Online and Reinforcement Learning (2025)
Home Assignment 1

Davide Marchi 777881

Contents

1 Find an online learning problem from real life

Exercise 5.1 (Find an online learning problem from real life). Find two examples of real life problems that fit into the online learning framework (online, not reinforcement!). For each of the two examples explain what is the set of actions an algorithm can take, what are the losses (or rewards) and what is the range of the losses/rewards, whether the problem is stateless or contextual, and whether the problem is i.i.d. or adversarial, and with full information or bandit feedback

1.1 Post Recommendation on Social Media

Deciding in real time which article or advertisement to display to a user.

- **Actions:**
  The algorithm chooses one item (news article or ad) from a finite set of options.

- **Reward:**
  1 if the user clicks on the suggested content, 0 if the user does not (range [0, 1]).
  In more sophisticated systems, the reward could be based on the time spent by the user on the suggested content.

- **Stateless vs. Contextual:**
  In the simplest implementation, it could be stateless, basing recommendations exclusively on the outcomes of previous interactions. However, in actual implementations, it is contextual, as the algorithm considers the user's profile and related information.

- **Environment (i.i.d. vs. Adversarial):**
  While an idealized model might assume users arrive from an i.i.d. process, real-world user behavior is often adversarial (or non-stationary) as preferences and trends shift over time.

- **Feedback:**
  The feedback is bandit feedback; the algorithm only observes the outcome (click or no click) for the displayed item, not for all items in the set.

1.2 Pricing of Products Over Time

Deciding in real time with what price to sell a product to maximize profit.

- **Set of Actions:**
  Each day, the algorithm can change the price of a product, selecting from a fixed number of price options.

- **Reward:**
  The reward is defined as the profit obtained at the end of the day, which is the revenue from sales minus the cost of production.
  The range of the reward is $(0, +\infty)$ since we assume that none of the prices are lower than the production cost.

- **Stateless vs. Contextual:**
  The problem could be considered stateless if the algorithm only considers the prices and profits of the previous days.

- **Environment (i.i.d. vs. Adversarial):**
  The environment could be considered i.i.d. if the demand for the product remains constant over time. However, in reality, it is adversarial, as demand can change over time, especially for products that are not purchased repeatedly.

- **Feedback Type:**
  The feedback in dynamic pricing is of the bandit type, as the algorithm only observes the reward for the chosen action (applied price).

2 Follow The Leader (FTL) algorithm for i.i.d. full information games

Setting

- We have **two actions**, a^* and a, with rewards in $[0, 1]$.
- The rewards for each action are *i.i.d. across rounds*, and *fully observed* each round.
- Let $\mu(a)$ be the true expected reward of action a. Without loss of generality, let

$$a^* \;=\; \arg\max_a \mu(a).$$

- Define the *gap* of the suboptimal arm by

$$\Delta \;=\; \mu(a^*) - \mu(a) \;>\; 0.$$

- We consider T rounds, indexed by $t = 1, 2, \ldots, T$.

Algorithm: Follow the Leader (FTL)

- **Initialization:** In the very first round(s), you may pick any arm(s).
- **At each round $t \geq 2$:**
  1. For each arm $b \in \{a, a^*\}$, compute its *empirical mean* based on *all* observed rewards of b so far:
  $$\hat{\mu}_{t-1}(b) \;=\; \frac{1}{t-1} \sum_{s=1}^{t-1} X_s(b),$$
  where $X_s(b)$ is the reward observed for arm b at round s.
  2. *Choose* the arm whose empirical mean is larger:
  $$A_t \;=\; \arg\max_{b \in \{a, a^*\}} \hat{\mu}_{t-1}(b).$$

Because this is a **full-information** setting, we observe $X_t(a)$ and $X_t(a^*)$ for both arms at every round t, not just the chosen one.

