

Online and Reinforcement Learning (2025)

Home Assignment 2

Davide Marchi 777881

Contents

1	Short Questions	2
2	MDPs with Similar Parameters Have Similar Values	3
3	Policy Evaluation in RiverSwim	5
4	Solving a Discounted Grid-World	5
5	Off-Policy Evaluation in Episode-Based River-Swim	7

1 Short Questions

Determine whether each statement below is True or False and provide a very brief justification.

1. **Statement:** “In a finite discounted MDP, every possible policy induces a Markov Reward Process.”

Answer: False. This statement assumes that the policy depends only on the current state. If we allow policies to depend on the *entire* past history (*history-dependent* policies), then the resulting transitions in the state space may no longer satisfy the Markov property, since the chosen action at each step might be a function of all previous states and actions. Hence not *every* (fully history-dependent) policy necessarily induces a Markov Reward Process in the *original* state space.

2. **Statement:** “Consider a finite discounted MDP, and assume that π is an optimal policy. Then, the action(s) output by π does not depend on history other than the current state (i.e., π is necessarily stationary).”

Answer: False. While it is true that there *exists* an optimal policy which is stationary deterministic, it does not follow that *all* optimal policies must be so. In fact, multiple distinct policies (some stationary, others possibly history-dependent or randomized) can achieve exactly the same optimal value. Hence it is incorrect to say that *any* optimal policy π must be purely state-dependent (stationary).

3. **Statement:** “In a finite discounted MDP, a greedy policy with respect to optimal action-value function, Q^* , corresponds to an optimal policy.”

Answer: True. From the Bellman optimality equations for Q^* , a policy that selects

$$\arg \max_a Q^*(s, a)$$

at each state s is indeed an optimal policy. This policy attains the same value as Q^* itself, thus achieving the optimal value.

4. **Statement:** “Under the coverage assumption, the Weighted Importance Sampling Estimator \hat{V}_{wIS} converges to V^π with probability 1.”

Answer: True. The coverage assumption ensures that the target policy’s state-action probabilities are absolutely continuous w.r.t. the behavior policy. Under this assumption, Weighted Importance Sampling (though slightly biased) is a *consistent* estimator of V^π , meaning it converges almost surely to V^π as the sample size grows unbounded.

2 MDPs with Similar Parameters Have Similar Values

Setup: We have two discounted MDPs

$$M_1 = (S, A, P_1, R_1, \gamma) \quad \text{and} \quad M_2 = (S, A, P_2, R_2, \gamma),$$

sharing the same discount factor $\gamma \in (0, 1)$, the same finite state-action space, and rewards bounded in $[0, R_{\max}]$. For all state-action pairs (s, a) :

$$|R_1(s, a) - R_2(s, a)| \leq \alpha, \quad \|P_1(\cdot | s, a) - P_2(\cdot | s, a)\|_1 \leq \beta.$$

Consider a fixed stationary policy π , and let V_1^π and V_2^π be its value functions in M_1 and M_2 , respectively. The goal is to show that

$$|V_1^\pi(s) - V_2^\pi(s)| \leq \frac{\alpha + \gamma R_{\max} \beta}{(1 - \gamma)^2} \quad \text{for every state } s \in S.$$

Step 1: Write down the Bellman equations for each MDP.

By definition of π , the Bellman fixed-point form is:

$$V_1^\pi = r_1^\pi + \gamma P_1^\pi V_1^\pi, \quad V_2^\pi = r_2^\pi + \gamma P_2^\pi V_2^\pi,$$

where

$$r_m^\pi(s) = R_m(s, \pi(s)), \quad (P_m^\pi f)(s) = \sum_{s'} P_m(s' | s, \pi(s)) f(s'), \quad m = 1, 2.$$

Define $\delta = V_1^\pi - V_2^\pi$. Then

$$\delta = (r_1^\pi - r_2^\pi) + \gamma (P_1^\pi V_1^\pi - P_2^\pi V_2^\pi).$$

To facilitate the separation of terms, we introduce and subtract $\gamma P_1^\pi V_2^\pi$, which allows us to rewrite the second term as:

$$P_1^\pi V_1^\pi - P_2^\pi V_2^\pi = (P_1^\pi V_1^\pi - P_1^\pi V_2^\pi) + (P_1^\pi V_2^\pi - P_2^\pi V_2^\pi).$$

Substituting this back, we obtain:

$$\delta = (r_1^\pi - r_2^\pi) + \gamma P_1^\pi (V_1^\pi - V_2^\pi) + \gamma (P_1^\pi - P_2^\pi) V_2^\pi.$$

Step 3: Take norms and use triangle/inequality bounds.

