Applied Computational Mathematics Lab

TEDDY LAZEBNIK'S COURSE · LECTURE 4

Reinforcement Learning

Lecture 4 · Theory of dynamic programming: norms, contractions and fixed points; Bellman operators; why DP converges; approximate DP and the value of a greedy policy



Teddy Lazebnik · Applied Computational Mathematics Laboratory

BEFORE WE START

Recap, and what this lecture does

- Last lecture: MDPs, dynamic programming, value iteration and policy iteration.
- We claimed these algorithms converge. Today we prove it.

TODAY

Deepen the formalism

Bellman equations become **operators**; DP becomes “apply an operator until nothing moves”.

TODAY

What breaks

Approximate DP: what happens when values can’t be computed or stored exactly.



IN PLAIN WORDS

One idea carries the whole lecture: if a machine pulls any two inputs closer together, repeating it lands on one single answer.

WHERE WE ARE GOING

Next lectures use sampled and approximate versions of these algorithms, mostly without knowing the environment. Today’s theory tells us what to expect from them.

BEFORE WE START

Today's map

PART 1

Tools from analysis

Norms, contractions, fixed points, and Banach's theorem.

PART 2

Bellman operators

The optimality and expectation operators, and why they are contractions.

PART 3

Dynamic programming, revisited

Value iteration and policy evaluation converge, with a rate.

PART 4

Approximate dynamic programming

When the model or the values are approximate: divergence, and the value of a greedy policy.



IN PLAIN WORDS

By the end you should be able to explain why value iteration converges, how fast, and how much a small error in the values can cost a greedy policy.

Applied Computational Mathematics Lab

PART 1

Tools from analysis

Distances between value functions, contractions, and fixed points.



Normed vector spaces

NORMED VECTOR SPACE

A vector space $\mathcal{X}$ with a norm $\|\cdot\| : \mathcal{X} \to \mathbb{R}$ such that

- $\|x\| \geq 0$, and $\|x\| = 0$ only if $x = 0$;
- $\|\alpha x\| = |\alpha| \, \|x\|$ (homogeneity);
- $\|x_1 + x_2\| \leq \|x_1\| + \|x_2\|$ (triangle inequality).

- Today: $\mathcal{X} = \mathbb{R}^d$, one number per state, with two norms:
  - the max norm $\|x\|_\infty = \max_i |x_i|$;
  - the weighted L2 norm $\|x\|_{2,\rho} = \sqrt{\sum_i \rho_i x_i^2}$.

TRY IT

$x = (3, -4, 1)$.
Max norm: $\|x\|_\infty = 4$.
L2 norm: $\sqrt{9 + 16 + 1} \approx 5.10$.
Weighted, $\rho = (0.5, 0.25, 0.25)$:
$\sqrt{4.5 + 4 + 0.25} \approx 2.96$.



IN PLAIN WORDS

A norm is a ruler for whole vectors. Here each vector is a value function: one number per state.

The max norm: the worst state

- To compare two value functions u and v, go state by state and keep the **biggest** gap:

$$\|u - v\|_\infty = \max_s |u(s) - v(s)|$$

- Not the average gap: the worst one.
- A disaster in one state can hide behind a good average; the max norm doesn't let it.

STATE	u	v	$\lvert u - v \rvert$
A	5	5	0
B	2	3	1
C	7	7	0
D	−1	9	**10**

Two guesses that agree almost everywhere. The max-norm distance is 10; the average gap is only 2.75.



IN PLAIN WORDS

An agent is only as right as its most wrong state.

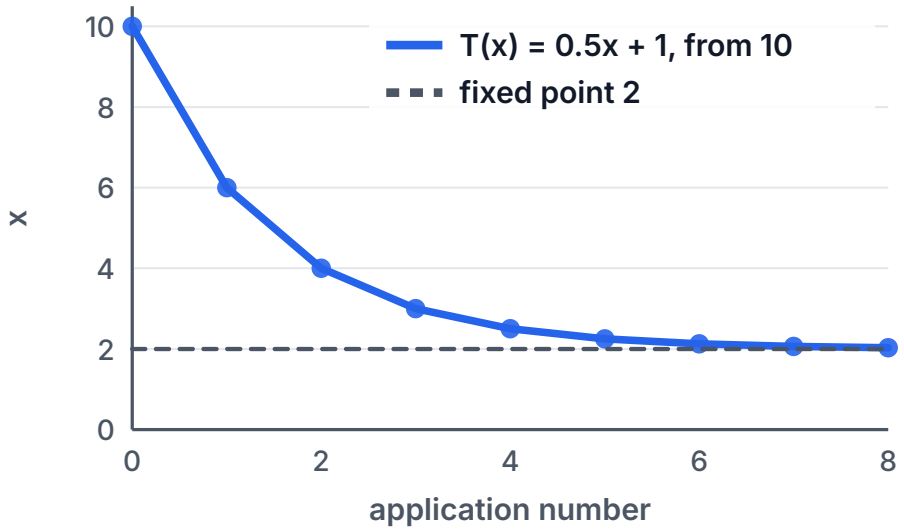
Contraction mappings

CONTRACTION MAPPING

Let $\mathcal{X}$ be a vector space with a norm. A mapping $T : \mathcal{X} \to \mathcal{X}$ is an α-contraction if there is $\alpha \in [0, 1)$ such that for all x_1, x_2

$$\|Tx_1 - Tx_2\| \leq \alpha \|x_1 - x_2\|$$

- If $\alpha \in [0, 1]$, T is called **non-expanding**.
- Every contraction is Lipschitz, hence continuous: if $x_n \to x$ then $Tx_n \to Tx$.



