Optimality Equations and the Principle of Optimality

Reinforcement Learning

Sai Sampath Kedari
sampath@umich.edu

Contents

1 Optimality Equations and the Principle of Optimality

In this section we introduce the *optimality equations* (sometimes referred to as Bellman equations or functional equations) and state their fundamental properties. These equations characterize the optimal value functions of a finite-horizon Markov decision process and form the basis for computing and verifying optimal policies.

Throughout this section, we assume that the state space S is finite (or countable), so that no technical measurability issues arise.

1.1 Optimal Value Functions

Recall that for a fixed policy $\pi \in \Pi^{\mathrm{HR}}$, the value function

$$u_t^\pi(h_t) = \mathbb{E}^\pi \left[\sum_{n=t}^{N-1} r_n(X_n, Y_n) + r_N(X_N) \;\middle|\; H_t = h_t \right] \tag{4.2.1}$$

represents the expected total reward obtained from decision epoch t onward, given that the history up to time t is h_t and that policy π is followed thereafter.

The *optimal value function* at decision epoch t is defined by

$$u_t^*(h_t) := \sup_{\pi \in \Pi^{\mathrm{HR}}} u_t^\pi(h_t). \tag{4.3.1}$$

This quantity represents the maximum expected total reward that can be achieved from epoch t onward when the system has reached history h_t.

For $t > 1$, it is not necessary to take the supremum over entire policies from epoch 1 onward. Since the history h_t is already fixed, only the portion of the policy from decision epoch t onward is relevant. Thus, the supremum in (4.3.1) may equivalently be taken over

$$(d_t, d_{t+1}, \ldots, d_{N-1}) \in D_t^{\mathrm{HR}} \times D_{t+1}^{\mathrm{HR}} \times \cdots \times D_{N-1}^{\mathrm{HR}}.$$

Equation (4.3.1) therefore provides a precise but abstract definition of optimality. A direct approach to computing u_t^* would be to enumerate all admissible policies, evaluate $u_t^\pi(h_t)$ for each policy using policy evaluation, and then select the largest value. This approach is computationally infeasible and serves only as a conceptual definition.

1.2 Optimality Equations

The optimal value functions satisfy the following system of equations, called the *optimality equations*.

For $t = 1, 2, \ldots, N-1$ and for every history $h_t \in H_t$ with current state s_t,

$$u_t(h_t) = \max_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j \mid s_t, a)\, u_{t+1}(h_t, a, j) \right\}, \qquad t = 1, 2, \ldots, N-1. \tag{4.3.4}$$

At the terminal epoch, the boundary condition is

$$u_N(h_N) = r_N(s_N). \tag{4.3.3}$$

where s_N denotes the terminal state embedded in h_N.

A *solution* to the system (4.3.4)–(4.3.3) is a sequence of functions

$$u_t : H_t \to \mathbb{R}, \qquad t = 1, 2, \ldots, N,$$

satisfying the terminal condition at $t = N$ and the recursive relation for all $t < N$.

At this stage, the optimality equations are stated without proof. Their significance lies in the fact that they characterize the optimal value functions defined in (4.3.1).

1.3 Illustrative Example

We illustrate the structure and interpretation of the optimality equations by expanding a simple finite-horizon example and explicitly writing out the corresponding system of equations.

Consider a finite-horizon Markov decision process with

$$S = \{s_1, s_2\}, \qquad A_{s_1} = A_{s_2} = \{a_1, a_2\},$$

and horizon $N = 2$. Thus, there is a single decision epoch at $t = 1$, followed by a terminal epoch at $t = 2$.

We work throughout with history-dependent randomized (HR) policies, so value functions are defined on the full history spaces.