Taking the supremum norm ($\|\cdot\|_\infty$) on both sides we obtain

$$\|\delta\|_\infty = \|(r_1^\pi - r_2^\pi) + \gamma P_1^\pi \delta + \gamma (P_1^\pi - P_2^\pi) V_2^\pi\|_\infty.$$

By the *triangle inequality*, the norm of a sum is at most the sum of the norms, so we can split the right-hand side as:

$$\|\delta\|_\infty \leq \|r_1^\pi - r_2^\pi\|_\infty + \gamma \|P_1^\pi \delta\|_\infty + \gamma \|(P_1^\pi - P_2^\pi) V_2^\pi\|_\infty.$$

Now we can proceed with:

- *Reward difference:*

Since $|R_1(s, a) - R_2(s, a)| \leq \alpha$, it follows that $\|r_1^\pi - r_2^\pi\|_\infty \leq \alpha$.

- *Term with $P_1^\pi \delta$:*

We have

$$\|P_1^\pi \delta\|_\infty \leq \|\delta\|_\infty,$$

since P_1^π is a probability kernel and thus a contraction in sup norm.

- *Term with $(P_1^\pi - P_2^\pi) V_2^\pi$:*

For each s ,

$$|(P_1^\pi - P_2^\pi) V_2^\pi(s)| \leq \sum_{s'} |P_1(s' | s, \pi(s)) - P_2(s' | s, \pi(s))| |V_2^\pi(s')|.$$

By assumption, $\|P_1(\cdot | s, a) - P_2(\cdot | s, a)\|_1 \leq \beta$, and $\|V_2^\pi\|_\infty \leq \frac{R_{\max}}{1-\gamma}$. Hence,

$$\|(P_1^\pi - P_2^\pi) V_2^\pi\|_\infty \leq \beta \frac{R_{\max}}{1-\gamma}.$$

Putting these bounds together,

$$\|\delta\|_\infty \leq \alpha + \gamma \|\delta\|_\infty + \gamma \beta \frac{R_{\max}}{1-\gamma}.$$

Step 4: Solve for $\|\delta\|_\infty$.

We isolate $\|\delta\|_\infty$ on one side:

$$(1-\gamma) \|\delta\|_\infty \leq \alpha + \gamma \beta \frac{R_{\max}}{1-\gamma}.$$

Thus

$$\|\delta\|_\infty \leq \frac{\alpha}{1-\gamma} + \frac{\gamma \beta R_{\max}}{(1-\gamma)^2}.$$

Since $\alpha/(1-\gamma) \leq \alpha/(1-\gamma)^2$ whenever $0 < \gamma < 1$, we can write

$$\|\delta\|_\infty \leq \frac{\alpha + \gamma \beta R_{\max}}{(1-\gamma)^2}.$$

Hence, for every state $s \in S$,

$$|V_1^\pi(s) - V_2^\pi(s)| \leq \|\delta\|_\infty \leq \frac{\alpha + \gamma R_{\max} \beta}{(1-\gamma)^2}.$$

This is the desired result.

3 Policy Evaluation in RiverSwim

We provide here a short report on the Monte Carlo simulation and exact computation of V^π for the RiverSwim MDP, where the policy π takes *right* action with probability 0.65 in states $\{1, 2, 3\}$ and always takes *right* in states $\{4, 5\}$. The code used to run the experiments is contained in `HA2_RiverSwim.py`.

i) Monte Carlo Estimation of V^π .

We generated $n = 50$ trajectories of length $T = 300$ each, for each possible start state, accumulating returns

$$G = \sum_{t=0}^{T-1} \gamma^t r_t,$$

and then averaged over the simulated trajectories to obtain an approximate value $V^\pi(s)$. We used the discount factor $\gamma = 0.96$. Below are the resulting Monte Carlo estimates (in a few seconds of execution time):

State	MC Estimate	Exact Value
1	4.01793	4.12090
2	4.61536	4.71119
3	5.96556	6.33460
4	9.47802	9.73803
5	11.21253	11.17784

ii) Exact Computation using the Bellman Equation.

We also computed the exact value function

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi$$

by constructing the transition matrix P^π and reward vector r^π under policy π , and then numerically solving the linear system $(I - \gamma P^\pi) v = r^\pi$ in Python with `numpy.linalg.solve`. The table above (right column) presents the resulting exact values of $V^\pi(s)$ for $s = 1, \dots, 5$.

Comment: Although the Monte Carlo estimates are not the finest approximation and slightly deviate from the exact values (particularly in states 3 and 4) due to limited sample size, they were obtained in just a few seconds of execution. Increasing the number of trajectories (or their length) would make the approximation even closer in practice.