Key Idea: Probability of Confusion

Let us define the event

$$\big\{\text{“FTL picks the suboptimal arm } a \text{ at round } t\text{”}\big\} \quad\Longleftrightarrow\quad \big\{\,\hat{\mu}_{t-1}(a) \;\geq\; \hat{\mu}_{t-1}(a^*)\big\}.$$

Because $\Delta = \mu(a^*) - \mu(a) > 0$, the only way for $\hat{\mu}_{t-1}(a)$ to overtake $\hat{\mu}_{t-1}(a^*)$ is if

$$\big[\hat{\mu}_{t-1}(a) - \mu(a)\big] \;-\; \big[\hat{\mu}_{t-1}(a^*) - \mu(a^*)\big] \;\geq\; \Delta.$$

By Hoeffding's inequality (for $[0,1]$-bounded i.i.d. samples),

$$\mathbb{P}\Big(\hat{\mu}_n(b) - \mu(b) \;\geq\; \epsilon\Big) \;\leq\; \exp\big(-2\,n\,\epsilon^2\big),$$

one sees that

$$\mathbb{P}\Big(\hat{\mu}_{t-1}(a) \;\geq\; \hat{\mu}_{t-1}(a^*)\Big) \;\leq\; \exp\big(-c\,(t-1)\,\Delta^2\big) \quad \text{for some constant } c > 0.$$

We will sometimes refer to this as the *probability of confusion* at round t.

Bounding the Regret

Regret definition. The (pseudo-)regret is

$$R_T \;=\; \sum_{t=1}^{T} \big[\mu(a^*) \;-\; \mu(A_t)\big].$$

Since $A_t \in \{a, a^*\}$, the only time we incur regret $\Delta > 0$ is precisely when $A_t = a$. Thus,

$$R_T \;=\; \Delta \sum_{t=1}^{T} \mathbf{1}\{\text{FTL picks } a \text{ at round } t\}.$$

Taking expectation,

$$\mathbb{E}[R_T] \;=\; \Delta \sum_{t=1}^{T} \mathbb{P}\big(A_t = a\big) \;=\; \Delta \sum_{t=1}^{T} \mathbb{P}\Big(\hat{\mu}_{t-1}(a) \;\geq\; \hat{\mu}_{t-1}(a^*)\Big).$$

Using the exponential bound from above, define

$$\delta(t) \;=\; \exp\big(-c\,(t-1)\,\Delta^2\big).$$

Hence,

$$\mathbb{E}[R_T] \;\leq\; \Delta \sum_{t=1}^{T} \delta(t).$$

Geometric series. Note that $\sum_{t=1}^{\infty} \exp(-c\,(t-1)\,\Delta^2)$ converges (it is a geometric series). Concretely,

$$\sum_{t=1}^{\infty} \exp\bigl(-c\,(t-1)\,\Delta^2\bigr) \;=\; \frac{1}{1-e^{-\,c\,\Delta^2}} \;<\; \infty.$$

Hence there is a constant $C(\Delta)$ such that

$$\sum_{t=1}^{T} \exp\bigl(-c\,(t-1)\,\Delta^2\bigr) \;\le\; C(\Delta), \quad \text{independently of } T.$$

Multiplying by Δ,

$$\mathbb{E}[R_T] \;\le\; \Delta\, C(\Delta).$$

That is, *the total expected regret* remains **bounded** by a constant with respect to T.

Final Statement of the Bound

Because FTL eventually "locks onto" the better arm (and stays there with high probability), we conclude

$$R_T \;=\; O(1) \quad \text{as } T \to \infty.$$

A more explicit expression for the constant bound is

$$\mathbb{E}[\,R_T\,] \;\le\; \Delta\,\frac{1}{1-e^{-\,c\,\Delta^2}} \;=\; \frac{\Delta}{1-e^{-\,c\,\Delta^2}},$$

which is finite for every $\Delta > 0$. If there are multiple suboptimal arms $a = 1, \ldots, K-1$, one simply sums similar terms for each suboptimal $\Delta(a)$; the result is still a finite constant.