Histories at the terminal epoch. At epoch $t = 2$, histories consist of a state at $t = 1$, an action taken at $t = 1$, and the resulting state at $t = 2$. Hence,

$$\begin{aligned} H_2 = \{&(s_1, a_1, s_1), (s_1, a_1, s_2), (s_1, a_2, s_1), (s_1, a_2, s_2), \\ &(s_2, a_1, s_1), (s_2, a_1, s_2), (s_2, a_2, s_1), (s_2, a_2, s_2)\}. \end{aligned}$$

At the terminal epoch, the boundary condition of the optimality equations is

$$u_2(h_2) = r_2(s_2). \tag{4.3.3}$$

where s_2 denotes the terminal state embedded in the history h_2.

Thus, explicitly,

$$\begin{aligned} &u_2(s_1, a_1, s_1) = r_2(s_1), \quad u_2(s_1, a_1, s_2) = r_2(s_2), \\ &u_2(s_1, a_2, s_1) = r_2(s_1), \quad u_2(s_1, a_2, s_2) = r_2(s_2), \\ &u_2(s_2, a_1, s_1) = r_2(s_1), \quad u_2(s_2, a_1, s_2) = r_2(s_2), \\ &u_2(s_2, a_2, s_1) = r_2(s_1), \quad u_2(s_2, a_2, s_2) = r_2(s_2). \end{aligned}$$

Histories at the decision epoch. At epoch $t = 1$, histories consist only of the initial state:

$$H_1 = \{s_1, s_2\}.$$

Optimality equations at $t = 1$. For each $h_1 = s_i \in H_1$, the optimality equation is

$$u_1(s_i) = \max_{a \in \{a_1, a_2\}} \left\{ r_1(s_i, a) + \sum_{j \in S} p_1(j \mid s_i, a)\, u_2(s_i, a, j) \right\}. \tag{4.3.4}$$

Substituting the terminal values from (4.3.3), we obtain

$$u_1(s_1) = \max_{a \in \{a_1, a_2\}} \left\{ r_1(s_1, a) + \sum_{j \in S} p_1(j \mid s_1, a)\, r_2(j) \right\},$$

$$u_1(s_2) = \max_{a \in \{a_1, a_2\}} \left\{ r_1(s_2, a) + \sum_{j \in S} p_1(j \mid s_2, a)\, r_2(j) \right\}.$$

Interpretation. This example shows explicitly how the optimality equations define a system of equations over all possible histories. The terminal values are fixed by the terminal reward function, and the values at earlier epochs are obtained by maximizing, at each history, the sum of immediate reward and expected future value.

Rather than enumerating all admissible policies and evaluating their returns, one solves this system of equations backward in time. The solution yields the optimal value function at every history and implicitly determines optimal actions through the maximization operators.

1.4 Fundamental Properties of the Optimality Equations

The optimality equations are the central object of finite-horizon Markov decision theory. They connect three tasks that arise repeatedly in practice: (i) defining what it means to be optimal, (ii) checking whether a given policy is optimal, and (iii) computing optimal value functions and policies efficiently. They also reveal important structural properties of optimal solutions.

The following four statements summarize their role and meaning.

(a) **Characterization of optimal returns.**

The optimality equations (4.3.3)–(4.3.4) provide an alternative and computationally tractable characterization of the optimal value functions defined abstractly by

$$u_t^*(h_t) = \sup_{\pi \in \Pi^{\mathrm{HR}}} u_t^{\pi}(h_t), \qquad t = 1, \ldots, N.$$

Rather than computing u_t^* by enumerating all admissible policies and comparing their expected returns, one may instead solve the system of optimality equations backward in time. Any sequence of functions $\{u_t\}_{t=1}^N$ that satisfies the optimality equations automatically coincides with the optimal value functions $\{u_t^*\}_{t=1}^N$.

In this sense, the optimality equations do not merely approximate optimal returns; they exactly encode the supremum over all policies in (4.3.1) in a recursive form. This equivalence is the fundamental idea underlying dynamic programming.

(b) **Verification of optimality.**

Suppose a policy π is given and we wish to determine whether it is optimal. A direct approach would be to enumerate all policies, compute their value functions, and compare them to u^π. This is clearly infeasible.

