COMP-579: Reinforcement Learning - Assignment 1

Posted Wednesday January 14, 2026
Due Wednesday, January 28, 2026

The assignment can be undertaken individually or in teams of two. Alongside this document, you have been provided with a partially completed .IPYNB (Jupyter Notebook) file. Your task is to complete the .IPYNB file by adding the necessary code, displaying results, and providing explanations for questions. **For the theory part, all written answers must be typed (computer-written); handwritten submissions will not be accepted.**

Following the notation and interfaces of the predefined classes and functions is mandatory; however, you may adapt variable names and intermediate notations in your own code to better align with your thought process and coding style. For coding questions, ensure that you only submit the completed .IPYNB file for evaluation.

You must submit two separate files: one PDF containing all theory answers, and one completed `.ipynb` file containing all code, results, and code-related explanations. **You must also include an author contribution statement that explains the contributions of the various team members in detail. Additionally, you are required to include a detailed statement on LLM usage for every aspect of usage in the assignment.**

If you have any questions or require clarification, feel free to reach out for assistance on the Ed. We will try to answer your questions within 24 hours. Good luck with your assignment!

Theoretical questions[45 points]

Application of the Lai-Robbins bound The asymptotic lower bound on the total regret L_T for any consistent bandit algorithm is given by the Lai-Robbins bound:

$$\liminf_{T\to\infty} \frac{\mathbb{E}[L_T]}{\ln T} \geq \sum_{a:\Delta_a>0} \frac{\Delta_a}{D_{KL}(P_a \| P_*)},$$

where D_{KL} is the Kullback-Leibler divergence between the distribution of a suboptimal arm a (P_a) and the optimal arm (P_*), and Δ_a is the gap in expected reward between the optimal arm and arm a.

1. [7 points] Derive the explicit formula for the KL-divergence between two Bernoulli distributions with parameters p and q: $D_{KL}(\text{Ber}(p)\|\text{Ber}(q))$.

2. [7 points] Same question for two Gaussian distributions sharing the same variance.

3. [7 points]Show that for Bernoulli bandit, it is "easier" (i.e., theoretically implies lower regret) to distinguish an arm with mean $p = 0.9$ from an optimal arm with $p_* = 0.99$ than it is to distinguish an arm with $p = 0.55$ from an optimal arm with $p_* = 0.64$, even though the difference in means is identical ($\Delta = 0.09$) in both cases. What about the Gaussian case?

A proof for a simplified Lai-Robbins bound The Lai-Robbins bound states that regret grows with $\frac{1}{D_{KL}}$. To understand why, let's analyze a simplified "Best Arm Identification" problem.

Consider two Bernoulli arms:

- Arm 1 has a known success probability $p_1 = 0.5$.
- Arm 2 has an unknown success probability $p_2 = 0.5 + \Delta$

You want to determine if Arm 2 is better than Arm 1 with high confidence. You collect n samples from Arm 2 and compute the empirical mean $\hat{p}_n$. You decide Arm 2 is "Better" if $\hat{p}_n > 0.5$.

1. [8 points] Suppose the truth is that Arm 2 is actually *worse* ($\Delta < 0$). Use Hoeffding's Inequality to find an upper bound on the probability that you incorrectly classify it as "Better" (i.e., $P(\hat{p}_n > 0.5)$) after n samples.
2. [8 points] Set this error probability to be at most δ (e.g., $\delta = 1/T$). Rearrange your bounds to show that the number of samples n required to avoid this error must be at least:

$$n \geq \frac{\ln(1/\delta)}{2\Delta^2}$$

3. [8 points] For small gaps, $D_{KL} \approx 2\Delta^2$. Substitute D_{KL} into the inequality above. Explain how this explains the Lai-Robbins term $\frac{\ln T}{D_{KL}}$.

Bandit algorithms [55 points]

For this assignment, you will carry out some experimentation with bandit algorithms, in order to help you understand what we discussed in class, and to get used to the way in which we will run experiments for other assignments as well. You should submit a notebook with your code, results and explanations.

1. [5 points] Write a small simulator for a **Bernoulli bandit** with k arms. Each arm $i \in \{1, \ldots, k\}$ has a success probability p_i (provided as input). When you pull arm i, the reward is a Bernoulli random variable:

$$r \sim \text{Bernoulli}(p_i),$$

   meaning $r = 1$ with probability p_i and $r = 0$ with probability p_i.

   Your bandit should implement a function called `sample` that takes as input the index of an action (arm) and returns one reward sample (0 or 1).

   Use $k = 3$ arms with parameters

$$p_* = [0.5,\ 0.5 - \delta,\ 0.5 + \delta], \quad \delta = 0.2,$$

   so $p_* = [0.5,\ 0.3,\ 0.7]$.

   For each action, generate and save 50 samples. Then create one plot per action showing:

   - the 50 reward samples (y-values are 0 or 1) versus draw index $t = 1, \ldots, 50$,
   - a horizontal line for the empirical mean of the 50 samples,
   - a horizontal line for the true probability p_i.

Each plot should have an x-axis from 1 to 50, and clearly label the empirical mean and true p_i.