4 Solving a Discounted Grid-World

All of the Python code used to run these experiments (i.e. PI, VI, and Anchored VI, plus the 4-room environment and the visualizations) can be found in the file `HA2_gridworld.py`.

Environment Setup:

We have a 7×7 grid with walls, yielding 20 accessible states labeled 0 to 19. State 19

(bottom-right corner) has reward 1 and is effectively absorbing once reached. The agent chooses from 4 compass actions (0 = Up, 1 = Right, 2 = Down, 3 = Left), but transitions are *slippery*: probability 0.7 for the chosen direction, 0.1 each for perpendicular directions, and 0.1 for staying in place. We set $\gamma = 0.97$ by default, except in part (iii) where $\gamma = 0.998$.

(i) Solve the grid-world task using PI (Policy Iteration).

We implemented *Policy Iteration*, which alternates:

$$\text{Policy Iteration: } \begin{cases} \text{(a) Evaluate current policy } \pi : & V^\pi = (I - \gamma P^\pi)^{-1} r^\pi, \\ \text{(b) Improve } \pi : & \pi \leftarrow \arg \max_a \left[r(s, a) + \gamma \sum_{s'} P(s' | s, a) V^\pi(s') \right]. \end{cases}$$

It converged in 5 iterations. The resulting *optimal policy* (array of size 20) is

$$\pi_{\text{PI}} = [1, 1, 1, 2, 2, 2, 0, 2, 3, 2, 2, 1, 1, 1, 1, 2, 1, 0, 1, 0].$$

The *optimal value function* $V^*(s)$, rounded to 2 decimals, is

$$V^* = [23.07, 23.97, 25.15, 26.26, 25.40, 23.97, 23.07, 27.71, 26.40, 25.15, 29.12, 26.26, 27.71, 29.12, 30.40, 31.74, 25.40, 26.40, 31.74, 33.33].$$

We can visualize π on the 7×7 grid as:

```
[ [Wall, Wall, Wall, Wall, Wall, Wall, Wall],
  [Wall, Right, Right, Right, Down, Down, Wall],
  [Wall, Down, Up, Wall, Down, Left, Wall],
  [Wall, Down, Wall, Wall, Down, Wall, Wall],
  [Wall, Right, Right, Right, Right, Down, Wall],
  [Wall, Right, Up, Wall, Right, Up, Wall],
  [Wall, Wall, Wall, Wall, Wall, Wall, Wall] ].
```

(ii) Implement VI and use it to solve the grid-world task.

We then implemented *Value Iteration*, repeatedly applying

$$V_{n+1}(s) = \max_a \left[r(s, a) + \gamma \sum_{s'} P(s' | s, a) V_n(s') \right].$$

With $\gamma = 0.97$, VI converged in 48 iterations and recovered *the same* optimal policy and value function as in part (i). The same map display applies.

(iii) Repeat (ii) with $\gamma = 0.998$.

Now we let $\gamma = 0.998$. VI converged more slowly (56 iterations). The *value function* is larger (about 487–500 in some states), and the policy is slightly modified in some early

states. We still see a path leading to state 19. Overall, it remains optimal, just at a bigger scale of values.

Regarding the increasing number of iterations to converge that is because when γ is larger (closer to 1), the updates contract the value function differences more slowly, so it typically takes *more* iterations to reduce $\|V_{n+1} - V_n\|_\infty$ below a given threshold. Intuitively, a higher γ means future rewards are weighted more heavily, causing bigger swings or slower settling of the estimated values across states, and thus lengthening the convergence process.

(iv) Anchored Value Iteration (Anc-VI) with different initial points.

We added an anchoring approach with:

$$V_{n+1} = \beta_{n+1} V_0 + (1 - \beta_{n+1}) \max_a \{ \dots \}.$$

Using three different anchors:

- (a) $V_0 = 0$ **(614 iterations to converge)**,
- (b) $V_0 = 1$ **(613 iterations to converge)**,
- (c) random V_0 in $[0, 1/(1 - \gamma)]^{nS}$ **(606 iterations to converge)**,

they all converged to an *optimal policy* as well, matching the same final V^* observed before.

(v) Compare the convergence speed of VI vs. Anc-VI (with the same anchor).

Finally, we compared standard VI (`anc=False`) and the anchored variant (`anc=True`) for each available initialization of the values of V at time 0:

	$V_0 = 0$	$V_0 = 1$	$V_0 = \text{Random}$
Standard VI	592iters	591iters	583iters
Anchored VI	614iters	613iters	606iters

In this experiment, the *anchored* method needed slightly *more* iterations than the standard VI. However, they still converged to optimal policies. With different discount factors or parameters, anchored VI can sometimes speed up or slow down convergence.

5 Off-Policy Evaluation in Episode-Based River-Swim