Comparison with the Bandit Setting

In the professor's *bandit* slides, one observes only the reward of the chosen arm each round and thus must *explore* explicitly, leading to a regret growing like $\log T$. Here, in *full-information* feedback, both arms' rewards are seen every time, so $\hat{\mu}_t(a)$ and $\hat{\mu}_t(a^*)$ concentrate exponentially fast for *both* arms. As a result, FTL typically makes only a finite number of mistakes (suboptimal plays), yielding a **constant** regret bound.

3 Improved Parametrization of UCB1

Below is a figure showing the **empirical pseudo-regret** of two UCB algorithms: "original UCB" (red) vs. "modified UCB" (blue), for three different gap values $\Delta = 0.25, 0.125,$ and 0.0625. Each curve is the average over 20 runs of length $T = 100000$, with shaded ± 1 standard-deviation bands.

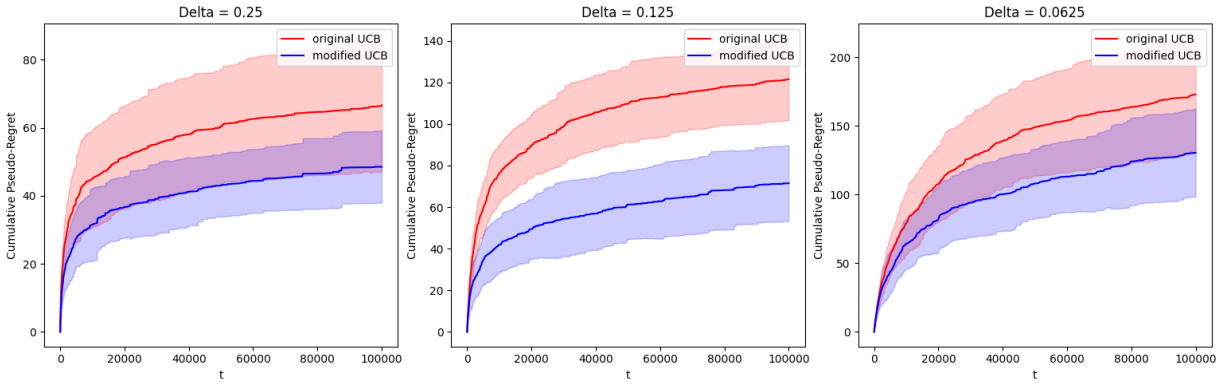

Figure 1: Empirical pseudo-regret over time for different values of Δ.

3.1 Plot Description

- **Horizontal axis**: time $t \in \{1, \ldots, 100000\}$.
- **Vertical axis**: cumulative pseudo-regret $\widehat{R}_t$, i.e. $\sum_{s=1}^{t} \Delta(A_s)$, where $\Delta(A_s)$ is the gap between the chosen arm's mean and the optimal arm's mean at time s.
- **Subplots**: from left to right, we show the same experiment but with $\Delta = 0.25,\ 0.125,$ and 0.0625.

In all three cases, the **blue (modified UCB)** curve remains below the **red (original UCB)** curve, indicating the modified parametrization typically achieves lower pseudo-regret. We also see that for smaller Δ (rightmost plot), overall regret magnitudes are higher.

3.2 1) Which values of Δ lead to higher regret?

We see from the three subplots that:

- $\Delta = 0.25$ (left) yields smaller final regrets, on the order of ≈ 80 for the original UCB and ≈ 50 for the modified UCB by $t = 100000$.
- $\Delta = 0.125$ (middle) has higher final regret (≈ 140 vs. ≈ 100).
- $\Delta = 0.0625$ (right) yields still higher regrets (≈ 220 vs. ≈ 160).

Hence, **the smaller Δ is, the higher the pseudo-regret** tends to be. Intuitively, when the arms' means are closer together, it is harder to distinguish the optimal arm from the suboptimal arm; more exploration is needed before we can reliably exploit.

3.3 2) Relative performance of the two parametrizations

Throughout the plots, the **modified UCB** ("blue") systematically attains lower pseudo-regret than the "original UCB" ("red"), and the gap between them grows larger over time. This is consistent with the theoretical claim that using a $\sqrt{\ln t}$ (rather than $\sqrt{\ln t^3}$ or similar) confidence term can remove an unnecessary union-bound factor, thereby tightening the exploration bonus and improving performance. In short:

- **Modified UCB** explores more selectively, resulting in **lower regret** in practice.
- **Original UCB** maintains a bigger confidence-radius term and thus "over-explores" somewhat, leading to a higher cumulative regret curve.