TRY IT

$T(x) = 0.5x + 1$ on the real line: $|Tx_1 - Tx_2| = 0.5|x_1 - x_2|$, a 0.5-contraction. From 10: 10, 6, 4, 3, 2.5, … heading to 2.

Fixed points

FIXED POINT

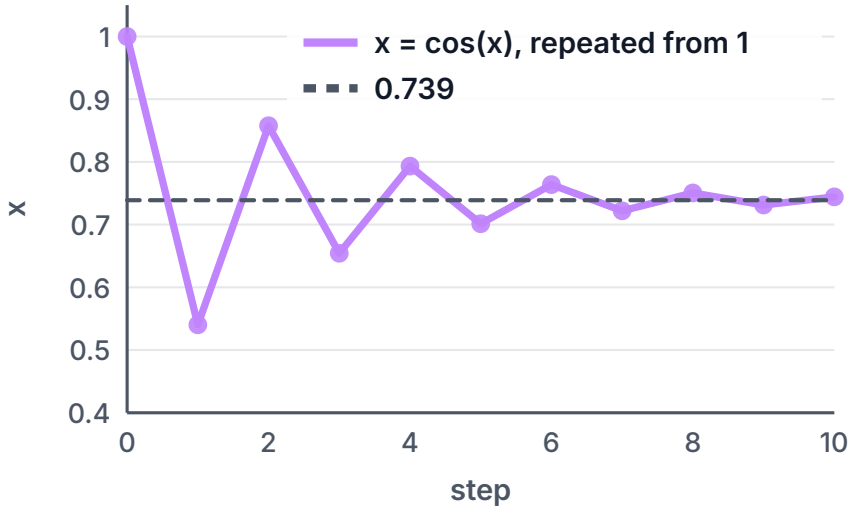
A point $x \in \mathcal{X}$ is a fixed point of an operator T if $Tx = x$.

MAP	CONTRACTION?	FIXED POINTS
$x \mapsto 0.5x + 1$	yes, $\alpha = 0.5$	exactly one: 2
$x \mapsto x$	no (non-expanding)	every point
$x \mapsto x + 1$	no	none
$x \mapsto 2x$	no (expanding)	only 0, and iterates run away



IN PLAIN WORDS

A thermostat set to 21 degrees: it pushes the room toward 21, and once there, it stays. 21 is its fixed point.



Press “cos” on a calculator again and again: you land on 0.739, the only solution of $\cos x = x$ (cosine is a contraction on $[-1, 1]$).

Banach's fixed-point theorem

THEOREM (BANACH, 1922)

Let $\mathcal{X}$ be a complete normed vector space and T a γ-contraction. Then

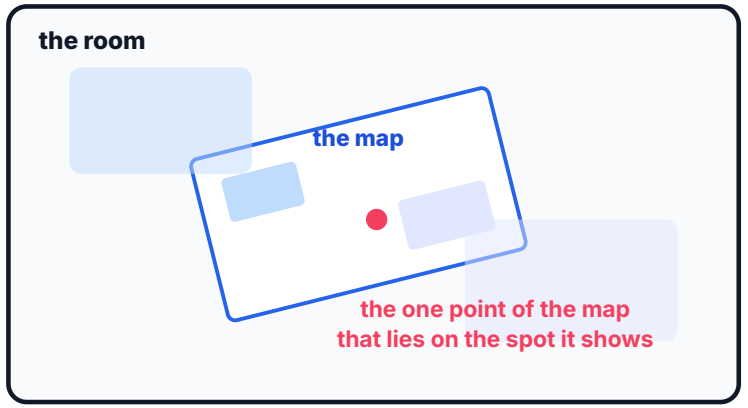
- T has a **unique** fixed point: there is exactly one x^* with $Tx^* = x^*$;
- for **any** start x_0, the sequence $x_{n+1} = Tx_n$ converges to x^* geometrically:

$$\|x_n - x^*\| \leq \gamma^n \|x_0 - x^*\| \longrightarrow 0$$

- Exactly one answer, reached from anywhere, and fast: every step removes the same fraction of the remaining error.

TRY IT

$T(x) = 0.5x + 1$ from $x_0 = 10$: the error to $x^* = 2$ is 8, 4, 2, 1, … exactly $0.5^n \times 8$. From $x_0 = -6$ the errors are the same, and the end point is the same.



