Lecture 3: Value Iteration

From Theory to Algorithm

In the previous lectures, we established the theoretical foundations for optimal decision-making in MDPs. We proved that the Bellman optimality operator $\mathcal{T}^\star$ is a γ-contraction, and by the Banach Fixed Point Theorem, repeatedly applying $\mathcal{T}^\star$ to any initial value function converges to the unique optimal value function $V^\star$. This theoretical result immediately suggests an algorithm: simply iterate $V_k = \mathcal{T}^\star V_{k-1}$ until convergence.

The Computational Question: We know $V^\star$ exists and is unique. We know how to characterize it via the Bellman optimality equation. But how do we actually *compute* it? This lecture and the next present two fundamental algorithms: **Value Iteration** (this lecture) and **Policy Iteration** (next lecture).

Planning vs. Learning: In this lecture (and the next), we assume the MDP model—the transition function $\mathcal{P}$ and reward function R—is **fully known**. This setting is called **planning**: given complete knowledge of the environment dynamics, we compute the optimal policy. In contrast, **reinforcement learning** addresses the case where the model is unknown and must be learned from interaction data. We will cover learning algorithms in later lectures.

Recall the key objects from the previous lecture:

- The Bellman optimality operator: $(\mathcal{T}^\star V)(s) \coloneqq \max_{a\in\mathcal{A}} \left(R(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[V(s')]\right)$
- The optimal value function $V^\star$ satisfies $V^\star = \mathcal{T}^\star V^\star$
- $\mathcal{T}^\star$ is a γ-contraction: $\|\mathcal{T}^\star U - \mathcal{T}^\star V\|_\infty \le \gamma \|U - V\|_\infty$

Key Insight. The Banach Fixed Point Theorem tells us that for any V_0, the sequence $V_1 = \mathcal{T}^\star V_0$, $V_2 = \mathcal{T}^\star V_1$, ... converges to $V^\star$. This iteration *is* the algorithm!

The Value Iteration Algorithm

Value Iteration directly implements the Banach fixed point iteration for the Bellman optimality operator.

Algorithm 1 Value Iteration

Require: MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, tolerance $\varepsilon > 0$

Initialize $V_0(s) \leftarrow 0$ for all $s \in \mathcal{S}$
for $k = 1, 2, \ldots$ **do**
  for each $s \in \mathcal{S}$ **do**
    $V_k(s) \leftarrow \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s, a)\, V_{k-1}(s') \right)$ ▷ *Bellman backup*
  end for
  if $\|V_k - V_{k-1}\|_\infty < \dfrac{\varepsilon(1-\gamma)}{\gamma}$ **then** ▷ *Stopping criterion*
    break
  end if
end for
return V_k and greedy policy $\pi(s) \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s'} \mathcal{P}(s' \mid s, a) V_k(s') \right)$

The algorithm has a simple structure: repeatedly apply the Bellman optimality operator until the value function stabilizes. In operator notation:

$$V_k = \mathcal{T}^\star V_{k-1} \quad (k \geq 1)$$

Remarks on Initialization.

- **Any initialization works:** The contraction property guarantees convergence from any starting point.
- **Zero initialization ($V_0 = 0$):** Simple and commonly used. Provides a lower bound on $V^\star$ when rewards are non-negative.
- **Optimistic initialization ($V_0 = R_{\max}/(1-\gamma)$):** May speed up convergence in some cases by starting from an upper bound.
- **Warm-starting:** If solving a modified MDP (e.g., slightly changed rewards), initializing with the previous solution can reduce iterations.

Convergence Analysis

The convergence of Value Iteration follows directly from the contraction property of $\mathcal{T}^\star$ and the Banach Fixed Point Theorem. We now make this precise and derive the iteration complexity.

Theorem 1 (Value Iteration Convergence)**.** *Let $V_0 \in \mathcal{B}(\mathcal{S})$ be arbitrary and define $V_k = \mathcal{T}^\star V_{k-1}$ for $k \geq 1$. Then:*

1. *(Linear convergence)* $\|V_k - V^\star\|_\infty \le \gamma^k \|V_0 - V^\star\|_\infty$
2. *(Absolute bound) When* $V_0 = 0$*:* $\|V_k - V^\star\|_\infty \le \gamma^k \cdot \dfrac{R_{\max}}{1-\gamma}$

Proof. **Part 1.** Since $\mathcal{T}^\star$ is a γ-contraction and $V^\star = \mathcal{T}^\star V^\star$:

$$\|V_k - V^\star\|_\infty = \|\mathcal{T}^\star V_{k-1} - \mathcal{T}^\star V^\star\|_\infty \le \gamma \|V_{k-1} - V^\star\|_\infty.$$

By induction: $\|V_k - V^\star\|_\infty \le \gamma^k \|V_0 - V^\star\|_\infty$.

