Chapter 3: Exercises & Labs

Contents

1 Chapter 3 — Exercises & Labs

We use these labs to keep the operator-theoretic proofs in sync with runnable Bellman code. Each snippet is self-contained so we can execute it directly (e.g., `.venv/bin/python - <<'PY' ... PY`) while cross-referencing Sections 3.7–3.9.

1.1 Lab 3.1 — Contraction Ratio Tracker

Objective: $\log \|\mathcal{T}V_1 - \mathcal{T}V_2\|_\infty / \|V_1 - V_2\|_\infty$ and compare it to γ.

```
import numpy as np

gamma = 0.9
P = np.array(
    [
        [[0.7, 0.3, 0.0], [0.4, 0.6, 0.0]],
        [[0.0, 0.6, 0.4], [0.0, 0.3, 0.7]],
        [[0.2, 0.0, 0.8], [0.1, 0.0, 0.9]],
    ]
)
R = np.array(
    [
        [1.0, 0.5],
        [0.8, 1.2],
        [0.0, 0.4],
    ]
)

def bellman_operator(V, P, R, gamma):
```

```
    Q = R + gamma * np.einsum("ijk,k->ij", P, V)
    return Q.max(axis=1)

rng = np.random.default_rng(0)
V1 = rng.normal(size=3)
V2 = rng.normal(size=3)
ratio = np.linalg.norm(
    bellman_operator(V1, P, R, gamma) - bellman_operator(V2, P, R, gamma),
    ord=np.inf,
) / np.linalg.norm(V1 - V2, ord=np.inf)
print(f"Contraction ratio: {ratio:.3f} (theory bound = {gamma:.3f})")
```

Output:

```
Contraction ratio: 0.872 (theory bound = 0.900)
```

Tasks 1. Explain the slack between the bound and the observation (hint: the max operator is 1-Lipschitz, so the true ratio is often strictly less than γ). 2. Log the ratio across multiple seeds and include the extrema in Chapter 3 to make EQ-3.16 concrete.

1.2 Lab 3.2 — Value Iteration Wall-Clock Profiling

Objective: verify the $O\left(\frac{1}{1-\gamma}\right)$ convergence rate numerically by reusing the same toy kernel.

```
import numpy as np

P = np.array(
    [
        [[0.7, 0.3, 0.0], [0.4, 0.6, 0.0]],
        [[0.0, 0.6, 0.4], [0.0, 0.3, 0.7]],
        [[0.2, 0.0, 0.8], [0.1, 0.0, 0.9]],
    ]
)
R = np.array(
    [
        [1.0, 0.5],
        [0.8, 1.2],
        [0.0, 0.4],
    ]
)

def value_iteration(P, R, gamma, tol=1e-6, max_iters=500):
    V = np.zeros(P.shape[0])
    for k in range(max_iters):
        Q = R + gamma * np.einsum("ijk,k->ij", P, V)
        V_new = Q.max(axis=1)
        if np.linalg.norm(V_new - V, ord=np.inf) < tol:
            return V_new, k + 1
        V = V_new
    raise RuntimeError("Value iteration did not converge")

stats = {}
for gamma in [0.5, 0.7, 0.9]:
    _, iters = value_iteration(P, R, gamma=gamma)
    stats[gamma] = iters
print(stats)
```

Output:

```
{0.5: 21, 0.7: 39, 0.9: 128}
```

Tasks 1. Plot the iteration counts against $\frac{1}{1-\gamma}$ and reference the figure when explaining COR-3.7.3 (value iteration convergence rate). 2. Re-run the sweep after perturbing `R` with zero-mean noise to visualize the reward-perturbation sensitivity bound PROP-3.7.4.

2 Chapter 3 — Lab Solutions

Vlad Prytula

These solutions demonstrate the operator-theoretic foundations of reinforcement learning. Every solution weaves rigorous theory ([DEF-3.6.3], THM-3.6.2-Banach, [THM-3.7.1]) with runnable implementations, following the principle that proofs illuminate practice and code verifies theory.

All numeric outputs shown are from running the code with fixed seeds. Some blocks are abbreviated for readability (ellipses indicate omitted lines).

2.1 Lab 3.1 — Contraction Ratio Tracker

Goal: Log $\|\mathcal{T}V_1 - \mathcal{T}V_2\|_\infty / \|V_1 - V_2\|_\infty$ and compare it to γ.