Drop a map of a room on the floor of that room. Exactly one point of the map lies on the spot it shows. The map is a contraction of the room.



IN PLAIN WORDS

Banach proved this in his doctoral thesis.

Check yourself: Part 1

CHECK YOURSELF

1. For $u = (1, 4, 2)$ and $v = (3, 1, 2)$, compute $\|u - v\|_\infty$.

CHECK YOURSELF

2. Is $T(x) = 0.9x - 3$ a contraction on the real line? What is its fixed point?

CHECK YOURSELF

3. A 0.5-contraction starts with error 8. What is the largest possible error after 3 steps?

CHECK YOURSELF

4. Why does $x \mapsto x + 1$ have no fixed point, even though it is non-expanding?





HINT

Answers are at the end of the deck.

Applied Computational Mathematics Lab

PART 2

Bellman operators

Turning the Bellman equations into machines that act on whole value functions.



Recap: MDPs and Bellman equations

- An MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$ formally describes an environment.
- Values: $v_\pi(s) = \mathbb{E}[\sum_{t \geq 0} \gamma^t R_{t+1} \mid S_0 = s, \pi]$, and $q_\pi(s, a)$ likewise with $A_0 = a$; optimal values $v_* = \max_\pi v_\pi$, $q_* = \max_\pi q_\pi$.

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

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



IN PLAIN WORDS

Last lecture these were equations to **solve**. Today we read the right-hand sides as machines that turn one value function into another.

RUNNING EXAMPLE

Throughout, we reuse the recycling robot of Lecture 3: $\alpha = 0.8$, $\beta = 0.6$, $r_{\text{search}} = 2$, $r_{\text{wait}} = 1$, rescue -3, $\gamma = 0.9$.

The Bellman optimality operator

BELLMAN OPTIMALITY OPERATOR

Let $\mathcal{V}$ be the space of bounded real-valued functions over $\mathcal{S}$. The Bellman optimality operator $T^* : \mathcal{V} \to \mathcal{V}$ is defined state by state:

$$(T^* f)(s) = \max_a \left[r(s,a) + \gamma \sum_{s'} p(s' \mid s,a)\, f(s') \right] \quad \forall f \in \mathcal{V}$$

- It takes a **whole** value function in and gives a whole value function out.
- One sweep of value iteration is exactly one application of T^*.

TRY IT

The robot. Feed in $f = (0, 0)$ (high, low):
high: $\max(2, 1) = 2$; low: $\max(0.6 \cdot 2 + 0.4 \cdot (-3),\ 1,\ 0) = 1$.
So $T^* f = (2, 1)$.



IN PLAIN WORDS

Picture every possible value function as a point in a huge space. T^* moves points around in that space.

Three properties of T^*

1

Unique fixed point

$T^*v_* = v_*$, and nothing else is fixed.

2

γ-contraction

$\|T^*v - T^*u\|_\infty \le \gamma \|v - u\|_\infty$ for all u, v.

3

Monotonic

If $u \le v$ in every state, then $T^*u \le T^*v$.



IN PLAIN WORDS

Start below v_* and you stay below it, creeping up every sweep (property 3). Wherever you start, you get there (properties 1 and 2).

TRY IT

Robot: $u = (0, 0)$, $v = (10, 10)$, so $\|v - u\|_\infty = 10$.
$T^*u = (2, 1)$ and $T^*v = (11, 10)$.
The gap is now 9, which is $\le 0.9 \times 10$. And $u \le v$ gave $T^*u \le T^*v$.

Proof: T* is a γ-contraction

$$
\begin{aligned}
|T^*v(s) - T^*u(s)| &= \left| \max_a \left[r(s,a) + \gamma \mathbb{E}_{s'|s,a} v(s') \right] - \max_b \left[r(s,b) + \gamma \mathbb{E}_{s''|s,b} u(s'') \right] \right| \\
&\leq \max_a \left| \gamma \mathbb{E}_{s'|s,a} \left[v(s') - u(s') \right] \right| \\
&\leq \gamma \max_{s'} |v(s') - u(s')| = \gamma \|v - u\|_\infty
\end{aligned}
$$

- Taking the max over s on the left gives $\|T^*v - T^*u\|_\infty \leq \gamma \|v - u\|_\infty$.
- Line 2 uses $|\max_a f(a) - \max_b g(b)| \leq \max_a |f(a) - g(a)|$; the rewards cancel.



IN PLAIN WORDS

Three facts, no calculus: **maxes** don't stretch gaps, **averages** don't stretch gaps, and multiplying by **γ** shrinks them.

TRY IT

The max fact, with numbers: $f = (3, 1)$ and $g = (2, 4)$. Then $|\max f - \max g| = |3 - 4| = 1$, while $\max_a |f(a) - g(a)| = \max(1, 3) = 3$.