Hence, **the modified parametrization outperforms the original** for all three gap values, especially noticeable when the gap is small.

4 Example of Policies in RiverSwim

Part 1

We classify the given policies into one of the four policy classes: Π^{SD} (stationary deterministic), Π^{SR} (stationary randomized), Π^{HD} (history-dependent deterministic), and Π^{HR} (history-dependent randomized).

(i) π_a: Swim to the right if the current state is not 1; otherwise, swim to the left.

Analysis: This policy depends only on the current state, meaning it is **stationary**. Furthermore, it selects actions deterministically (without randomness).

Conclusion: $\pi_a \in \Pi^{SD}$.

(ii) π_b: If the time slot t is an even integer, swim to the right; otherwise, flip a fair coin and swim right (resp. left) if the outcome is 'Head' (resp. 'Tail').

Analysis: Since the decision process explicitly depends on t, the policy is **history-dependent**. Additionally, since the agent flips a coin on odd time steps, it involves randomness.

Conclusion: $\pi_b \in \Pi^{HR}$ (history-dependent randomized).

(iii) π_c: At $t = 1$, swim to the left. For $t > 1$, swim to the right if the index of the previous state is odd; otherwise, swim to the left.

Analysis: The decision process depends on both the time step t and the previous state, meaning the policy is **history-dependent**. However, it does not involve randomness.

Conclusion: $\pi_c \in \Pi^{HD}$ (history-dependent deterministic).

(iv) π_d: Flip a fair coin. If the outcome is 'Head' and the current state is either $L-1$ or L, then swim to the right; otherwise, swim to the left.

Analysis: The decision depends only on the current state, meaning it is **stationary**. However, since it involves coin flips, it is a **randomized** policy.

Conclusion: $\pi_d \in \Pi^{SR}$ (stationary randomized).

(v) π_e: If it rains, swim to the right; otherwise, swim to the left. (Raining is independent of the agent's swimming direction and position.)

Analysis: The decision at each time step depends only on the current state and an external random variable (whether it rains). Since rain is independent of past history, the policy is **stationary** but **randomized**.

Conclusion: $\pi_e \in \Pi^{SR}$ (stationary randomized).

Part 2

We construct an example of a history-dependent deterministic policy in RiverSwim that is not stationary. The proposed policy is:

> *Swim two times to the right, then rest for one step by following the current and actively swimming to the left. Repeat this pattern indefinitely.*

Analysis: The policy follows a structured cycle (right-right-left) based on past actions, meaning it is **history-dependent**. Since the decision at any given step is fully determined by the policy's rule, it is **deterministic**.

Conclusion: This policy belongs to Π^{HD} (history-dependent deterministic).

5 Policy Evaluation in RiverSwim

Moved to Home Assignment 2.

6 Robbing Banks

Formulation of the Robber-Police Game as an MDP

(i) State and Action Spaces.

We define each state s as a 4-tuple:

$$s = \big(r_{\text{agent}},\, c_{\text{agent}},\, r_{\text{police}},\, c_{\text{police}}\big),$$

where

$$r_{\text{agent}},\, r_{\text{police}} \in \{0,1,2,3,4,5\}, \quad c_{\text{agent}},\, c_{\text{police}} \in \{0,1,2\}.$$

Therefore, the total number of states is

$$|S| \;=\; 6 \times 3 \;\times\; 6 \times 3 \;=\; 324.$$

For simplicity, we take the initial state as

$$s_{\text{init}} \;=\; (0,\, 0,\, 1,\, 2).$$

The agent has five possible actions at each step:

$$A \;=\; \{\uparrow,\, \downarrow,\, \leftarrow,\, \rightarrow,\, \text{stay}\}.$$

Hence, $|A| = 5$.

(ii) Reward Function.