The optimality equations provide a much simpler test. First, compute the value functions $\{u_t^\pi\}_{t=1}^N$ of the given policy using policy evaluation and backward induction. If these functions satisfy the optimality equations (4.3.3)–(4.3.4) at every decision epoch and history, then π achieves the supremum in (4.3.1) and is therefore an optimal policy.

Thus, optimality can be verified by checking whether the evaluated value functions satisfy the optimality equations, without ever comparing π to other policies.

(c) **Basis for computing optimal policies and value functions.**

If the goal is to find an optimal policy from scratch, the optimality equations provide a direct computational procedure. Starting from the terminal condition (4.3.3), one solves the equations backward in time. At each history h_t, the maximization in (4.3.4) determines both the optimal value $u_t^*(h_t)$ and the action(s) that attain the maximum.

By selecting, at each epoch, an action that achieves this maximum, one constructs an optimal policy. This procedure avoids enumeration of policies entirely and replaces a global optimization problem with a sequence of local maximizations. It is this backward induction algorithm that makes finite-horizon Markov decision problems computationally tractable.

(d) **Structural implications.**

The form of the optimality equations has important structural consequences. Although the value functions are defined on full histories and policies are allowed to be history-dependent and randomized, the maximization in (4.3.4) depends only on the current state s_t.

As a result, history dependence and randomization are not required for optimality. One may restrict attention to Markov deterministic policies without loss of optimality. More generally, the optimality equations can be used to derive additional structural properties of optimal value functions and policies, such as monotonicity or threshold behavior in specific models.

These structural results are not assumptions but consequences of the optimality equations themselves, and they will be developed in the remainder of this chapter.

Understanding these four properties is sufficient to understand the role of the optimality equations. The remainder of this chapter is devoted to proving these statements rigorously and exploring their consequences.

2 Fundamental Results for the Optimality Equations

In this section we establish several key theoretical results that justify the optimality equations introduced in Section 4.3. These results show that the optimality equations characterize optimal value functions, provide a method for verifying optimality of a given policy, and form the basis for constructing optimal policies.

We begin with a simple but fundamental inequality that underlies the principle of optimality and explains the appearance of maximization operators in the optimality equations.

2.1 Lemma 4.3.1 (Supremum Dominates Expectation)

Lemma (4.3.1)**.** *Let W be a finite or countable set, let $w : W \to \mathbb{R}$ be a real-valued function, and let $q(\cdot)$ be a probability distribution on W. Then*

$$\sum_{x \in W} q(x)\, w(x) \;\leq\; \sup_{x \in W} w(x). \tag{4.3.5}$$

If the supremum is attained, the right-hand side may be replaced by a maximum.

Remark (4.3.1 — Interpretation)**.** *Lemma 4.3.1 states that the expected value of a function under any probability distribution cannot exceed the largest value that the function attains. Equivalently, randomization cannot improve upon the best deterministic choice.*

Proof. By definition of the supremum,

$$\sup_{x \in W} w(x) \;\geq\; w(x), \qquad \forall\, x \in W.$$

Since $q(x) \geq 0$ for all $x \in W$, multiplying both sides by $q(x)$ preserves the inequality:

$$q(x) \sup_{x \in W} w(x) \;\geq\; q(x)\, w(x), \qquad \forall\, x \in W.$$

Summing over $x \in W$ yields

$$\sum_{x \in W} q(x) \sup_{x \in W} w(x) \;\geq\; \sum_{x \in W} q(x)\, w(x).$$

Because $\sup_{x \in W} w(x)$ does not depend on x, it may be factored out:

$$\sup_{x \in W} w(x) \sum_{x \in W} q(x) \;\geq\; \sum_{x \in W} q(x)\, w(x).$$

Since $q(\cdot)$ is a probability distribution, $\sum_{x \in W} q(x) = 1$, and therefore

$$\sup_{x \in W} w(x) \;\geq\; \sum_{x \in W} q(x)\, w(x).$$

□

Remark (4.3.2 — Role of Lemma 4.3.1 in the Optimality Equations)**.** *Lemma 4.3.1 is the basic inequality underlying the principle of optimality. It shows that, at any fixed history, the expected return obtained by randomizing over actions is bounded above by the return obtained by selecting an action that maximizes the immediate reward plus expected future value. This fact justifies the maximization operator in the optimality equations (4.3.4) and explains why deterministic action choices suffice when constructing optimal policies.*