2. [5 points] Code two rules for estimating action values. First, implement a function `update` that updates the estimate using a fixed learning rate α. Second, implement a function `updateAvg` that updates the estimate using the incremental computation of the sample mean.

   Using the data generated in the previous question, plot for each action the estimated q value as a function of the number of samples. On each plot, include curves corresponding to the incremental mean method and fixed learning rates $\alpha = 0.01$, $\alpha = 0.05$, and $\alpha = 0.1$, along with a horizontal line indicating the true value of q.

   Each graph should have the number of samples on the x-axis and the estimated q value on the y-axis, and should clearly distinguish between the different update methods.

3. [5 points] Repeat the above experiment 100 independent times, with all action-value estimates initialized to 0. Each run should contain 100 samples for each action.

   For each action, plot the same type of graph as in the previous question, but now show the average estimated q value over the 100 runs. Include a measure of variability across runs (e.g., standard error using error bars or a shaded region). Plot curves for the incremental mean method and for fixed learning rates $\alpha = 0.01$, $\alpha = 0.05$, and $\alpha = 0.1$, together with a horizontal line indicating the true value of q.

   In a few sentences, explain what you observe. Describe how different values of α affect the speed of learning and the stability of the estimates, and which learning rate performs best overall. Based on the observed behavior, explain the underlying phenomenon (e.g., the bias–variance tradeoff) and indicate the range of α values you would explore to further improve performance.

4. [10 points] Code the ϵ-greedy algorithm discussed in class, with averaging updates, with ϵ provided as an input. Additionally, implement a decaying ϵ such that $\epsilon_t = \frac{\epsilon_0}{1+\lambda t}$, where ϵ_0 is the initial ϵ value and λ is a decay rate constant. Note that if multiple actions have maximum value, you should choose randomly among them (and not always pick, e.g., the one with the lowest index). You will run 100 independent runs, each consisting of 1000 time steps. Plot the following graphs:

   (a) The reward received over time, averaged at each time step over the 100 independent runs (with no smoothing over the time steps), and the standard error over the 100 runs.

   (b) The cumulative reward received over time, averaged over the 100 independent runs.

   (c) The running average reward up to each time step, averaged over the 100 independent runs.

   (d) The instantaneous regret I_t (averaged over the 100 runs).

   (e) The total regret I_t up to time step t (averaged over the 100 runs).

   Generate this set of graphs for the following values of ϵ: 0, 1/8, 1/4, 1/2, 1, and for a decaying ϵ with $\epsilon_0 = 1/2$ and $\lambda = 0.1$. Note that the x-axis for the graph will go up to 1000, and you should have on each graph a set of 6 curves, one for each value of ϵ. Explain what you observe in the graphs and discuss the effect of ϵ, including the decaying ϵ.

5. [10 points] Implement the gradient bandit algorithm discussed in class, using softmax (Boltzmann) action selection. The algorithm maintains a preference $H_t(a)$ for each action a, and selects actions according to:
$$\pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$
Use a baseline (average of past rewards) to normalize the rewards when updating preferences via gradient ascent. Note that, for Boltzmann exploration $H_t(a)$ takes a particular form: $H_t(a) = Q_t(a)/T$, where T is the temperature parameter.

   Run 100 independent runs, each with 1000 time steps. Plot the same 5 graphs as in the previous question for $T = 0.05$, $T = 0.1$, $T = 0.5$, and $T = 1.0$. Explain briefly the behavior you observe and the effect of different temperature values.

6. [10 points] Implement Thompson Sampling for the Bernoulli bandit using Beta priors. Run 100 independent runs of 1000 time steps. Plot the same 5 graphs as in the previous questions, comparing the following prior initializations: Beta$(1, 1)$, Beta$(10, 1)$, and Beta$(1, 10)$.

   Explain briefly how the choice of prior affects early exploration behavior. Do the priors have a significant long-term effect? Why or why not?

7. [10 points] Let us now consider a non-stationary problem. Use $\delta = 0.2$ as in Q1, with initial parameters $p_* = [0.5, 0.5-\delta, 0.5+\delta]$. Every 500 time steps, actions 2 and 3 swap their probabilities (i.e., $0.5 - \delta \leftrightarrow 0.5 + \delta$), alternating which action is optimal.

   Run for 2000 time steps total, with 100 independent runs. Plot the same 5 graphs as in the previous questions, comparing the following algorithms:

   ϵ-greedy:

   - $\epsilon = 1/8$ with $\alpha = 0.1$
   - $\epsilon = 1/8$ with incremental averaging
   - Decaying ϵ with $\alpha = 0.1$, where $\epsilon_t = \frac{\epsilon_0}{1+\lambda t}$, $\epsilon_0 = 1/4$, $\lambda = 0.1$

   Gradient bandit: $T = 0.1$ and $T = 0.5$

   Thompson sampling: Beta(1,1) prior

   Plot the average instantaneous reward over time. Explain what you observe, particularly how each algorithm adapts when the optimal action changes. Which algorithm is best suited to cope with non-stationarity?