We define the per-step reward function $R(s, a)$ as follows:

$$R(s,a) \;=\; \begin{cases} 100000, & \text{if the agent is on a bank and the police is not there,} \\ -10000, & \text{if the agent and the police share the same coordinates,} \\ 0, & \text{otherwise.} \end{cases}$$

We then apply the usual discounted return framework with discount factor $\gamma \in (0, 1)$. As discussed in class, we do not explicitly encode γ into the states.

(iii) Transition Probabilities.

We now specify the transition kernel

$$P\big(s' \mid s, a\big),$$

focusing on the example where the agent is at Bank 1 and the police is at Bank 4. In coordinates, let

$$s \;=\; (\,0,\, 0,\, 0,\, 2).$$

(Here $(0, 0)$ denotes Bank 1 in row 0, column 0, and $(0, 2)$ denotes Bank 4 in row 0, column 2.)

We assume that *walls act as reflectors*. Concretely, if either the agent or the police attempts to move outside the grid, they are moved in the opposite direction by the same distance. For instance:

- If the agent (or police) is at row 0 and attempts to move $\uparrow$, they end up at row 1.
- If at column 0 and an action tries to move $\leftarrow$, they end up at column 1.
- All other moves within the grid follow the usual row or column increment/decrement (bounded between row 0 and row 5, and column 0 and column 2).

Agent's move. From $(r_{\text{agent}},\, c_{\text{agent}}) = (0,\, 0)$, the next agent position $(r'_{\text{agent}},\, c'_{\text{agent}})$ is:

- $\uparrow$: attempts to go out of the grid (row -1), then reflects and ends at $(1,\, 0)$.
- $\downarrow$: moves to $(1,\, 0)$ normally (assuming row 0 is top).
- $\leftarrow$: attempts $(0,\, -1)$, reflects to $(0,\, 1)$.
- $\rightarrow$: moves to $(0,\, 1)$ normally.
- stay: remains at $(0,\, 0)$.

Police's move. Meanwhile, since the police is in the same row as the agent but to the right $\big((0, 2)$ vs. $(0, 0)\big)$, the problem statement tells us that it moves $\uparrow$, $\downarrow$, or $\leftarrow$ with probability $1/3$ each, again with reflection at the walls as described. Specifically:

$$\begin{aligned}
&\text{With probability } \tfrac{1}{3}: \text{ police tries "Up";}\\
&\quad \text{since it's at row 0, it bounces to row 1, yielding } (1, 2).\\
&\text{With probability } \tfrac{1}{3}: \text{ police tries "Down";}\\
&\quad \text{this is valid and leads to } (1, 2).\\
&\text{With probability } \tfrac{1}{3}: \text{ police tries "Left";}\\
&\quad \text{this moves it from } (0, 2) \text{ to } (0, 1).
\end{aligned}$$

Combining to get next-state probabilities. The next state $s' = (r'_{\text{agent}}, c'_{\text{agent}}, r'_{\text{police}}, c'_{\text{police}})$ is thus determined by the chosen action a (the agent's move, which is deterministic given reflection rules), and then by the police's random move (with probabilities $1/3$ or $1/4$ depending on whether they are on the same row/column or neither). More explicitly, for the example action $a = \rightarrow$, the agent's location becomes $(0, 1)$. The police's location then becomes:

$(1, 2)$ with probability $\frac{2}{3}$ (sum of Up/Down probabilities), $(0, 1)$ with probability $\frac{1}{3}$.

So the next state is

$$(0, 1,\, 1, 2) \text{ with probability } \tfrac{2}{3}, \qquad (0, 1,\, 0, 1) \text{ with probability } \tfrac{1}{3}.$$

One can similarly list the possible next states and their probabilities for the other agent actions.

In this manner, the transition probabilities $P(s' \mid s, a)$ are obtained for all $s \in S$ and $a \in A$. When the agent is caught, one may either define an immediate penalty and restart the game in state $(0, 0,\, 1, 2)$, or encode that penalty in $R(s, a)$ as above (i.e. -10000) and transition to a reset state.