Homework 2

6501 Reinforcement Learning (Spring 2025)

Submission deadline: 11:59pm, February 26

Contextual Bandit Algorithms

In this homework, we will implement bandit algorithms we have covered in the lecture. Specifically, we will implement

- Algorithms based on regression oracle and exploration mechanisms including ϵ-greedy (EG), Boltzmann exploration (BE), and inverse gap weighting (IGW), and
- Proximal Policy Optimization (PPO).

The starter code can be accessed at `http://bahh723.github.io/rl2025sp_files/HW2.py`.
The latex template can be accessed at `https://www.overleaf.com/read/bktngfjrszmx#3e0687`
Submit the answer as .pdf and the code as a .py file to Gradescope.

1 Data

To simulate a contextual bandit environment, we use existing classification dataset for supervised learning. Specifically, in this homework, we use the mnist dataset with some pruning and modification. Originally, it is a classification dataset where features are 28 pixel $\times$ 28 pixel gray-scale images of digits, and the classes correspond to 10 digits.

2 Data Conversion

For simplicity, we trim the dataset so that it only contains 4 classes corresponding to digits '0', '1', '2', and '3', each having 2500 samples, summing up to 10000 samples. In each round, the environment will reveal the image (context), and the learner has to pick one of the classes (action). We artificially make the reward function change once in the middle. Specifically, in our case the time horizon is $T = 10000$. For $t \leq 5000$ (Phase 1), the reward function is the following:

$$R(x, a) = \begin{cases} 0.5 & \text{if } a \text{ is the correct digit of image } x \\ 0 & \text{otherwise} \end{cases}$$

For $t > 5000$ (Phase 2), the reward is the following:

$$R(x, a) = \begin{cases} 0.5 & \text{if } a \text{ is the correct digit of image } x \\ 1 & \text{if } (a-1) \bmod 4 \text{ is the is the correct digit of image } x \\ 0 & \text{otherwise} \end{cases}$$

For example, in Phase 2, if the learner chooses action 3 when seeing an image of digit 2, then it receives a reward of 1. We assume that the learner sees the true noiseless reward $R(x_t, a_t)$.

This conversion is already implemented in the starter code, so you don't need to do anything here.

3 Algorithms based on regression oracle

In this part, we will implement the simple value-based algorithm based on regression on the reward function. In the class, we have introduce three common exploration mechanism in this case: ϵ-Greedy, Boltzmann Exploration, and Inverse Gap Weighting. They share the same pseudo-code outlined in Algorithm 1.

Algorithm 1 Algorithms based on regression

Randomly initialize a reward network R_θ that takes the context as input and outputs the reward of each action.
Let θ_1 be the initial weights for the reward network.
for $t = 1, \ldots, T$ **do**
  for $n = 1, \ldots, N$ **do**
    Receive context $x_{t,n}$.
    Sample action $a_{t,n} \sim \pi(\cdot \mid x_{t,n})$ where $\pi(\cdot \mid x) = f\left(R_{\theta_t}(x, \cdot)\right)$
    Receive reward $r_{t,n}$.
  $\theta \leftarrow \theta_t$
  for $m = 1, \ldots, M$ **do**

$$\theta \leftarrow \theta - \lambda \nabla_\theta \left(\frac{1}{N} \sum_{n=1}^{N} \left(R_\theta(x_{t,n}, a_{t,n}) - r_{t,n} \right)^2 \right). \tag{1}$$

  $\theta_{t+1} \leftarrow \theta$

In Algorithm 1, $f : \mathbb{R}^A \to \Delta_A$ is a link function that maps an A-dimensional real vector to a distribution over A actions. The three exploration mechanisms correspond to the following three choices of f:

- ϵ-Greedy: $[f(v)]_a = \frac{\epsilon}{A} + (1 - \epsilon)\mathbb{I}\{a = \operatorname{argmax}_{a'} v(a')\}$.
- Boltzmann Exploration: $[f(v)]_a = \frac{e^{\lambda v(a)}}{\sum_{a'} e^{\lambda v(a')}}$.
- Inverse Gap Weighting: $[f(v)]_a = \begin{cases} \frac{1}{A + \lambda(\max_{a'} v(a') - v(a))} & \text{if } a \neq \operatorname{argmax}_{a'} v(a') \\ 1 - \sum_{a' \neq a} [f(v)]_{a'} & \text{if } a = \operatorname{argmax}_{a'} v(a') \end{cases}$.