2.1.1 Theoretical Foundation

Recall from THM-3.7.1 that the Bellman operator for an MDP with discount factor $\gamma < 1$ is a γ-**contraction** in the $\|\cdot\|_\infty$ norm (see [EQ-3.16]):

$$\|\mathcal{T}V_1 - \mathcal{T}V_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty \quad \forall V_1, V_2.$$

This property is fundamental: it guarantees that value iteration converges to a unique fixed point V^* exponentially fast ([THM-3.6.2-Banach], Banach Fixed-Point Theorem).

This lab empirically verifies the contraction inequality by sampling random value function pairs and measuring the actual ratio.

2.1.2 Solution

```python
from scripts.ch03.lab_solutions import lab_3_1_contraction_ratio_tracker

results = lab_3_1_contraction_ratio_tracker(n_seeds=20, verbose=True)
```

Actual Output:

```
======================================================================
Lab 3.1: Contraction Ratio Tracker
======================================================================

MDP Configuration:
  States: 3, Actions: 2
  Discount factor gamma = 0.9

Theoretical bound [THM-3.7.1]: ||T V1 - T V2||_inf <= 0.9 * ||V1 - V2||_inf

Computing contraction ratios across 20 random V pairs...
```

```
Seed       Ratio        <= gamma?
------------------------------
8925       0.7763       OK
77395      0.7240       OK
65457      0.7509       OK
43887      0.6131       OK
43301      0.7543       OK
85859      0.8856       OK
8594       0.4745       OK
69736      0.7208       OK
20146      0.4828       OK
9417       0.5208       OK
... (10 more seeds)

================================================
CONTRACTION RATIO STATISTICS
================================================
  Theoretical bound (gamma): 0.900
  Empirical mean:        0.624
  Empirical std:         0.183
  Empirical min:         0.202
  Empirical max:         0.886
  Slack (gamma - max):   0.014
  All ratios <= gamma?   YES
...
```

2.1.3 Task 1: Explaining the Slack

Why is the observed ratio strictly less than γ?

The proof of THM-3.7.1 uses several inequalities that are not always tight:

1. **The max operator is 1-Lipschitz**: The key step uses

$$|\sup_a f(a) - \sup_a g(a)| \leq \sup_a |f(a) - g(a)|$$

   In the finite-action case, sup is max. Equality holds only when the maximizers coincide for both functions. When V_1 and V_2 induce different optimal actions at some state, the actual difference is smaller.

2. **Transition probability averaging**: The expected future value is

$$\sum_{s'} P(s'|s,a)[V_1(s') - V_2(s')]$$

   Unless $V_1 - V_2$ has constant sign across all states, this sum is strictly less than $\|V_1 - V_2\|_\infty$.

3. **Structure in the MDP**: Real MDPs have structure. Not all (V_1, V_2) pairs achieve the worst case. The bound γ is tight only for adversarially constructed examples.

Practical implication: Value iteration can converge faster than the worst-case γ^k bound; the contraction argument provides a guarantee, not a runtime prediction.

2.1.4 Task 2: Multiple Seeds and Extrema

Running across 100 seeds:

```
results = lab_3_1_contraction_ratio_tracker(n_seeds=100, verbose=False)
print(f"Min ratio: {results['min']:.4f}")
print(f"Max ratio: {results['max']:.4f}")
print(f"Slack (gamma - max): {results['slack']:.4f}")
```

Output:

```
Min ratio: 0.2015
Max ratio: 0.8937
Slack (gamma - max): 0.0063
```

The maximum observed ratio approaches but never exceeds $\gamma = 0.9$, confirming THM-3.7.1.

2.1.5 Key Insight

> **The contraction inequality is a *bound*, not an equality.** In practice, convergence is often faster than theory predicts. However, the bound provides *guaranteed* worst-case behavior—essential for algorithm analysis and safety-critical applications.

2.2 Lab 3.2 — Value Iteration Wall-Clock Profiling

Goal: Verify the $O\left(\frac{1}{1-\gamma}\right)$ convergence rate numerically.

2.2.1 Theoretical Foundation

From COR-3.7.3 (see the rate bound [EQ-3.18]), value iteration satisfies:

$$\|V_k - V^*\|_\infty \leq \frac{\gamma^k}{1-\gamma}\|\mathcal{T}V_0 - V_0\|_\infty.$$

To achieve tolerance ε, it suffices to choose k so that the right-hand side is at most ε. Writing $C := \|\mathcal{T}V_0 - V_0\|_\infty/(1-\gamma)$, this is the condition $\gamma^k C \leq \varepsilon$, hence:

$$k > \frac{\log(C/\varepsilon)}{\log(1/\gamma)} \approx \frac{\log(C/\varepsilon)}{1-\gamma},$$

where we used $\log(1/\gamma) \approx 1-\gamma$ for γ close to 1. For fixed tolerance (and problem-dependent C), **iteration complexity scales as** $O(1/(1-\gamma))$.