Proof: T* is monotonic

- Suppose $v(s) \le u(s)$ for all s. Then for every s, a: $r(s,a) + \gamma \mathbb{E}_{s'|s,a} v(s') \le r(s,a) + \gamma \mathbb{E}_{s'|s,a} u(s')$.

$$
\begin{aligned}
T^* v(s) - T^* u(s) &= \max_a \left[r(s,a) + \gamma \mathbb{E} v(s') \right] - \max_b \left[r(s,b) + \gamma \mathbb{E} u(s'') \right] \\
&\le \max_a \left(\left[r(s,a) + \gamma \mathbb{E} v(s') \right] - \left[r(s,a) + \gamma \mathbb{E} u(s') \right] \right) \le 0
\end{aligned}
$$

- So $T^* v(s) \le T^* u(s)$ for every s.
- Line 2: the max of a difference bounds the difference of the maxes, taking $b = a$.



IN PLAIN WORDS

Bigger input, bigger (or equal) output. If one guess is everywhere more optimistic than another, it stays so after an update.

WHY IT IS USEFUL

Start value iteration at $v_0 = 0$ on a problem with non-negative rewards: $v_1 = T^* v_0 \ge v_0$, and by monotonicity the values only grow, sweep after sweep, up to v_*.

Value iteration through the operator

VALUE ITERATION

1. Start with any v_0.
2. Update: $v_{k+1} = T^* v_k$.
3. As $k \to \infty$, $v_k \to v_*$ in the max norm.

$$\begin{aligned} \|v_k - v_*\|_\infty &= \|T^* v_{k-1} - T^* v_*\|_\infty \quad \text{(fixed point)} \\ &\le \gamma \|v_{k-1} - v_*\|_\infty \quad \text{(contraction)} \\ &\le \gamma^k \|v_0 - v_*\|_\infty \quad \text{(repeat)} \end{aligned}$$

- A direct application of Banach's theorem.



Our run: a random MDP we made up (10 states, 2 actions, $\gamma = 0.9$). The error falls from 4.6 to 0.06 in 40 sweeps, always under the bound.

How fast is fast?

- Each sweep cuts the worst error by at least a factor γ.
- To cut the error 10 times you need k sweeps with $\gamma^k \le 0.1$, that is $k \ge \log 0.1 / \log \gamma$.

γ	SWEEPS PER 10× SMALLER ERROR	SWEEPS FOR 1,000× SMALLER
0.5	4	10
0.9	22	66
0.99	230	688
0.999	2,302	6,905



IN PLAIN WORDS

Looking far ahead (γ close to 1) costs computation: the horizon and the number of sweeps grow together.

TRY IT

With $\gamma = 0.9$ and an initial error of 100, the bound promises an error below 1 after $\log(0.01)/\log(0.9) \approx 44$ sweeps.

The Bellman expectation operator

BELLMAN EXPECTATION OPERATOR

For any policy π, the Bellman expectation operator $T^\pi : \mathcal{V} \to \mathcal{V}$ is

$$(T^\pi f)(s) = \sum_a \pi(a \mid s) \Big[r(s,a) + \gamma \sum_{s'} p(s' \mid s,a)\, f(s') \Big] \quad \forall f \in \mathcal{V}$$

1

Unique fixed point

$T^\pi v_\pi = v_\pi$.

2

γ-contraction

in the max norm.

3

Monotonic

$u \le v \Rightarrow T^\pi u \le T^\pi v$

.

TRY IT

The robot, policy “always wait”: $T^\pi(0,0) = (1,1)$, then $(1.9, 1.9)$, then $(2.71, 2.71)$, ... heading to the fixed point $v_\pi = (10, 10)$.



IN PLAIN WORDS

Same as T^*, but instead of the max over actions we average over what π picks.

Proof: T^π is a contraction

$$T^\pi v(s) - T^\pi u(s) = \sum_a \pi(a \mid s)\Big[r(s,a) + \gamma \mathbb{E}_{s'|s,a} v(s') - r(s,a) - \gamma \mathbb{E}_{s'|s,a} u(s')\Big]$$
$$= \gamma \sum_a \pi(a \mid s)\, \mathbb{E}_{s'|s,a}\big[v(s') - u(s')\big]$$

- So $|T^\pi v(s) - T^\pi u(s)| \le \gamma \max_{s'} |v(s') - u(s')|$, and hence $\|T^\pi v - T^\pi u\|_\infty \le \gamma \|v - u\|_\infty$.
- The second line also shows monotonicity: if $v \ge u$ everywhere, the right side is ≥ 0.



IN PLAIN WORDS

Even simpler than for T^*: the rewards cancel, and a weighted average of gaps is never bigger than the biggest gap.

TRY IT

Robot, "always wait": $u = (0, 0)$, $v = (10, 10)$. $T^\pi u = (1, 1)$, $T^\pi v = (10, 10)$: the gap shrinks from 10 to 9.

LINK TO LECTURE 3

The operator is affine: $T^\pi v = \mathbf{r}^\pi + \gamma P^\pi v$. Its fixed point solves $(I - \gamma P^\pi)v = \mathbf{r}^\pi$: the matrix form from Lecture 3.