Part 2. When $V_0 = 0$, we bound $\|V_0 - V^\star\|_\infty = \|V^\star\|_\infty$. Since rewards are bounded in $[0, R_{\max}]$:

$$V^\star(s) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r_h \,\middle|\, s_0 = s, \pi^\star\right] \le \sum_{h=0}^{\infty} \gamma^h R_{\max} = \frac{R_{\max}}{1-\gamma}.$$

Combined with Part 1, we obtain the stated bound. □

Corollary 2 (Iteration Complexity)**.** *Let* $V_0 = 0$*. To achieve* $\|V_k - V^\star\|_\infty \le \varepsilon$*, it suffices to take*

$$k \ge \frac{\log\left(\frac{R_{\max}}{\varepsilon(1-\gamma)}\right)}{\log(1/\gamma)}.$$

Using $\log(1/\gamma) \ge 1 - \gamma$*, a simpler sufficient bound is*

$$k \ge \frac{1}{1-\gamma} \log\left(\frac{R_{\max}}{\varepsilon(1-\gamma)}\right) = O\left(\frac{1}{1-\gamma} \log \frac{R_{\max}}{\varepsilon(1-\gamma)}\right).$$

iterations.

Proof. We need $\gamma^k \cdot R_{\max}/(1-\gamma) \le \varepsilon$, which gives $\gamma^k \le \varepsilon(1-\gamma)/R_{\max}$.

Taking logarithms: $k \log \gamma \le \log(\varepsilon(1-\gamma)/R_{\max})$, so $k \ge \log(R_{\max}/(\varepsilon(1-\gamma)))/\log(1/\gamma)$.

Using the inequality $\log(1/\gamma) = -\log\gamma \ge 1-\gamma$ for $\gamma \in (0,1)$, we have $1/\log(1/\gamma) \le 1/(1-\gamma)$, so it suffices to take:

$$k \ge \frac{\log(R_{\max}/(\varepsilon(1-\gamma)))}{1-\gamma} = \frac{1}{1-\gamma} \log\left(\frac{R_{\max}}{\varepsilon(1-\gamma)}\right).$$ □

Dependence on γ: The iteration complexity scales as $O(1/(1-\gamma))$, which can be large when γ is close to 1. This is the "effective horizon" of the MDP—when $\gamma = 0.99$, we need roughly 10 times more iterations than when $\gamma = 0.9$.

Stopping Criterion. In practice, we cannot compute $\|V_k - V^\star\|_\infty$ directly since $V^\star$ is unknown. The following lemma justifies using $\|V_k - V_{k-1}\|_\infty$ as a proxy.

Lemma 3 (Stopping Criterion)**.** *If* $\|V_k - V_{k-1}\|_\infty < \dfrac{\varepsilon(1-\gamma)}{\gamma}$*, then* $\|V_k - V^\star\|_\infty < \varepsilon$.

Proof. By the triangle inequality and the contraction property:

$$\begin{aligned}\|V_{k-1} - V^\star\|_\infty &\le \|V_{k-1} - V_k\|_\infty + \|V_k - V^\star\|_\infty \\ &\le \|V_{k-1} - V_k\|_\infty + \gamma\|V_{k-1} - V^\star\|_\infty.\end{aligned}$$

Rearranging: $(1-\gamma)\|V_{k-1} - V^\star\|_\infty \le \|V_{k-1} - V_k\|_\infty$, so

$$\|V_{k-1} - V^\star\|_\infty \le \frac{\|V_{k-1} - V_k\|_\infty}{1-\gamma}.$$

Therefore:

$$\|V_k - V^\star\|_\infty \le \gamma\|V_{k-1} - V^\star\|_\infty \le \frac{\gamma\|V_{k-1} - V_k\|_\infty}{1-\gamma}.$$

Setting this less than ε requires $\|V_{k-1} - V_k\|_\infty < \varepsilon(1-\gamma)/\gamma$. □

Computational Complexity

We now analyze the computational cost of Value Iteration in terms of the MDP parameters.

