CS 486/686
Value & Policy Iteration

Yuntian Deng

Lecture 14

RN 17.2 · PM 9.5.2–9.5.3

Learning goals

- Trace + implement the **value iteration** algorithm for solving an MDP.
- Trace + implement the **policy iteration** algorithm for solving an MDP.

Recap: $\pi^\star = \arg\max_a Q^\star$

From L13, the optimal policy is one greedy step from $V^\star$:

$$\pi^\star(s) = \arg\max_a Q^\star(s,a), \quad Q^\star(s,a) = R(s) + \gamma \sum_{s'} P(s' \mid s,a)\, V^\star(s').$$

Today's question

How do we actually *compute* $V^\star$?

Two algorithms: **value iteration** (iterate values) and **policy iteration** (iterate policies).

Value functions

- $V^{\pi}(s)$ — value of being in state s under policy π.
- $V^{\star}(s)$ — value of s under the **optimal** policy $\pi^{\star}$.
- $Q^{\pi}(s, a)$ — value of taking action a in s, then following π.
- $Q^{\star}(s, a)$ — same, but under $\pi^{\star}$.
- $\pi(a \mid s)$ — the policy, a distribution over actions per state.

"Value" = expected discounted future reward.

Formal definitions

Return (discounted future reward from time t):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = R_{t+1} + \gamma G_{t+1}.$$

Value functions are expected returns:

$$V^\pi(s) = \mathbb{E}_\pi\big[G_t \mid s_t = s\big], \quad Q^\pi(s,a) = \mathbb{E}_\pi\big[G_t \mid s_t = s, a_t = a\big].$$

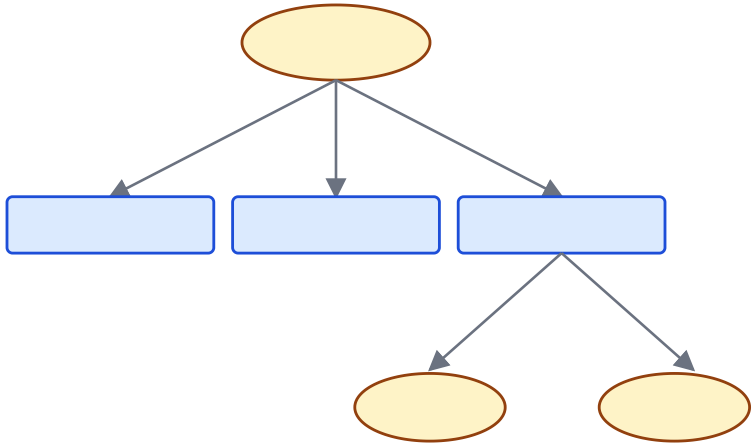
The two are related (each is a sum of the other):

$$V^\pi(s) = \sum_a \pi(a \mid s)\, Q^\pi(s,a)$$

$$Q^\pi(s,a) = R(s) + \gamma \sum_{s'} P(s' \mid s,a)\, V^\pi(s')$$

The backup diagram

A picture of the V/Q recursion: **policy** branches on actions; **environment** branches on next states.



The Bellman optimality equation

For the optimal policy, choose the *best* action at every state:

$$V^\star(s) = \max_a Q^\star(s,a), \quad Q^\star(s,a) = R(s) + \gamma \sum_{s'} P(s' \mid s,a)\, V^\star(s').$$

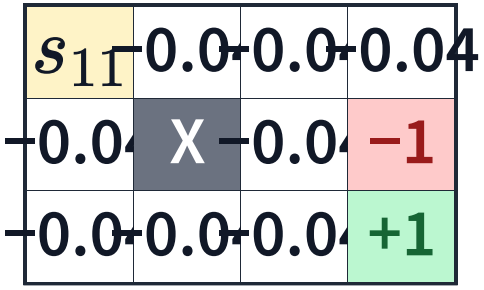
Substitute one into the other:

Bellman equation

$$V^\star(s) = R(s) + \gamma \max_a \sum_{s'} P(s' \mid s,a)\, V^\star(s').$$

$V^\star$ is the **unique** solution to this system of $|S|$ equations.

Apply Bellman to $V^\star(s_{11})$



From s_{11}, each action gives a different mixture (0.8 intended + 0.1 each side):

$$
\begin{aligned}
V^\star(s_{11}) = -0.04 + \gamma\ \max\{ & \\
& 0.8\,V^\star(s_{12}) + 0.1\,V^\star(s_{21}) + 0.1\,V^\star(s_{11}), \ \texttt{// right} \\
& 0.9\,V^\star(s_{11}) + 0.1\,V^\star(s_{12}), \ \texttt{// up} \\
& 0.9\,V^\star(s_{11}) + 0.1\,V^\star(s_{21}), \ \texttt{// left} \\
& 0.8\,V^\star(s_{21}) + 0.1\,V^\star(s_{12}) + 0.1\,V^\star(s_{11}) \ \texttt{// down} \\
\}
\end{aligned}
$$

Q1. Can we solve this system of Bellman equations in polynomial time?

> **No.** The system is *nonlinear* because of the $\max$ — there is no general efficient solver. We'll use an **iterative** algorithm instead.