Policy evaluation converges

ITERATIVE POLICY EVALUATION

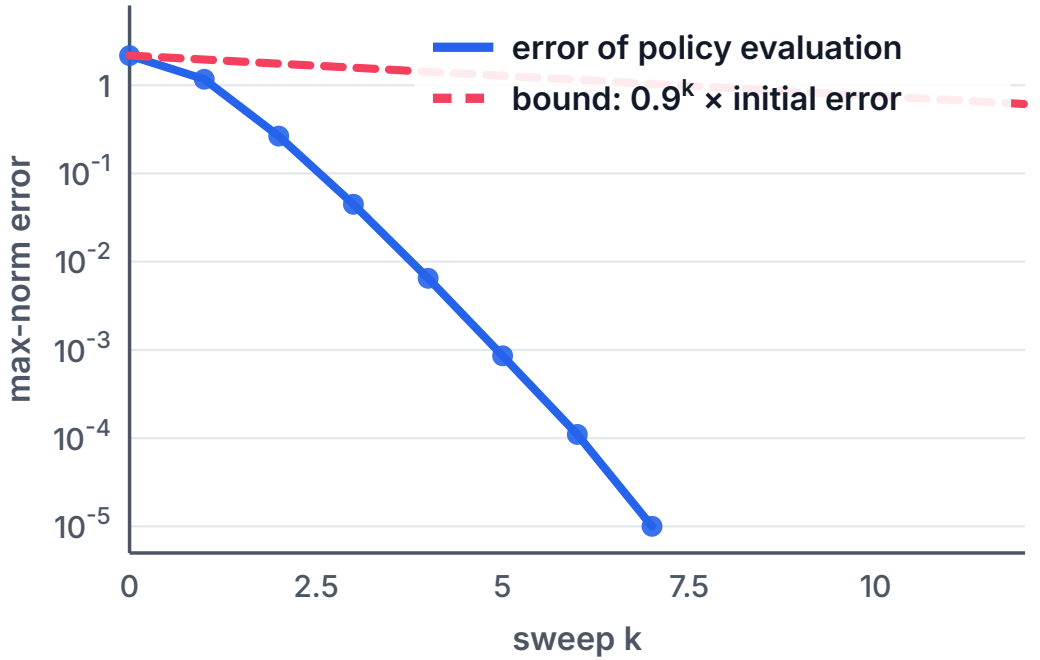
1. Start with any v_0.
2. Update: $v_{k+1} = T^\pi v_k$.
3. As $k \to \infty$, $v_k \to v_\pi$: Banach again.

- So every algorithm of Lecture 3 is the same idea: **apply a contraction until it stops moving**.



IN PLAIN WORDS

This is why the course can make promises: here is the proof behind them.



Our run on the 4-state chain of Lecture 3 ("always left", $\gamma = 0.9$). The bound is a guarantee, not a prediction: here the error is about 10^{-5} after 7 sweeps, and below the printed precision after that.

Dynamic programming with operators

VALUE ITERATION

Apply T* again and again

Start with v_0; $v_{k+1} = T^* v_k$. Stop when the change is small.

POLICY ITERATION

Evaluate, then improve

Start with π_0. Evaluate v_{π_i}, for instance by iterating $v_k = T^{\pi_i} v_{k-1}$. Improve greedily: $\pi_{i+1}(s) = \operatorname{argmax}_a q_{\pi_i}(s, a)$.



IN PLAIN WORDS

All of DP fits in one line: contract, and when you have a policy to improve, improve it greedily.

TRADE-OFF

Policy iteration takes few iterations but each is expensive (a full evaluation). Value iteration takes many cheap sweeps. Both are guaranteed to reach the optimum in a finite MDP.

TRY IT

The robot (high, low), our numbers. Policy iteration from "always wait": (wait, wait), then (search, wait), then (search, recharge), optimal after two improvements. Value iteration from 0 acts optimally after 3 sweeps, but needs 71 sweeps to get within 0.01 of $v_* = (16.95, 15.25)$.

The same operators for q

BELLMAN OPERATORS ON ACTION VALUES

On $\mathcal{Q}$, the bounded functions over $\mathcal{S} \times \mathcal{A}$:

$$(T_Q^\pi f)(s,a) = r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) \sum_{a'} \pi(a' \mid s')\, f(s',a')$$

$$(T_Q^* f)(s,a) = r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) \max_{a'} f(s',a')$$

- Their unique fixed points are q_π and q_*. Same properties: γ-contractions and monotonic.
- Bonus: with q_* in hand we can act without the model, $\pi_*(s) = \operatorname{argmax}_a q_*(s,a)$.



IN PLAIN WORDS

Notice where the max sits: inside, at the next state. That is exactly where Q-learning (Lecture 6) puts it.

TRY IT

Robot, $f = 0$: $T_Q^* f(\text{low}, \text{search}) = 0.6 \cdot 2 + 0.4 \cdot (-3) = 0$, $T_Q^* f(\text{low}, \text{wait}) = 1$, $T_Q^* f(\text{low}, \text{recharge}) = 0$.