2.2 Theorem 4.3.2 (Characterization of Optimal Value Functions)

Theorem (4.3.2)**.** *Suppose* $\{u_t\}_{t=1}^N$ *is a solution of the optimality equations*

$$u_N(h_N) = r_N(s_N),$$

and for $t = 1, \ldots, N-1$,

$$u_t(h_t) = \sup_{a \in A_{s_t}} \left(r_t(s_t, a) + \sum_{j \in S} p_t(j \mid s_t, a)\, u_{t+1}(h_t, a, j) \right).$$

Then:

(a) $u_t(h_t) = u_t^*(h_t)$ *for all* $h_t \in H_t$, $t = 1, \ldots, N$.

(b) $u_1(s_1) = v_N^*(s_1)$ *for all* $s_1 \in S$.

Proof. **What are we trying to prove and why?**

We have three objects in play:

1. For any policy $\pi \in \Pi^{HR}$, the policy value functions

$$u_t^\pi(h_t), \quad h_t \in H_t.$$

2. The optimal value functions defined by

$$u_t^*(h_t) := \sup_{\pi \in \Pi^{HR}} u_t^\pi(h_t).$$

3. A sequence of functions $\{u_t\}_{t=1}^N$ that satisfies the optimality equations.

The theorem claims that any solution of the Bellman optimality equations must coincide with the supremum over all policies. In other words, solving the Bellman equations is equivalent to solving the original optimization problem over policies.

To prove

$$u_t(h_t) = u_t^*(h_t),$$

it is enough to prove two inequalities:

1. $u_t(h_t) \geq u_t^*(h_t)$,
2. $u_t(h_t) \leq u_t^*(h_t)$.

We begin with the first inequality.

2.2.1 Part 1: Upper Bound $u_n(h_n) \geq u_n^*(h_n)$

Prove $u_n(h_n) \geq u_n^*(h_n)$ for all n.

The goal is to show that for every time n and history h_n,

$$u_n(h_n) \geq u_n^\pi(h_n) \quad \forall \pi,$$

which implies

$$u_n(h_n) \geq \sup_\pi u_n^\pi(h_n) = u_n^*(h_n).$$

We prove this by backward induction.

Step 1: Base case $n = N$.

At the terminal time N, there is no decision left. Every policy yields

$$u_N^\pi(h_N) = r_N(s_N).$$

Since $\{u_t\}$ satisfies the boundary condition,

$$u_N(h_N) = r_N(s_N).$$

Therefore for all π,

$$u_N(h_N) = u_N^\pi(h_N),$$

and hence

$$u_N(h_N) \geq u_N^*(h_N),$$

with equality.

Step 2: Induction hypothesis.

Assume that for some $n < N$,

$$u_t(h_t) \geq u_t^*(h_t) \quad \text{for all } h_t \in H_t \quad \text{and all } t = n+1, \ldots, N.$$

That is,

$$\begin{aligned} u_{n+1}(h_{n+1}) &\geq u_{n+1}^*(h_{n+1}) \quad \forall h_{n+1} \in H_{n+1}, \\ u_{n+2}(h_{n+2}) &\geq u_{n+2}^*(h_{n+2}) \quad \forall h_{n+2} \in H_{n+2}, \\ &\vdots \\ u_N(h_N) &\geq u_N^*(h_N). \end{aligned}$$

Step 3: Write the optimality equation at time n.

