5.2 Policy Iteration: The Dance of Evaluation and Improvement

Policy Iteration finds the optimal policy by alternating between two steps: evaluating the current policy and then improving it.

5.2.1 Step 1: Policy Evaluation

For a given policy π_k, compute its state-value function v_{π_k} by solving the Bellman expectation equation:

$$v_{\pi_k}(s) = r(s, \pi_k(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi_k(s)) v_{\pi_k}(s') \tag{28}$$

This is a system of $|\mathcal{S}|$ linear equations, which can be solved directly or, more commonly, via an iterative process that is itself guaranteed to converge.

5.2.2 Step 2: Policy Improvement and the Policy Improvement Theorem

Once v_{π_k} is known, form a new, better policy π_{k+1} by acting greedily with respect to it:

$$\pi_{k+1}(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) v_{\pi_k}(s') \right) \tag{29}$$

The guarantee that this new policy is better is provided by the Policy Improvement Theorem.

Theorem 5.1 (Policy Improvement Theorem)**.** *Let π and π' be two deterministic policies such that for all $s \in \mathcal{S}$, $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then the policy π' must be as good as, or better than, π. That is, $v_{\pi'}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$.*

This theorem guarantees that the new greedy policy π_{k+1} is a strict improvement over π_k, unless π_k is already optimal.

5.3 The Convergence of Policy Iteration

The algorithm alternates: $\pi_0 \xrightarrow{\text{Eval}} v_{\pi_0} \xrightarrow{\text{Improve}} \pi_1 \xrightarrow{\text{Eval}} v_{\pi_1} \xrightarrow{\text{Improve}} \cdots \rightarrow \pi^*$. Convergence is guaranteed for a finite MDP because:

1. **Monotonic Improvement:** Each policy is strictly better than the last (unless optimal).
2. **Finite Number of Policies:** For a finite MDP, there are a finite number of deterministic policies ($|\mathcal{A}|^{|\mathcal{S}|}$).

Since each iteration finds a better policy, and there are a finite number of them, the algorithm must terminate at the optimal policy in a finite number of iterations.

6 A Comparative Synthesis of Dynamic Programming Methods

6.1 Generalized Policy Iteration (GPI): The Unifying Theme

Generalized Policy Iteration (GPI) is the general idea of letting policy evaluation and policy improvement processes interact. An agent maintains a policy and a value function. The value function is updated to be consistent with the policy (evaluation), and the policy is updated to be greedy with respect to the value function (improvement).

- **Policy Iteration** performs a full, complete policy evaluation in each loop.
- **Value Iteration** can be seen as a version of GPI where the evaluation step is truncated to just one Bellman backup.

This interplay is the core mechanism of nearly all value-based RL algorithms.

Table 1: Comparison of Value Iteration and Policy Iteration

Feature	Value Iteration	Policy Iteration
Core Idea	Iterates on the value function until it converges to V^*.	Alternates between evaluating and improving the policy until it converges to π^*.
Main Update	Bellman Optimality Backup: $V_{k+1} = \mathcal{T}^* V_k$.	Bellman Expectation Backup (Eval) + Greedy Update (Improvement).
Complexity per Iteration	Lower per iteration: $O(\lvert\mathcal{A}\rvert\lvert\mathcal{S}\rvert^2)$.	Higher per iteration: Policy evaluation is expensive (e.g., $O(\lvert\mathcal{S}\rvert^3)$).
Typical # of Iterations	More iterations. Convergence can be slow as it drives V to its limit.	Fewer iterations. Often converges very quickly.
Convergence Guarantee	Yes, via the Contraction Mapping Theorem.	Yes, via the Policy Improvement Theorem and the finite number of policies.
When to Prefer	When a full policy evaluation is too expensive; simpler implementation.	When faster convergence (in iterations) is critical and evaluation is efficient.

6.2 A Detailed Comparison of Value Iteration and Policy Iteration

The choice between the two involves a trade-off between the number of iterations and the computational cost per iteration.

7 Conclusion: The Enduring Legacy of Theoretical Guarantees

This report has navigated the theoretical foundations that underpin value-based reinforcement learning. The central conclusion is that RL methods are not merely effective heuristics; they are principled algorithms grounded in the solid bedrock of mathematical analysis.

The Bellman optimality equation frames the problem as a search for a function satisfying a specific recursive condition. The application of the **Banach Fixed-Point Theorem** is the theoretical linchpin, proving that the Bellman optimality operator is a contraction mapping. This guarantees that a **unique optimal value function exists** and that iterative algorithms like **Value Iteration** will provably converge to it. Similarly, the **Policy Improvement Theorem** guarantees the monotonic improvement and eventual convergence of **Policy Iteration**.

Finally, the analysis illuminates the unified role of the discount factor, γ. It is simultaneously a tool for mathematical convergence, a parameter for modeling an agent's behavior, and the contraction constant governing algorithmic convergence speed. This elegant unification of mathematical, behavioral, and algorithmic properties is a testament to the power and coherence of the MDP framework.