Check yourself: Part 2

CHECK YOURSELF

5. Write one sweep of value iteration using an operator.

CHECK YOURSELF

6. With $\gamma = 0.8$, two value functions differ by at most 5. What is the largest possible difference after one application of T^*?

CHECK YOURSELF

7. Which property of T^* guarantees that value iteration started at different points ends at the same values?

CHECK YOURSELF

8. How many sweeps does $\gamma = 0.5$ need to cut the error 1,000 times?



Applied Computational Mathematics Lab

PART 3

Approximate dynamic programming

When we can’t compute or store the values exactly.



Two luxuries we usually don’t have

- So far we assumed perfect knowledge of the MDP, and an exact table of values.
- Realistically, more often than not:

SAMPLING ERROR

We don’t know the MDP

We estimate the Bellman update from samples, so we never apply the true T^π or T^*.

APPROXIMATION ERROR

We can’t store every value

We fit values within a class (linear functions, neural networks), so each update is squeezed into that class.

- Objective: under these conditions, still find a policy that is optimal, or close to it.



IN PLAIN WORDS

Two errors from two directions. Sometimes they cancel; usually they don’t.

EXAMPLE

A chess program can’t keep a table with one value per board, and it doesn’t know its opponent’s replies. Both errors at once.

WHY IT MATTERS

This is where deep RL lives: big networks, and samples instead of the true model.

Approximate value iteration

APPROXIMATE VALUE ITERATION

1. Start with v_0.
2. Update $v_{k+1} = \mathcal{A}T^* v_k$, where $\mathcal{A}$ squeezes the result into our function class, so $v_{k+1} \approx T^* v_k$.

QUESTION

As $k \to \infty$, does $v_k \to v_*$?

- **In general, no.** The fitting step $\mathcal{A}$ need not be a contraction in the max norm, so Banach's promise is gone.

HOW THE FIT WORKS

With a parametric approximator $v_\theta(s)$, $\theta \in \mathbb{R}^m$:

- compute the targets $T^* v_k(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s]$;
- fit θ_{k+1} so $v_{\theta_{k+1}} \approx T^* v_k$, for example by least squares over states:

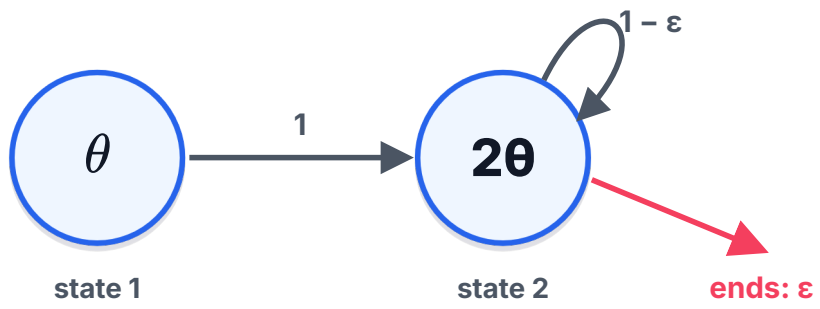
$$\theta_{k+1} = \operatorname*{argmin}_\theta \sum_s \left(v_\theta(s) - T^* v_k(s)\right)^2$$



IN PLAIN WORDS

Each fit on its own is fine. It is the repetition that can hurt.

A two-state example that diverges



all rewards are 0, so the true values are 0

After Tsitsiklis & Van Roy (1997). Our drawing.

