Lab 1: Bandits and Contextual Bandits in News Recommendation

Sindri Magnússon

Overview

In this lab, you take on the role of a data scientist at **NewsNet**, a company that personalizes news recommendations. Every time a user logs in, you observe their profile (context) and recommend one of several articles. The user may *click* (reward = 1) or *ignore* (reward = 0).

You must learn to make better decisions over time by balancing **exploration** and **exploitation**. This is modeled using **bandit algorithms**.

Part 1: Multi-Armed Bandits (MAB)

Bandit Environment

We use an MAB **environment** to simulate the news recommendation scenario, as implemented in the provided notebook code. Each article (or *arm*) has an unknown probability of being clicked. When you recommend an article (choose an arm), the environment returns a reward of 1 (user clicks) or 0 (user ignores). Your goal is to learn which article performs best through interaction with the given environment.

Exploration Strategies and Evaluation

We compare two exploration strategies for the multi-armed bandit problem: **Epsilon-Greedy** and **Upper Confidence Bound (UCB)**. Both methods aim to balance *exploration* (trying different actions to gather information) and *exploitation* (choosing the action that currently seems best) in order to maximize the overall reward over time.

We evaluate their performance using two common measures: the **average reward** and the **regret**.

Average reward. The average reward up to time T is defined as

$$\bar{r}_T = \frac{1}{T} \sum_{t=1}^{T} r_t,$$

where r_t is the reward obtained at time step t. A higher average reward indicates that the algorithm is making better decisions on average.

Regret. Regret measures how much the algorithm loses compared to always selecting the optimal arm. Let μ_i denote the true expected reward (click probability) of arm i and $\mu^* = \max_i \mu_i$ be the expected reward of the best arm. The cumulative regret after T steps is

$$\texttt{Regret}_T = \sum_{t=1}^{T} \left(\mu^* - \mu_{a_t}\right),$$

where a_t is the arm chosen at time t.

Regret quantifies the efficiency of learning—how quickly the algorithm approaches optimal behavior. While we can compute $\texttt{Regret}_T$ in a simulated environment (since μ_i are known), it cannot be measured directly in a real-world system where the true optimal arm is unknown. Nevertheless, it provides valuable theoretical insight into the algorithm's performance and is widely used as a benchmark in research on bandit algorithms.

Task 1.a: Epsilon-Greedy Exploration

Goal: Implement the epsilon-greedy strategy to learn the best arm.

Instructions:

1. Initialize $Q(a) = 0$ and $N(a) = 0$ for all arms.
2. At each time step $t = 1, \ldots, 1000$:
   - With probability ε, choose a random arm.
   - Otherwise, choose the arm with the highest $Q(a)$.
   - Pull the arm, observe reward r_t.
   - Update $Q(a)$ using:

$$Q(a) \leftarrow Q(a) + \frac{1}{N(a)}(r_t - Q(a))$$

Hyperparameters:

- Try $\varepsilon = 0.1$, 0.01, 0.2.

Output:

- Plot average reward vs time.
- Plot the regret vs time.
- Plot how many times each arm was selected.
- Plot the same figures for a random action policy.

Be prepared to explain what you have done and the main results.

Task 1.b: UCB Exploration

Goal: Implement the Upper Confidence Bound algorithm.

Instructions:

1. At each time step t, select:

$$a_t = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\log t}{N(a)}} \right]$$

2. Pull arm a_t, observe reward, and update $Q(a)$.

Hyperparameters:

- Try $c = 1, 2, 5$.

Output:

- Plot the same figures as in Task 1 (average reward, regret, and how many times each arm was selected) for the UBC algorithm.
- Compare your results with both the random action policy and epsilon-greedy, and provide a brief interpretation of the differences.

Be prepared to explain what you have done and the main results.

Part 2: Contextual Bandits

In this part, we extend the bandit problem by introducing a **context** that represents user features. Instead of treating all users the same, the algorithm now observes a feature vector

$$x_t \in \mathbb{R}^d,$$

which captures user characteristics such as political interest, sports preference, technical orientation, mobile usage, morning reading habits, and age.

At each time step t, the agent observes the context x_t, chooses an article (arm) a_t, and receives a binary reward $r_t \in \{0, 1\}$, where $r_t = 1$ indicates that the user clicked the article.

The goal is to learn a recommendation strategy that adapts to user features and maximizes the expected number of clicks over time.

Context and Actions in the Simulation

In this example, we consider a setting with $d = 6$ contextual features and $k = 4$ possible articles (arms).

Each user is represented by a vector $x_t = [x_{t,1}, \ldots, x_{t,6}]$ that encodes these attributes. Users are generated from different "types" (e.g., sports-oriented, tech-savvy, or morning readers) with small random variations to make the environment realistic.