2.2.2 Solution

```
from scripts.ch03.lab_solutions import lab_3_2_value_iteration_profiling

results = lab_3_2_value_iteration_profiling(
    gamma_values=[0.5, 0.7, 0.9, 0.95, 0.99],
    tol=1e-6,
    verbose=True
)
```

Actual Output:

```
======================================================================
Lab 3.2: Value Iteration Wall-Clock Profiling
======================================================================
```

```
MDP Configuration:
  States: 3, Actions: 2
  Convergence tolerance: 1e-06

Running value iteration for gamma in [0.5, 0.7, 0.9, 0.95, 0.99]...

gamma    Iters    1/(1-gamma)    Ratio
---------------------------------------
0.50     21       2.0        10.50
0.70     39       3.3        11.70
0.90     128      10.0       12.80
0.95     261      20.0       13.05
0.99     1327     100.0      13.27

================================================
ANALYSIS: Iteration Count vs 1/(1-gamma)
================================================

Linear fit: Iters ~ 13.32 * 1/(1-gamma) + -5.5

Theory predicts: Iters ~ C * 1/(1-gamma) * log(1/eps)
  With eps = 1e-06, log(1/eps) ~ 13.8
  Expected slope ~ log(1/eps) ~ 13.8
  Observed slope: 13.32
...
```

2.2.3 Analysis: The $O(1/(1-\gamma))$ Relationship

The iteration count scales approximately as $1/(1-\gamma)$. Let's understand why:

γ	Effective Horizon $\frac{1}{1-\gamma}$	Iterations	Ratio
0.5	2	21	10.5
0.7	3.3	39	11.7
0.9	10	128	12.8
0.95	20	261	13.1
0.99	100	1327	13.3

Key observations:

1. **Linear scaling confirmed**: Iterations grow approximately linearly with $1/(1-\gamma)$.
2. **The constant factor**: The ratio (Iters / Horizon) is roughly constant and close to log(1/ *varepsilon*) for fixed tolerance ε (up to problem-dependent constants hidden in the initial error term). This is consistent with the contraction bound derived from THM-3.6.2-Banach.
3. **Computational cost diverges**: As $\gamma \to 1$ (infinite horizon):
   - $\gamma = 0.99$: ~1300 iterations (this run)
   - $\gamma = 0.999$: ~13000 iterations (rough extrapolation)
   - $\gamma = 0.9999$: ~130000 iterations (rough extrapolation)

Practical guidance: Use the smallest γ that captures the relevant planning horizon. Unnecessarily large γ wastes computation.

2.2.4 Task 2: Perturbation Analysis

We perturb the reward matrix $R \to R + \Delta R$ and measure the effect on V^*:

```
from scripts.ch03.lab_solutions import extended_perturbation_sensitivity

perturb_results = extended_perturbation_sensitivity(
    noise_scales=[0.01, 0.05, 0.1, 0.2, 0.5],
    gamma=0.9,
    verbose=True
)
```

Actual Output:

```
======================================================================
Extended Lab: Perturbation Sensitivity Analysis
======================================================================

Original MDP (gamma = 0.9):
  V* = [7.45069962 6.50651745 5.63453744]

Theoretical bound [PROP-3.7.4]: ||V*_perturbed - V*||_inf <= ||DeltaR||_inf / (1-gamma)
  With gamma = 0.9, bound = ||DeltaR||_inf / 0.10 = 10.0 * ||DeltaR||_inf

||DeltaR||_inf Bound        Mean ||DeltaV*||   Max ||DeltaV*||    Bound OK?
----------------------------------------------------------------------
0.01       0.100        0.0400          0.0840          OK
0.05       0.500        0.2000          0.3623          OK
0.10       1.000        0.4028          0.6865          OK
0.20       2.000        0.7811          1.7013          OK
0.50       5.000        1.5165          3.2851          OK

================================================
All perturbation bounds satisfied: YES
================================================
...
```

Interpretation: The sensitivity bound $\|V^*_{perturbed} - V^*\|_\infty \leq \|\Delta R\|_\infty/(1-\gamma)$ tells us:

1. **Reward errors amplify**: Small errors in reward estimation cause larger errors in value function, amplified by $1/(1-\gamma)$.
2. γ **controls sensitivity**: Higher γ means more sensitivity:
   - $\gamma = 0.9$: 10x amplification
   - $\gamma = 0.99$: 100x amplification
3. **Practical implication**: In long-horizon problems, reward modeling errors matter more; this motivates careful reward design and validation.