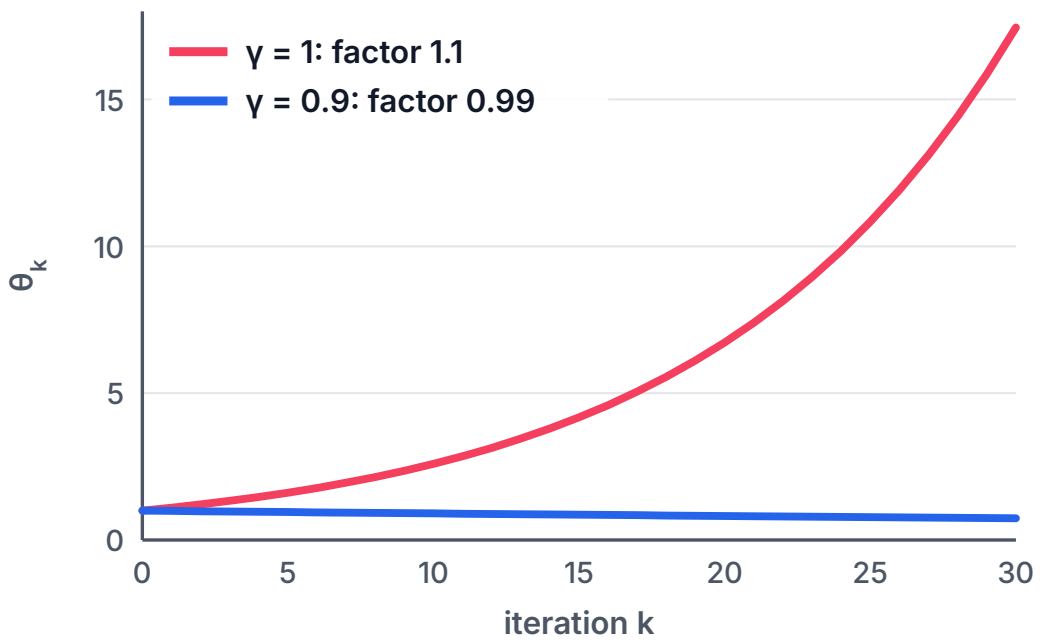
TRY IT

$\varepsilon = 1/8$, $\gamma = 1$: the factor is $(6 - 0.5)/5 = 1.1$. From $\theta_0 = 1$: 1.1, 1.21, 1.33, 1.46, ... Each fit is the best possible for its targets, yet the estimate grows without bound.

- One parameter θ: the first state's value is θ, the second's is 2θ.
- No decisions, all rewards 0, so the true values are 0.
- State 1 always moves to state 2; state 2 stays, except that with probability ε the episode ends.
- Fit θ by least squares to the targets $\gamma \mathbb{E}[v_{\theta_k}(S_{t+1})]$, weighting both states equally:

$$\theta_{k+1} = \operatorname*{argmin}_{\theta} \left[(\theta - 2\gamma\theta_k)^2 + (2\theta - 2\gamma(1-\varepsilon)\theta_k)^2 \right] = \frac{(6 - 4\varepsilon)\gamma}{5}\,\theta_k$$

Watching it diverge



Our run, $\varepsilon = 1/8$, $\theta_0 = 1$. With $\gamma = 1$: $\theta_{10} \approx 2.6$, $\theta_{30} \approx 17$. The truth is 0.

- Each step multiplies θ by $(6 - 4\varepsilon)\gamma/5$. With $\varepsilon = 1/8$ and $\gamma = 1$ that is 1.1: the estimate runs away.
- It diverges whenever $\gamma > 5/(6 - 4\varepsilon) \approx 0.909$.
- Why? To fix state 2, the fit pushes θ up, and drags state 1 along with it.



IN PLAIN WORDS

The fix in this example: weight each state by how often it is actually visited (the on-policy distribution). Then it converges.

Hopeless? Not quite

1

Sampled versions converge

Online, sample-based versions of these algorithms converge under mild conditions.

2

Rare in practice

Even with function approximation, divergence rarely shows up in practice.

3

Order beats accuracy

Many value functions induce the same optimal policy.



IN PLAIN WORDS

We don’t need the values to be right; we need them to be in the right **order**.

REMEMBER

In Lecture 3, the values of the **random** policy on the 4×4 grid were far from optimal, yet acting greedily on them gave an optimal policy.

TRY IT

Two actions with true values 5 and 3. Estimates 9 and 1 are far off, but the greedy choice is still correct. Estimates 4.1 and 4.2 are close, but greedy picks the wrong action.

How good is a greedy policy?

THEOREM (VALUE OF A GREEDY POLICY)

Let $q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be any function, and π the greedy policy with respect to q. Then

$$\|q_* - q_\pi\|_\infty \leq \frac{2\gamma}{1-\gamma} \|q_* - q\|_\infty$$

- Left: how much worse the greedy policy really is, in the worst state.
- Right: how wrong our value estimate is, times an exchange rate.



IN PLAIN WORDS

Small errors in values cost only a small loss in performance. Value error goes in, performance loss comes out, at a fixed exchange rate.

TRY IT

With $\gamma = 0.9$ and estimates off by at most 0.5 everywhere, the greedy policy loses at most $18 \times 0.5 = 9$ in any state.

The proof in five lines

$$
\begin{aligned}
\|q_* - q_\pi\|_\infty &= \|q_* - T^\pi q + T^\pi q - q_\pi\|_\infty \\
&\le \|q_* - T^\pi q\|_\infty + \|T^\pi q - q_\pi\|_\infty \\
&= \|T^* q_* - T^* q\|_\infty + \|T^\pi q - T^\pi q_\pi\|_\infty \\
&\le \gamma \|q_* - q\|_\infty + \gamma \|q - q_\pi\|_\infty \\
&\le 2\gamma \|q_* - q\|_\infty + \gamma \|q_* - q_\pi\|_\infty
\end{aligned}
$$

- Rearranging the last line: $(1-\gamma)\|q_* - q_\pi\|_\infty \le 2\gamma \|q_* - q\|_\infty$.

LINE	USES
1	add and subtract $T^\pi q$
2	triangle inequality
3	fixed points; π greedy on q, so $T^\pi q = T^* q$
4	both operators are γ-contractions
5	triangle inequality again: $\|q - q_\pi\| \le \|q - q_*\| + \|q_* - q_\pi\|$



IN PLAIN WORDS

Every step uses only what we proved today.