The value iteration algorithm

Treat the Bellman equation as an *update rule*. Let $V_i(s)$ be the i^{th} estimate.

value-iteration(MDP, ε)

1. Initialize $V_0(s)$ arbitrarily (e.g. 0).
2. For $i = 0, 1, 2, \ldots$: for each state s, apply the Bellman update

$$V_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} P(s' \mid s, a)\, V_i(s').$$

3. Stop when $\max_s |V_{i+1}(s) - V_i(s)| < \varepsilon$.

Guarantee: $V_i \to V^\star$ as $i \to \infty$.

- **Synchronous**: keep V_i frozen while computing all of V_{i+1}.
- **Asynchronous**: update entries in place, in any order. Often converges faster.

Apply VI to the grid

Setup: $\gamma = 1, R(s) = -0.04$ for non-goal states. Terminals: $V(s_{24}) = -1, V(s_{34}) = +1$. Init $V_0(s) = 0$ for all non-terminals.

$V_0(s)$

0	0	0	0
0	X	0	−1
0	0	0	+1

After one Bellman update, only the cells adjacent to a goal will see something different from -0.04.

One iteration: $V_1(s)$

Apply $V_1(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} P(s' \mid s, a) V_0(s')$ at every cell. Two cells worth a closer look:

$V_1(s)$

0.0	0.0	0.0	0.04
0.0	X	0.0	−1
0.0	0.0	0.76	+1

Q2. $V_1(s_{23})$ **(yellow):** all neighbors are 0; best action is Left (avoids -1). $\Rightarrow -0.04 + 0 = \boxed{-0.04}$.

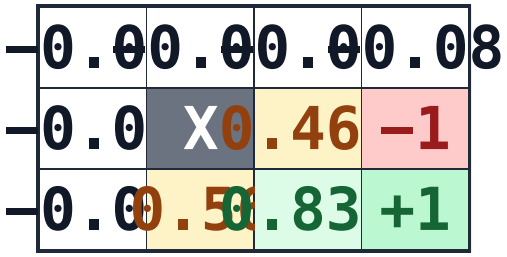
Q3. $V_1(s_{33})$ **(green):** Right has $0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0 = 0.8$. $\Rightarrow -0.04 + 0.8 = \boxed{0.76}$.

The +1 reward has "leaked" one cell up; the −1 hasn't propagated because we avoid it.

Two iterations: $V_2(s)$

Apply Bellman again, using V_1 as input. Three cells worth a closer look:

$V_2(s)$



Q4. $V_2(s_{33})$: Right gives $0.8(+1) + 0.1(-0.04) + 0.1(0.76) = 0.872. \Rightarrow$

$$-0.04 + 0.872 = \boxed{0.832}.$$

Q5. $V_2(s_{23})$: Down gives $0.8(0.76) + 0.1(-0.04) + 0.1(-1) = 0.504. \Rightarrow$

$$-0.04 + 0.504 = \boxed{0.464}.$$

Q6. $V_2(s_{32})$: Right gives $0.8(0.76) + 0.1(-0.04) + 0.1(-0.04) = 0.6. \Rightarrow$

$$-0.04 + 0.6 = \boxed{0.56}.$$

Values propagate one cell per iteration. After enough sweeps they converge to the L13 $V^\star$.

Policy iteration

Insight: if one action is clearly better, you don't need precise utility values — just enough to rank actions.

policy-iteration(MDP)

1. Start from an arbitrary policy π_0.
2. Repeat:
   - **Policy evaluation:** compute $V^{\pi_i}(s)$, the utility of every state under π_i.
   - **Policy improvement:** $\pi_{i+1}(s) = \arg\max_a \sum_{s'} P(s' \mid s, a)\, V^{\pi_i}(s')$.
3. Until $\pi_{i+1} = \pi_i$.

Terminates in finitely many steps: there are finitely many policies and each iteration strictly improves.

Policy evaluation vs Bellman: the max disappears

In PE the action is *fixed* by π, so the system becomes **linear** in V:

Policy evaluation (linear)

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s))\, V^{\pi}(s')$$

No $\max$ — the action is fixed. $|S|$ linear equations in $|S|$ unknowns.

linear: easy to solve

Bellman optimality (nonlinear)

$$V^{\star}(s) = R(s) + \gamma \max_{a} \sum_{s'} P(s' \mid s, a)\, V^{\star}(s')$$

The $\max$ makes it nonlinear — need an iterative algorithm (VI).

nonlinear: harder

Policy improvement reintroduces the $\arg\max$: $\pi_{i+1}(s) = \arg\max_{a} \sum_{s'} P(s' \mid s, a)\, V^{\pi_i}(s')$.

How do we perform policy evaluation?

Two ways to solve the linear system $V^\pi(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s))\, V^\pi(s')$:

Exactly

Linear algebra: a single solve of an $|S| \times |S|$ system.

cost: $O(|S|^3)$

Iteratively (cheap)

Run m simplified Bellman updates (no $\max$):

$$V_{j+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s))\, V_j(s')$$

cost: $O(m \cdot |S| \cdot |A_{\text{avg}}|)$

In practice, a small m is usually enough — PI converges in far fewer outer iterations than VI.

Learning goals (recap)

✓ Trace and implement **value iteration** for solving an MDP.

✓ Trace and implement **policy iteration** for solving an MDP.

Next: reinforcement learning

Today we knew the dynamics P and rewards R of the MDP. What if we *don't*?

L15: learn the policy by **interacting with the environment.**