2.3 Extended Lab: Banach Fixed-Point Theorem Verification

Goal: Empirically verify THM-3.6.2-Banach: 1. **Existence**: A unique fixed point V^* exists 2. **Convergence**: From any V_0, value iteration converges to V^* 3. **Rate**: $\|V_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty$

2.3.1 Solution

```
from scripts.ch03.lab_solutions import extended_banach_convergence_verification

banach_results = extended_banach_convergence_verification(
    n_initializations=10,
    gamma=0.9,
    verbose=True
)
```

Actual Output:

```
======================================================================
Extended Lab: Banach Fixed-Point Theorem Verification
======================================================================

Reference V* (from V0 = 0):
  V* = [7.45070814 6.50652596 5.63454596]
  Converged in 215 iterations

Testing convergence from 10 random initializations...

Init #   ||V0||_inf   Iters    ||V_final - V*||_inf
-------------------------------------------------
1        80.73        235      1.72e-09
2        13.44        220      3.15e-11
3        12.03        203      1.77e-09
4        72.98        203      1.16e-11
5        53.61        190      3.57e-11
6        92.98        220      1.71e-09
7        41.55        226      1.72e-09
8        34.85        206      1.74e-09
9        37.55        227      1.76e-09
10       11.65        206      5.25e-12

================================================
BANACH FIXED-POINT THEOREM VERIFICATION
================================================

[THM-3.6.2-Banach] Verification Results:
  (1) Existence:   V* exists OK
  (2) Uniqueness:  All 10 initializations -> same V*: OK
  (3) Convergence: All trials converged OK
  (4) Rate bound:  gamma^k bound violated 0 times across all trials OK
...
```

2.3.2 Why This Matters

The Banach Fixed-Point Theorem provides **ironclad guarantees**:

1. **Global convergence**: No matter where we start, we converge to V^*. This is why value iteration is robust—no clever initialization is required.
2. **Unique optimum**: There is exactly one optimal value function. No local optima to worry about, no sensitivity to initialization (for finding the optimal value).
3. **Exponential convergence**: Error shrinks by factor γ each iteration, in contrast to the $O(1/k)$ rates

typical of first-order optimization methods.

Contrast with general optimization: In neural network training, local optima, saddle points, and initialization sensitivity are major concerns. The contraction property of the Bellman operator eliminates all these issues. This is why dynamic programming works so reliably when it is applicable.

2.4 Extended Lab: Discount Factor Analysis

Goal: Understand how γ affects all aspects of value iteration.

2.4.1 Solution

```
from scripts.ch03.lab_solutions import extended_discount_factor_analysis

gamma_results = extended_discount_factor_analysis(
    gamma_values=[0.0, 0.3, 0.5, 0.7, 0.9, 0.95, 0.99],
    verbose=True
)
```

Actual Output:

```
======================================================================
Extended Lab: Discount Factor Analysis
======================================================================

MDP: 3 states, 2 actions
Convergence tolerance: 1e-08

gamma  Horizon    Iters    ||V*||_inf    Avg Ratio
--------------------------------------------------
0.00   1.0        2        1.200      0.000
0.30   1.4        16       1.459      0.299
0.50   2.0        27       1.950      0.500
0.70   3.3        52       3.008      0.700
0.90   10.0       172      7.451      0.900
0.95   20.0       351      13.696     0.950
0.99   100.0      1786     62.830     0.990
```

2.4.2 Key Insights

1. Horizon interpretation: $1/(1-\gamma)$ is the "effective planning horizon": - $\gamma = 0.9$: Look ~10 steps ahead - $\gamma = 0.99$: Look ~100 steps ahead

2. The bandit case ($\gamma = 0$): A single Bellman backup reaches the fixed point, because

$$V(s) = \max_a R(s, a)$$

requires no bootstrapping from future values. (With an update-based stopping rule $\|V_{k+1} - V_k\|_\infty < \varepsilon$, one extra iteration is used to *confirm* the fixed point numerically.)

3. Value magnitude grows: As γ increases, V^* accumulates more total discounted reward. In this toy MDP, $\|V^*\|_\infty$ at $\gamma = 0.99$ is about 32x larger than at $\gamma = 0.5$.

4. Contraction ratio approaches γ: The empirical contraction ratio closely tracks the theoretical bound as $\gamma \to 1$.