Reading the bound

γ	$2\gamma/(1-\gamma)$	MEANING
0	0	no future: every policy has $q_\pi = r = q_*$
0.5	2	short horizon, mild amplification
0.9	18	errors compound over about 10 steps
0.99	198	long horizon: small errors can cost a lot

- The longer you look ahead, the more a small value error can cost: errors compound along the way.
- If $q = q_*$, the bound is 0: the greedy policy is optimal, as we knew.



IN PLAIN WORDS

With long horizons, invest in accurate values. They are worth it.

IN PRACTICE

The bound is a worst case. In practice the loss is often much smaller, because only errors that change the greedy choice matter.

Approximate policy iteration

APPROXIMATE POLICY ITERATION

1. Start with π_0.
2. Evaluate approximately: $q_i = \mathcal{A} q_{\pi_i}$, so $q_i \approx q_{\pi_i}$.
3. Improve greedily: $\pi_{i+1}(s) = \operatorname{argmax}_a q_i(s, a)$.

QUESTION	ANSWER
Does $q_i \to q_*$?	In general, no.
Does π_i converge to an optimal policy?	In general, no.
Hopeless?	In some cases not: it depends on the approximation $\mathcal{A}$. More in later lectures.



IN PLAIN WORDS

With errors in every evaluation, the policy can oscillate around the optimum instead of settling on it.

WHAT CAN BE SAID

With evaluation errors at most ϵ, the policies eventually stay within about $2\gamma\epsilon/(1-\gamma)^2$ of optimal (Bertsekas & Tsitsiklis, 1996). A bound, not a convergence.

Summary: approximate DP

VALUE ITERATION

Approximate

Start with v_0; update $v_{k+1} = \mathcal{A}T^* v_k$, so $v_{k+1} \approx T^* v_k$.

POLICY ITERATION

Approximate

Evaluate $q_i \approx q_{\pi_i}$; improve $\pi_{i+1} = \operatorname{argmax}_a q_i(s, a)$.



IN PLAIN WORDS

Exact DP converges by Banach. Approximate DP can drift or diverge, but acting greedily degrades gracefully.

CHECK YOURSELF

9. Why is Banach's theorem no longer enough once we add the fitting step $\mathcal{A}$?

CHECK YOURSELF

10. Using the bound, what is the worst loss with $\gamma = 0.5$ and value errors at most 2?

CHECK YOURSELF

11. In the two-state example with $\varepsilon = 0$, above which γ does θ diverge?

WRAP-UP

Summary

- **Max norm**: the distance between value functions is the worst state.
- **Contractions** pull points together; **Banach**: a unique fixed point, reached from anywhere, geometrically fast.
- **Bellman operators** T^* and T^π (and their q versions) are γ-contractions and monotonic; their fixed points are v_* and v_π.
- So value iteration and policy evaluation converge, with error at most γ^k times the initial error.
- **Approximate DP** can diverge, but a greedy policy loses at most $2\gamma/(1-\gamma)$ times the value error.

$$\|Tx - Ty\| \leq \gamma \|x - y\|$$

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

$$\|q_* - q_\pi\|_\infty \leq \frac{2\gamma}{1-\gamma} \|q_* - q\|_\infty$$

Answers to check yourself

- **1.** Gaps 2, 3, 0, so $\|u - v\|_\infty = 3$.
- **2.** Yes, a 0.9-contraction. Fixed point: $x = 0.9x - 3 \Rightarrow x = -30$.
- **3.** $8 \times 0.5^3 = 1$.
- **4.** $x + 1 = x$ has no solution; non-expanding ($\alpha = 1$) is not enough for Banach.
- **5.** $v_{k+1} = T^* v_k$.
- **6.** $0.8 \times 5 = 4$.
- **7.** T^* is a contraction, so it has exactly one fixed point, v_*, and every start converges to it.
- **8.** $0.5^k \leq 0.001$ needs $k = 10$.
- **9.** The combined step $\mathcal{A}T^*$ need not be a contraction in the max norm.
- **10.** $\frac{2 \cdot 0.5}{0.5} \times 2 = 4$.
- **11.** Factor $6\gamma/5 > 1$, so $\gamma > 5/6 \approx 0.83$.

CREDITS

Sources and credits

- The order of topics follows *Lecture 4: Theoretical Fundamentals of Dynamic Programming* of the DeepMind × UCL Reinforcement Learning Lecture Series (2021), by Diana Borsa. This deck is not an official DeepMind or UCL material.
- The text, figures, examples and simulations were written, drawn and run for ACML. Numbers labelled “our run” come from our own code; examples taken from books or papers are credited where they appear.
- Banach (1922); Tsitsiklis & Van Roy (1997), the two-state counterexample (re-run by us); Bertsekas & Tsitsiklis (1996), *Neuro-Dynamic Programming*.
- Sutton & Barto (2018), chapters 4 and 11.
- Companion videos: **ACML Explains**, season 1, episodes 15–16. More from the lab: acml.teddylazebnik.com



Applied Computational Mathematics Lab

END OF LECTURE 4

Next: learning from experience

Lecture 5 replaces the model with experience: Monte Carlo and temporal-difference learning for prediction.