Lemma 4 (Per-Iteration Complexity)**.** *Each iteration of Value Iteration requires* $O(|\mathcal{S}|^2|\mathcal{A}|)$ *arithmetic operations.*

Proof. For each state $s \in \mathcal{S}$ (outer loop: $|\mathcal{S}|$ iterations):

- Compute $\max_{a\in\mathcal{A}}$ over all actions (inner loop: $|\mathcal{A}|$ iterations)
- For each action a: compute $\sum_{s'\in\mathcal{S}} \mathcal{P}(s'|s,a)V_{k-1}(s')$ (sum over $|\mathcal{S}|$ terms)

Total: $|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}| = O(|\mathcal{S}|^2|\mathcal{A}|)$. □

Theorem 5 (Total Complexity of Value Iteration)**.** *Value Iteration finds an* ε*-optimal value function in time*

$$O\left(\frac{|\mathcal{S}|^2|\mathcal{A}|}{1-\gamma}\log\frac{R_{\max}}{\varepsilon(1-\gamma)}\right).$$

Proof. Combine Corollary 2 (iteration bound) with Lemma 4 (per-iteration cost). □

Space Complexity: Value Iteration requires $O(|\mathcal{S}|)$ space to store the value function (synchronous VI typically stores both V_{k-1} and V_k, still $O(|\mathcal{S}|)$). The MDP representation requires $O(|\mathcal{S}|^2|\mathcal{A}|)$ space for the transition probabilities, unless accessed via an oracle.

Synchronous vs. Asynchronous Updates. The algorithm as stated performs *synchronous* updates: all states are updated simultaneously in each iteration. An alternative is *asynchronous* Value Iteration, where states are updated one at a time in any order. Asynchronous VI still converges to $V^\star$ as long as every state is updated infinitely often, but the convergence analysis is more subtle.

Worked Example: Navigation MDP

We illustrate Value Iteration on a simple 3-state navigation problem (see Figure 1).

Problem Setup. Consider a robot navigating a corridor with three locations: `Left` (L), `Center` (C), and `Right` (R). The goal is to reach state R.

- $\mathcal{S} = \{\texttt{L}, \texttt{C}, \texttt{R}\}$, $\mathcal{A} = \{\texttt{go-left}, \texttt{go-right}\}$
- **Transitions:** Action `go-right` succeeds with probability 0.9 and fails (stays in place) with probability 0.1. Action `go-left` is symmetric (except at the absorbing state R). At boundary states, actions toward the boundary stay in place.
- **Rewards:** $R(s, a) = 1$ if $s = \texttt{R}$, and $R(s, a) = 0$ otherwise.
- **State R is absorbing:** From R, any action keeps the robot at R.
- **Discount factor:** $\gamma = 0.9$

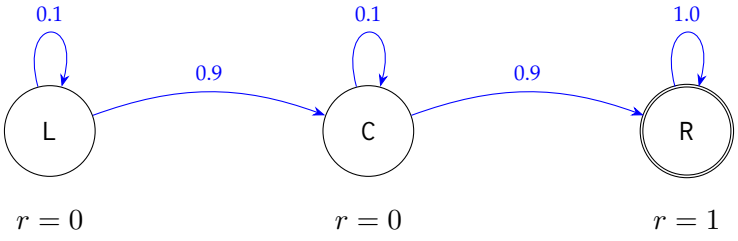

Figure 1: Transition diagram under action `go-right`. Double circle indicates the absorbing goal state.

