Lab 3: Contextual Bandit-Based News Article Recommendation System

Reinforcement Learning Fundamentals

Total: 105 Points February 6, 2026

1 Introduction

Contextual Multi-Armed Bandits (CMAB) extend the classic Multi-Armed Bandit (MAB) framework by introducing "side information" or *contexts* to the decision-making process. While a standard MAB agent operates in a stateless environment where rewards depend solely on the chosen action (*Arm*), a CMAB agent observes a specific *context* (e.g., user demographics or time of day) before choosing an action. In this setting, the goal shifts from finding the single best arm overall to learning a policy that selects the optimal arm conditioned on the current context, effectively mapping specific situations to the actions that yield the highest expected reward.

2 Objective

The primary objective of this assignment is to design and implement a **News Recommendation System** utilizing a **Contextual Bandit** framework. The system will train a Reinforcement Learning (RL) model to recommend news articles by treating user categories as *contexts* and news categories as arms.

Given a new user, the system must predict the most suitable news category and sample a specific article to maximize user engagement (*reward*).

3 Environment & Dataset Specifications

3.1 Datasets

You are provided with two primary datasets:

- **News Articles Dataset** (`news_articles.csv`): Each row contains a **news article** with various **features** and includes a label column **category** specifying the news category (represents the *Arms* of the bandit).
- **User Data** (`train_users.csv`, `test_users.csv`): Each row represents an individual **user** with various **features**, including a **label** column classifying the user into `User1`, `User2`, or `User3` (representing the *Contexts*).

3.2 Reward Distribution Guide (Sampler Utility)

To simulate the environment's response (rewards), you are provided with a Python package named `rlcmab-sampler`. This package contains a Python class `sampler` used to fetch rewards from unknown probability distributions. Students must use the provided package as-is. Any modification or reimplementation of the sampler will result in zero credit for the bandit component.

Installation

Install the package using `pip`:

```
pip install rlcmab-sampler
```

Usage

The `sampler` is initialized with your student Roll Number (i) and queried using an arm index (j).

```
from rlcmab_sampler import sampler

# Initialize with your roll number (i)
reward_sampler = sampler(i)

# Call the function to get a reward from arm j
reward = reward_sampler.sample(j)
```

Student ID Mapping

ID Number	Roll Number (i)
U202300078	78
U202300115	115
U202300001	1

4 Problem Formulation

The environment is modeled as a **Contextual Bandit** problem with the following structure:

- **Contexts:** 3 unique user types (`User1`, `User2`, `User3`).
- **Bandits:** 4 distinct news categories per context.
- **Total Arms:** 3 contexts $\times$ 4 categories = 12 arms.

The arm index j passed to the 'sample(j)' function must map to the specific combination of User Context and News Category as defined below:

j **Values**	**Configuration (News Category, User Context)**
$\{0, 1, 2, 3\}$	{(Entertainment, User1), (Education, User1), (Tech, User1), (Crime, User1)}
$\{4, 5, 6, 7\}$	{(Entertainment, User2), (Education, User2), (Tech, User2), (Crime, User2)}
$\{8, 9, 10, 11\}$	{(Entertainment, User3), (Education, User3), (Tech, User3), (Crime, User3)}

Table 1: Arm Index (j) Mapping

5 Implementation Tasks

5.1 Data Pre-processing (10 Points)

1. Load the provided user and article datasets.
2. Perform necessary data cleaning (e.g., handling missing values).
3. Apply **feature encoding** where required to prepare the data for classification and bandit training.

5.2 User Classification (10 Points)

Develop a classification model (e.g., Decision Tree, Logistic Regression) to predict the user category (`User1`, `User2`, or `User3`) based on input feature data.

- The model must be trained on the provided dataset `train_users.csv`.
- This classifier will serve as the "Context Detector" for your bandit system.

5.3 Contextual Bandit Algorithms (45 Points)

You must implement three distinct strategies. For each strategy, treat the **User Category** as the *context* and **News Category** as the *Arm*.

5.3.1 Epsilon-Greedy (15 Points)

- Train a separate model for each of the 3 user contexts.
- Compute the **Expected Reward Distribution** for each news category across all contexts.
- **Hyperparameter Tuning:** Experiment with multiple values of ϵ. Compare the expected payoffs for different ϵ values.

5.3.2 Upper Confidence Bound (UCB) (15 Points)

- Train a separate model for each of the 3 user contexts.
- Compute the **Expected Reward Distribution**.
- **Hyperparameter Tuning:** Experiment with multiple values of the exploration parameter C. Compare expected payoffs.

5.3.3 SoftMax (15 Points)

- Train a separate model for each of the 3 user contexts.
- Use a fixed temperature parameter $\tau = 1$.
- Compute the **Expected Reward Distribution**.

5.4 Recommendation Engine (20 Points)

Consolidate the classification and decision-making components to establish the end-to-end operational workflow for the CMAB recommendation engine:

1. **Classify:** Determine the User Category using the model from 5.2.
2. **Select Category:** Use the trained Bandit Policies from 5.3 to select the optimal `News Category`.
3. **Recommend:** Randomly sample an article from the selected category in `news_articles.csv`.
4. **Output:** Return the article details.

5.5 Evaluation & Reporting (20 Points)

1. **Classification Accuracy:** Evaluate accuracy of your classifier on `test_users.csv`.
2. **RL Simulation:** Run the RL models for a time horizon of $T = 10,000$ steps.
3. **Analysis Plots:**
   - Plot **Average Reward vs. Time** for each context.
   - Plot **Average Reward comparison** for different hyperparameters (ϵ and C). Test at least 3 distinct values for each.
4. **Final Report:** Compile a comprehensive analysis of the three models, observations on hyperparameter sensitivity, and comparative performance.

6 Submission Guidelines

- **Repository:** Students must `fork` this repository into their own GitHub account.
- **Branching:** All work must be completed in the forked repository and pushed to a branch named `firstname_U20230xxx`. **Do not push to the `master` branch. Submissions on `main` or `master` will be ignored.**
- **Contents:**
  - An IPython Notebook (`.ipynb`) file placed at the **root of the repository**, named exactly `lab3_results_<roll_number>.ipynb`.
  - A `README.md` serving as the project report.
  - Each plot must include labeled axes, a legend, and a descriptive title. Unlabeled or unclear plots may receive partial or no credit.
- **README Requirements:**
  - Summary of results and discussion.
  - You may refer to online resources (e.g., makeareadme.com) on creating effective README files.