Since u_n satisfies the optimality equation,

$$u_n(h_n) = \sup_{a \in A_{s_n}} \left(r_n(s_n, a) + \sum_{j \in S} p_n(j \mid s_n, a) u_{n+1}(h_n, a, j) \right).$$

Step 4: Use the induction hypothesis inside the expectation term.

For each next state j, the history (h_n, a, j) belongs to H_{n+1}. By the induction hypothesis,

$$u_{n+1}(h_n, a, j) \geq u^*_{n+1}(h_n, a, j) = \sup_{\pi \in \Pi^{HR}} u^{\pi}_{n+1}(h_n, a, j) \geq u^{\pi'}_{n+1}(h_n, a, j),$$

for any arbitrary policy π'.

Multiply both sides by $p_n(j \mid s_n, a) \geq 0$ and sum over j:

$$\sum_j p_n(j \mid s_n, a) u_{n+1}(h_n, a, j) \geq \sum_j p_n(j \mid s_n, a) u^{\pi'}_{n+1}(h_n, a, j).$$

Add $r_n(s_n, a)$ to both sides:

$$r_n(s_n, a) + \sum_j p_n(j \mid s_n, a) u_{n+1}(h_n, a, j) \geq r_n(s_n, a) + \sum_j p_n(j \mid s_n, a) u^{\pi'}_{n+1}(h_n, a, j).$$

Since this inequality holds for each $a \in A_{s_n}$, taking the supremum over a preserves the inequality:

$$u_n(h_n) \geq \sup_{a \in A_{s_n}} \left(r_n(s_n, a) + \sum_j p_n(j \mid s_n, a) u^{\pi'}_{n+1}(h_n, a, j) \right).$$

Step 5: Apply Lemma 4.3.1 (sup dominates expectation).

Let

$$w(a) := r_n(s_n, a) + \sum_j p_n(j \mid s_n, a) u^{\pi'}_{n+1}(h_n, a, j).$$

Policy π' at history h_n chooses a distribution $q_{d_n(h_n)}(\cdot)$ over A_{s_n}.

By Lemma 4.3.1,

$$\sup_a w(a) \geq \sum_a q_{d_n(h_n)}(a) w(a).$$

Therefore,

$$u_n(h_n) \geq \sum_{a \in A_{s_n}} q_{d_n(h_n)}(a) \left(r_n(s_n, a) + \sum_j p_n(j \mid s_n, a) u^{\pi'}_{n+1}(h_n, a, j) \right).$$

But the right-hand side is exactly the policy evaluation recursion for π':

$$u^{\pi'}_n(h_n).$$

Hence

$$u_n(h_n) \geq u^{\pi'}_n(h_n).$$

Since π' was arbitrary,

$$u_n(h_n) \geq \sup_{\pi'} u_n^{\pi'}(h_n) = u_n^*(h_n).$$

This completes Part 1.

Interpretation of Part 1.

Any function that satisfies the Bellman optimality recursion is an upper bound on what any policy can achieve. At each step it chooses the best possible action, whereas a policy may randomize or choose suboptimal actions. Therefore it cannot perform worse than any policy.

□

2.3 Part 2: The Bellman Solution is the Least Upper Bound

In Part 1 we proved that for every n and every $h_n \in H_n$,

$$u_n(h_n) \geq u_n^{\pi}(h_n) \quad \forall \pi \in \Pi^{HR}.$$

Hence $u_n(h_n)$ is an upper bound for the set

$$\mathcal{A}(n, h_n) = \{u_n^{\pi}(h_n) : \pi \in \Pi^{HR}\}.$$

Equivalently,

$$u_n(h_n) \geq u_n^*(h_n) = \sup_{\pi \in \Pi^{HR}} u_n^{\pi}(h_n).$$

To conclude that

$$u_n(h_n) = u_n^*(h_n),$$

it remains to prove that $u_n(h_n)$ is the *least* upper bound of this set. That is, we must show

$$u_n^*(h_n) \geq u_n(h_n).$$

2.3.1 Strategy via the ε-Characterization of Supremum

Recall the basic characterization of the supremum:

A number s is the supremum of a set A if and only if:

1. s is an upper bound of A, and
2. for every $\varepsilon > 0$, there exists $a \in A$ such that

$$a \geq s - \varepsilon.$$

Property (1) has already been established in Part 1. Thus, to prove that $u_n(h_n)$ equals the supremum of achievable policy values, it suffices to show that for every $\varepsilon > 0$, there exists a policy π^{ε} such that

$$u_n^{\pi^{\varepsilon}}(h_n) \geq u_n(h_n) - (\text{small error}).$$

Because the horizon is finite and consists of $N - n$ remaining decision stages, the small error will accumulate to $(N - n)\varepsilon$.

Step 1: Using the Supremum over Actions From the optimality equation,

$$u_n(h_n) = \sup_{a \in A_{s_n}} \left\{ r_n(s_n, a) + \sum_{j \in S} p_n(j \mid s_n, a)\, u_{n+1}(h_n, a, j) \right\}.$$

Since the supremum is taken over actions, by definition of supremum, for every $\varepsilon > 0$, there exists an action $a^\varepsilon \in A_{s_n}$ such that

$$r_n(s_n, a^\varepsilon) + \sum_{j \in S} p_n(j \mid s_n, a^\varepsilon)\, u_{n+1}(h_n, a^\varepsilon, j) \geq u_n(h_n) - \varepsilon. \tag{4.3.9}$$

For each (n, h_n) choose one such action and define

$$d_n^\varepsilon(h_n) := a^\varepsilon.$$

This defines a deterministic history-dependent decision rule

$$d_n^\varepsilon : H_n \to A_{s_n}.$$

Doing this for all $n = 1, \ldots, N-1$ produces a deterministic history-dependent policy

$$\pi^\varepsilon = (d_1^\varepsilon, \ldots, d_{N-1}^\varepsilon) \in \Pi^{HD}.$$

Step 2: Backward Induction on the Error Bound We now prove by backward induction that

$$u_n^{\pi^\varepsilon}(h_n) \geq u_n(h_n) - (N-n)\varepsilon \quad \forall n, \forall h_n. \tag{4.3.8}$$

Base Case ($n = N$). At the terminal epoch,

$$u_N^{\pi^\varepsilon}(h_N) = r_N(s_N) = u_N(h_N).$$

Since $(N-N)\varepsilon = 0$, the bound holds.

Inductive Step. Assume that for some $n < N$,

$$u_{n+1}^{\pi^\varepsilon}(h_{n+1}) \geq u_{n+1}(h_{n+1}) - (N-n-1)\varepsilon \quad \forall h_{n+1}.$$

Under π^ε, the action at h_n is $d_n^\varepsilon(h_n)$. Hence, by policy evaluation recursion,

$$u_n^{\pi^\varepsilon}(h_n) = r_n(s_n, d_n^\varepsilon(h_n)) + \sum_{j \in S} p_n(j \mid s_n, d_n^\varepsilon(h_n))\, u_{n+1}^{\pi^\varepsilon}(h_n, d_n^\varepsilon(h_n), j).$$

Applying the induction hypothesis to each next-history,

$$u_{n+1}^{\pi^\varepsilon}(\cdot) \geq u_{n+1}(\cdot) - (N-n-1)\varepsilon,$$

and summing over j gives

$$\sum_j p_n(\cdot) u_{n+1}^{\pi^\varepsilon}(\cdot) \geq \sum_j p_n(\cdot) u_{n+1}(\cdot) - (N-n-1)\varepsilon.$$

Therefore,

$$u_n^{\pi^\varepsilon}(h_n) \geq r_n(s_n, d_n^\varepsilon(h_n)) + \sum_j p_n(j \mid s_n, d_n^\varepsilon(h_n)) u_{n+1}(h_n, d_n^\varepsilon(h_n), j) - (N-n-1)\varepsilon.$$

Using inequality (4.3.9),

$$u_n^{\pi^\varepsilon}(h_n) \geq u_n(h_n) - \varepsilon - (N-n-1)\varepsilon = u_n(h_n) - (N-n)\varepsilon.$$

This completes the induction.