Value Iteration Execution. Initialize $V_0(s) = 0$ for all s. We compute several iterations:

Iteration 1: (Using $V_0 = 0$ for all states)

$$V_1(\mathsf{R}) = R(\mathsf{R}, \cdot) + 0.9 \cdot V_0(\mathsf{R}) = 1 + 0.9 \cdot 0 = 1$$
$$V_1(\mathsf{C}) = \max\{0 + 0.9(0.9 \cdot V_0(\mathsf{L}) + 0.1 \cdot V_0(\mathsf{C})),\ 0 + 0.9(0.9 \cdot V_0(\mathsf{R}) + 0.1 \cdot V_0(\mathsf{C}))\} = \max\{0, 0\} = 0$$
$$V_1(\mathsf{L}) = \max\{0 + 0.9 \cdot V_0(\mathsf{L}),\ 0 + 0.9(0.9 \cdot V_0(\mathsf{C}) + 0.1 \cdot V_0(\mathsf{L}))\} = 0$$

Iteration 2: (Using $V_1 = (0, 0, 1)$)

$$\begin{aligned} V_2(\mathsf{R}) &= 1 + 0.9 \cdot V_1(\mathsf{R}) = 1 + 0.9 \cdot 1 = 1.9 \\ V_2(\mathsf{C}) &= \max\{0 + 0.9(0.9 \cdot V_1(\mathsf{L}) + 0.1 \cdot V_1(\mathsf{C})),\ 0 + 0.9(0.9 \cdot V_1(\mathsf{R}) + 0.1 \cdot V_1(\mathsf{C}))\} \\ &= \max\{0, 0.81\} = 0.81 \\ V_2(\mathsf{L}) &= \max\{0 + 0.9 \cdot V_1(\mathsf{L}),\ 0 + 0.9(0.9 \cdot V_1(\mathsf{C}) + 0.1 \cdot V_1(\mathsf{L}))\} = \max\{0, 0\} = 0 \end{aligned}$$

Continuing this process, the values converge to the optimal values. The table below shows the progression:

k	$V_k(\mathsf{L})$	$V_k(\mathsf{C})$	$V_k(\mathsf{R})$	$\|V_k - V_{k-1}\|_\infty$
0	0	0	0	—
1	0	0	1.0	1.0
2	0	0.81	1.9	0.9
3	0.656	1.612	2.71	0.81
4	1.365	2.340	3.439	0.729
⋮	⋮	⋮	⋮	⋮
∞	$V^\star(\mathsf{L}) \approx 7.92$	$V^\star(\mathsf{C}) \approx 8.90$	$V^\star(\mathsf{R}) = 10$	0

Optimal Policy. The greedy policy at convergence is $\pi^\star(s) = \texttt{go-right}$ for all s—always move toward the goal. Note that $V^\star(\mathsf{R}) = 1/(1 - 0.9) = 10$ (the absorbing state value).

Remark 1. *The optimal value at state L is less than at C because it takes longer (in expectation) to reach the goal from L. The value at R is highest because rewards are collected immediately and continuously.*

Policy Extraction and Quality Guarantees

After running Value Iteration, we extract a policy by acting greedily with respect to the computed value function.

Definition 1 (Greedy Policy). *Given a value function V, the* greedy policy π_V *is defined by:*

$$\pi_V(s) \in \operatorname*{argmax}_{a\in\mathcal{A}} \left(R(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[V(s')]\right) = \operatorname*{argmax}_{a\in\mathcal{A}} Q_V(s,a)$$

where $Q_V(s,a) \coloneqq R(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[V(s')]$ is the one-step Q-function induced by V.

If we run Value Iteration until exact convergence ($V_k = V^\star$), the greedy policy is optimal. But in practice, we stop when V_k is only ε-close to $V^\star$. The following theorem bounds the suboptimality of the resulting policy.

Theorem 6 (Policy Quality from Approximate Value (Singh and Yee, 1994)). *If $\|V_k - V^\star\|_\infty \le \varepsilon$, then the greedy policy $\pi_k = \pi_{V_k}$ satisfies*

$$\|V^{\pi_k} - V^\star\|_\infty \le \frac{2\gamma\varepsilon}{1-\gamma}.$$

Proof. Let π_k be greedy w.r.t. V_k, and define the Bellman evaluation operator for a (deterministic) policy π:

$$(\mathcal{T}^\pi V)(s) \coloneqq R(s,\pi(s)) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,\pi(s))}[V(s')].$$