5. Practical guidance: - Start with smaller γ for faster iteration during development - Increase γ only if the task requires longer planning - Consider γ as a hyperparameter, not a fundamental constant

2.5 Summary: Theory-Practice Insights

These labs validated the operator-theoretic foundations of Chapter 3:

Lab	Key Discovery	Chapter Reference
Lab 3.1	Contraction ratio <= gamma always (with slack)	THM-3.7.1, EQ-3.16
Lab 3.2	Iterations scale as $O(1/(1-\gamma))$	THM-3.6.2-Banach
Extended: Perturbation	Value errors <= reward errors / (1-gamma)	PROP-3.7.4
Extended: Discount	gamma controls horizon, sensitivity, complexity	Section 3.4
Extended: Banach	Global convergence from any initialization	THM-3.6.2-Banach

Key Lessons:

1. **Contractions guarantee convergence**: The γ-contraction property THM-3.7.1 is why value iteration works. It provides existence, uniqueness, AND exponential convergence—all in one theorem.
2. **The bound is worst-case**: Actual convergence is often faster than γ^k. The theoretical bound is for analysis and safety guarantees, not runtime prediction.
3. **γ is a complexity dial**: Higher γ means longer horizons, larger values, more iterations, and more sensitivity to errors. Choose wisely.
4. **Global convergence is remarkable**: Unlike many non-convex optimization problems where initialization can matter, value iteration converges to the same V^* from any starting point. This robustness comes from the contraction property.
5. **Perturbation sensitivity scales with horizon**: The $1/(1-\gamma)$ amplification factor appears everywhere—in iteration count, value magnitude, and error sensitivity. Long-horizon RL is fundamentally harder.

Connection to Practice:

These theoretical properties explain why: - **Value iteration is reliable** (contraction guarantees convergence) - **Deep RL is harder** (function approximation breaks contraction) - **Reward design matters** (errors amplify by $1/(1-\gamma)$) - **Discount tuning is important** (it controls the entire complexity profile)

Chapter 4 begins building the simulator where we will apply these foundations.

2.6 Running the Code

All solutions are in `scripts/ch03/lab_solutions.py`:

```
# Run all labs
.venv/bin/python scripts/ch03/lab_solutions.py --all

# Run specific lab
.venv/bin/python scripts/ch03/lab_solutions.py --lab 3.1
.venv/bin/python scripts/ch03/lab_solutions.py --lab 3.2

# Run extended labs
```

```
.venv/bin/python scripts/ch03/lab_solutions.py --extended perturbation
.venv/bin/python scripts/ch03/lab_solutions.py --extended discount
.venv/bin/python scripts/ch03/lab_solutions.py --extended banach

# Interactive menu
.venv/bin/python scripts/ch03/lab_solutions.py
```

2.7 Appendix: Mathematical Proofs

Note: The following proofs are included for standalone reference and reproduce arguments from the main chapter. For the canonical presentation with full context and remarks, see Sections 3.6–3.7.

2.7.1 Proof of THM-3.7.1: Bellman Operator is a γ-Contraction

Theorem. For an MDP with discount factor $\gamma < 1$, the Bellman operator

$$(\mathcal{T}V)(s) = \max_a \left\{ R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s') \right\}$$

is a γ-contraction in the $\|\cdot\|_\infty$ norm.

Proof. Let V_1, V_2 be arbitrary value functions. For any state s:

$$\begin{aligned}
|(\mathcal{T}V_1)(s) - (\mathcal{T}V_2)(s)| &= \left| \max_a Q_1(s,a) - \max_a Q_2(s,a) \right| && (1)\\
&\leq \max_a |Q_1(s,a) - Q_2(s,a)| \quad \text{(1-Lipschitz of max)} && (2)\\
&= \max_a \left| \gamma \sum_{s'} P(s'|s,a)[V_1(s') - V_2(s')] \right| && (3)\\
&\leq \gamma \max_a \sum_{s'} P(s'|s,a)|V_1(s') - V_2(s')| && (4)\\
&\leq \gamma \|V_1 - V_2\|_\infty \max_a \underbrace{\sum_{s'} P(s'|s,a)}_{=1} && (5)\\
&= \gamma \|V_1 - V_2\|_\infty && (6)
\end{aligned}$$

Taking supremum over all s: $\|\mathcal{T}V_1 - \mathcal{T}V_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$. □

2.7.2 Connection to Chapter 1: The Bandit Case

When $\gamma = 0$, the Bellman equation reduces to:

$$V^*(s) = \max_a R(s,a)$$

This is exactly the Chapter 1 bandit optimality condition ([EQ-1.9] and [EQ-1.10]): the optimal value equals the statewise supremum of the immediate Q-values. The Bellman operator with $\gamma = 0$ becomes $(\mathcal{T}V)(s) = \sup_a R(s,a)$, which does not depend on V; hence value iteration reaches the fixed point in one step.

End of Lab Solutions