Step 3: Concluding the Supremum Argument Since $u_n^*(h_n)$ is the supremum over all policies,

$$u_n^*(h_n) = \sup_\pi u_n^\pi(h_n) \geq u_n^{\pi^\varepsilon}(h_n).$$

Hence,

$$u_n^*(h_n) \geq u_n(h_n) - (N-n)\varepsilon.$$

Letting $\varepsilon \downarrow 0$,

$$u_n^*(h_n) \geq u_n(h_n).$$

Combining this with Part 1,

$$u_n(h_n) \geq u_n^*(h_n),$$

we conclude

$$u_n(h_n) = u_n^*(h_n) \quad \forall n, \forall h_n.$$

This proves that the solution of the optimality equations coincides with the optimal value function.

2.4 Theorem 4.3.3 (Construction and Verification of Optimal Policies)

Suppose u_t^*, $t = 1, \ldots, N$ are solutions of the optimality equations (4.3.4) subject to the boundary condition (4.3.3), and suppose a deterministic history-dependent policy

$$\pi^* = (d_1^*, d_2^*, \ldots, d_{N-1}^*) \in \Pi^{\text{HD}}$$

satisfies, for each $t = 1, \ldots, N-1$ and each $h_t \in H_t$,

$$r_t(s_t, d_t^*(h_t)) + \sum_{j \in S} P_t(j \mid s_t, d_t^*(h_t))\, u_{t+1}^*(h_t, d_t^*(h_t), j) = \max_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} P_t(j \mid s_t, a)\, u_{t+1}^*(h_t, a, j) \right\}. \tag{4.3.10}$$

Then:

(a) For each $t = 1, \ldots, N$,

$$u_t^{\pi^*}(h_t) = u_t^*(h_t), \qquad h_t \in H_t. \tag{4.3.11}$$

(b) π^* is an optimal policy and

$$v_N^{\pi^*}(s) = v_N^*(s), \qquad s \in S. \tag{4.3.12}$$

2.5 Interpretation and Algorithmic Meaning

Equation (4.3.10) states that at every history h_t, the decision rule $d_t^*(h_t)$ selects an action that attains the maximum in the Bellman optimality equation (4.3.4).

Therefore, an optimal policy can be constructed by the following **greedy backward procedure**:

1. Solve the optimality equations (4.3.4) with boundary condition (4.3.3) to obtain u_t^* for all t.
2. For each stage t and each history h_t, choose

$$d_t^*(h_t) \in \arg\max_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} P_t(j \mid s_t, a) u_{t+1}^*(h_t, a, j) \right\}.$$

3. Define

$$\pi^* = (d_1^*, \ldots, d_{N-1}^*).$$

Because maxima are attained, such a policy exists.

This theorem also provides a *verification principle*: if a policy satisfies (4.3.10), then it must be optimal.

2.6 The Principle of Optimality

Theorem 4.3.3 gives a formal statement of the *Principle of Optimality*, the central idea of dynamic programming.

Bellman (1957):

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with respect to the state resulting from the first decision.

Denardo (1982):

> There exists a policy that is optimal for every state (at every stage).

Mathematically, the principle states:

If a policy is optimal from stage t onward at history h_t, then for any next history (h_t, a, j) that occurs under that policy, the continuation policy must be optimal for that subproblem.

In other words, optimality is preserved under truncation. The tail of an optimal policy is itself optimal.

This recursive self-consistency is precisely what is encoded in the Bellman optimality equation (4.3.4), and Theorem 4.3.3 shows that selecting maximizing actions at each stage produces such a self-consistent policy.

Important Remark. The Principle of Optimality may fail under other optimality criteria. Its validity here relies on the expected total reward formulation and the finite-horizon structure.