Then V^π is the unique fixed point of $\mathcal{T}^\pi$ (i.e., $V^\pi = \mathcal{T}^\pi V^\pi$), and $\mathcal{T}^\pi$ is a γ-contraction in $\|\cdot\|_\infty$. Since π_k is greedy w.r.t. V_k, we have $\mathcal{T}^{\pi_k} V_k = \mathcal{T}^\star V_k$ (statewise equality). Using $\|V_k - V^\star\|_\infty \le \varepsilon$, for any s we get

$$\begin{aligned}
(\mathcal{T}^{\pi_k} V^\star)(s) &\ge (\mathcal{T}^{\pi_k} V_k)(s) - \gamma\varepsilon \\
&= (\mathcal{T}^\star V_k)(s) - \gamma\varepsilon \\
&\ge (\mathcal{T}^\star V^\star)(s) - 2\gamma\varepsilon \\
&= V^\star(s) - 2\gamma\varepsilon,
\end{aligned}$$

where the first and third inequalities use the γ-contraction property of $\mathcal{T}^{\pi_k}$ and $\mathcal{T}^\star$. Moreover, since $\mathcal{T}^{\pi_k} V^\star \le \mathcal{T}^\star V^\star = V^\star$ (statewise), we have

$$0 \le V^\star(s) - (\mathcal{T}^{\pi_k} V^\star)(s) \le 2\gamma\varepsilon \quad \text{for all } s,$$

and thus $\|V^\star - \mathcal{T}^{\pi_k} V^\star\|_\infty \le 2\gamma\varepsilon$.

Finally, since $V^{\pi_k} = \mathcal{T}^{\pi_k} V^{\pi_k}$, we have

$$\begin{aligned}
\|V^\star - V^{\pi_k}\|_\infty &= \|V^\star - \mathcal{T}^{\pi_k} V^{\pi_k}\|_\infty \\
&\le \|V^\star - \mathcal{T}^{\pi_k} V^\star\|_\infty + \|\mathcal{T}^{\pi_k} V^\star - \mathcal{T}^{\pi_k} V^{\pi_k}\|_\infty \\
&\le \|V^\star - \mathcal{T}^{\pi_k} V^\star\|_\infty + \gamma \|V^\star - V^{\pi_k}\|_\infty.
\end{aligned}$$

Rearranging gives $(1-\gamma)\|V^\star - V^{\pi_k}\|_\infty \le 2\gamma\varepsilon$, hence $\|V^{\pi_k} - V^\star\|_\infty \le 2\gamma\varepsilon/(1-\gamma)$. □

Amplification of Error: The policy quality bound has an extra factor of $\gamma/(1-\gamma)$ compared to the value approximation error. This "amplification" is unavoidable in general—a small error in V_k can cause the greedy policy to make wrong decisions that accumulate over time.

Summary: Value Iteration

Value Iteration Summary:

- **Algorithm:** $V_k = \mathcal{T}^\star V_{k-1}$ for $k \ge 1$ (apply Bellman optimality operator)
- **Convergence:** Linear rate γ^k (geometric convergence)
- **Iterations:** $O\left(\frac{1}{1-\gamma}\log\frac{R_{\max}}{\varepsilon(1-\gamma)}\right)$ for ε-accuracy
- **Per-iteration cost:** $O(|\mathcal{S}|^2|\mathcal{A}|)$
- **Total cost:** $O\left(\frac{|\mathcal{S}|^2|\mathcal{A}|}{1-\gamma}\log\frac{R_{\max}}{\varepsilon(1-\gamma)}\right)$
- **Policy quality:** Greedy policy is $\frac{2\gamma\varepsilon}{1-\gamma}$-optimal

Looking Ahead: Policy Iteration. Value Iteration converges asymptotically—for typical initializations (and $0 < \gamma < 1$), it generally does not *exactly* reach $V^\star$ in a finite number of iterations. Can we do better? In the next lecture, we present **Policy Iteration**, which works directly with policies and is guaranteed to terminate in a *finite* number of steps. The trade-off: each iteration is more expensive, but the total number of iterations can be dramatically smaller.

Remark 2 (Q-Value Iteration)**.** *An equivalent algorithm operates on Q-functions:*

$$Q_k(s,a) = R(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} Q_{k-1}(s',a')\right] = (\mathcal{T}_Q^\star Q_{k-1})(s,a).$$

This has the same convergence properties but requires $O(|\mathcal{S}||\mathcal{A}|)$ storage instead of $O(|\mathcal{S}|)$. The advantage is that the optimal policy can be read off directly: $\pi^\star(s) = \operatorname{argmax}_{a\in\mathcal{A}} Q^\star(s,a)$.

References

Satinder P Singh and Richard C Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.