The environment defines a true (but unknown) parameter vector $\theta_a \in \mathbb{R}^6$ for each article a, which determines the probability of a click given the user's context x_t. The reward is drawn as a Bernoulli random variable:

$$r_t \sim \text{Bernoulli}\big(\sigma(\theta_a^\top x_t)\big), \quad \text{where} \quad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Table 1: Contextual features used in the simulation ($d = 6$).

Feature	Description	Example values (approx.)
likes_politics	Interest in political content	1.5 (strong), 0.2 (low)
sports_fan	Interest in sports content	1.8 (strong), 0.1 (weak)
techie	Interest in technology and gadgets	1.9 (strong), 0.3 (low)
mobile_user	Preference for short or mobile-friendly articles	1.5 (high), 0.2 (low)
morning_reader	Reads mostly in the morning	1.8 (frequent), 0.4 (rare)
age_z	Standardized age $(\text{age} - 40)/15$	-1.0 (younger $\approx$25y), 0.0 (mid 40s), $+1.0$ (older $\approx$55y)

Table 2: Articles (arms) considered in the simulation ($k = 4$).

Article (Arm)	Main Themes	Appeals to Users Who...
Politics	Government, current affairs	like politics, older, morning readers
Sports	Match highlights, player news	are sports fans and mobile users
Tech	Gadgets, innovation, AI	are tech-oriented, younger readers
Lifestyle	Health, travel, entertainment	prefer general or mobile-friendly articles

Example of a User Context

To illustrate, consider the following user:

$$x_t = [1.6,\ 0.3,\ 0.2,\ 0.5,\ 1.7,\ 0.8],$$

which corresponds to a person who is politically interested, reads in the morning, and is slightly older. The true (hidden) expected click probabilities for each article might look as follows:

Table 3: Example of true click probabilities for a given user context.

Article	True click probability	Comment
Politics	0.86	Strongly aligned with interests (best arm)
Sports	0.24	Weak interest in sports
Tech	0.19	Low tech orientation
Lifestyle	0.48	Moderate interest due to mobile preference

The goal of the learning algorithms is to *discover* these relationships from data without ever seeing the true probabilities.

Algorithms

Two contextual bandit algorithms are studied in this part:

1. Contextual Epsilon-Greedy The Contextual Epsilon-Greedy algorithm extends the simple ϵ-greedy idea to the contextual setting. Each arm a has its own linear model $\hat{\theta}_a$, and the predicted reward for user x_t is

$$\hat{r}_a(x_t) = \hat{\theta}_a^\top x_t.$$

At each step,

$$a_t = \begin{cases} \text{random arm}, & \text{with probability } \epsilon, \\ \arg\max_a \hat{\theta}_a^\top x_t, & \text{with probability } 1-\epsilon. \end{cases}$$

The model parameters $\hat{\theta}_a$ are updated online whenever arm a is chosen, using the observed reward r_t. The exploration rate ϵ determines how often random exploration occurs.

2. Linear UCB (LinUCB) LinUCB models the expected reward as a linear function of the context:

$$\mathbb{E}[r_t | x_t, a] = \theta_a^\top x_t.$$

For each arm a, the algorithm maintains

$$A_a = \lambda I + \sum_{s:a_s=a} x_s x_s^\top, \quad b_a = \sum_{s:a_s=a} r_s x_s,$$

and computes

$$\hat{\theta}_a = A_a^{-1} b_a.$$

At each round, the algorithm chooses the arm that maximizes an upper confidence bound on the predicted reward:

$$a_t = \arg\max_a \left(\hat{\theta}_a^\top x_t + \alpha \sqrt{x_t^\top A_a^{-1} x_t} \right),$$

where α controls the degree of exploration and λ is a regularization parameter that ensures stability.

Running the Experiment

Run the provided notebook code for this part. The environment simulates a news recommendation scenario where user contexts vary in preferences and age. You will:

- Observe short illustrative runs that print user features, chosen articles, and click outcomes.
- Compare the behavior of the Random, Contextual Epsilon-Greedy, and LinUCB algorithms before and after learning from many samples.
- Examine how average reward and cumulative regret evolve over time.

Be Prepared to Explain

Be prepared to explain:

- How the context affects the choice of article.
- How the two algorithms learn over time.
- How Contextual Epsilon-Greedy and LinUCB use the context differently.
- The role of ϵ, α, and λ in balancing exploration and exploitation.

- Why LinUCB typically learns faster and explores more systematically than random or ϵ-greedy exploration.
- What the average reward and regret plots reveal about algorithm performance.
- Both Epsilon-Greedy and LinUCB make modeling assumptions regarding the linearity of the reward, how accurate is that? Can we do better?

Interpretation Tips

In the shorter illustrative runs, focus on the printed details of each decision:

- Early on, the algorithms make nearly random choices since little has been learned.
- After training on many samples, Contextual Epsilon-Greedy tends to choose more relevant articles for each user type, while LinUCB learns to focus on arms with high estimated reward but also high uncertainty.
- Compare how often each method selects the truly best arm and how their reasoning (scores or UCB values) becomes more consistent with the true click probabilities.