Algorithm 1 may be slightly different from the standard way of implementing such algorithm — usually, in Eq. (1) we may reuse data collected in the past by having a *replay buffer* that stores old data. But since our artificial problem has a non-stationary reward function, reusing data from the past becomes not a great idea.

Please complete the following tasks in (a)-(g). Note that the starter code already calculate the average reward in Phase 1, Phase 2, and overall horizon, respectively. The starter code also already implement the regression oracle (1). A figure of running average is also generated for a single parameter. The major task is to code up the three link functions above, run the experiments, and combine the figures for multiple parameters.

Note that you are allowed to change the default hyperparameters in the starter code (N, M, optimizer, learning rates, etc.). In the table below, you may also change the values of hyperparameters or add additional ones if you feel that the given values cannot reflect the trend.

(a) (5%) Implement ϵ-Greedy and, for different values of ϵ, record in the table below the average reward in Phase 1, Phase 2, and over the entire horizon.

ϵ	Phase 1	Phase 2	Overall
0.1			
0.03			
0.01			
0.003			
0.001			
0			

(b) (5%) Plot the running average reward over time for every parameter setting from part (a) on the same figure, following the example provided in the appendix.

(c) (5%) Implement Boltzmann Exploration and, for different values of λ, record in the table below the average reward in Phase 1, Phase 2, and over the entire horizon.

λ	Phase 1	Phase 2	Overall
2			
5			
10			
20			
50			

(d) (5%) Plot the running average reward over time for every parameter setting from part (c) on the same figure.

(e) (5%) Implement Inverse Gap Weighting and, for different values of λ, record in the table below the average reward in Phase 1, Phase 2, and over the entire horizon.

λ	Phase 1	Phase 2	Overall
10			
30			
100			
300			
1000			

(f) (5%) Plot the running average reward over time for every parameter setting from part (e) on the same figure.

(g) (5%) Have you noticed any trends in the experiments from (a)–(f)? In particular, how does the parameter influence the average reward in Phase 1 and Phase 2, respectively? What is the underlying explanation for this effect?

4 PPO

Algorithm 2 PPO without clipping

Randomly initialize a policy network π_θ that takes contexts as input and outputs an action distribution.
Let θ_1 be the initial weights for the policy network.
If using adaptive baseline, additionally initialize a baseline network b_ϕ.
You may choose $\frac{1}{\eta} = 0.1$.
for $t = 1, \ldots, T$ **do**
  for $n = 1, \ldots, N$ **do**
    Receive context $x_{t,n}$.
    Sample action $a_{t,n} \sim \pi_{\theta_t}(\cdot|x_{t,n})$
    Receive reward $r_{t,n}$.
  $\theta \leftarrow \theta_t$
  for $m = 1, \ldots, M$ **do**

$$\theta \leftarrow \theta + \lambda \nabla_\theta \left\{ \frac{1}{N} \sum_{n=1}^{N} \left[\frac{\pi_\theta(a_{t,n}|x_{t,n})}{\pi_{\theta_t}(a_{t,n}|x_{t,n})} (r_{t,n} - b_{t,n}) - \frac{1}{\eta} \left(\frac{\pi_\theta(a_{t,n}|x_{t,n})}{\pi_{\theta_t}(a_{t,n}|x_{t,n})} - 1 - \log \frac{\pi_\theta(a_{t,n}|x_{t,n})}{\pi_{\theta_t}(a_{t,n}|x_{t,n})} \right) \right] \right\}, \tag{2}$$

    where

$$b_{t,n} = \begin{cases} b & \text{for fixed baseline} \\ b_\phi(x_{t,n}) + b' & \text{for adaptive baseline} \end{cases} \tag{3}$$

    If using adaptive baseline, update

$$\phi \leftarrow \phi - \lambda' \nabla_\phi \left[\frac{1}{N} \sum_{n=1}^{N} \left(b_\phi(x_{t,n}) - r_{t,n} \right)^2 \right]. \tag{4}$$

  $\theta_{t+1} \leftarrow \theta$

4.1 Fixed Baseline

(a) (5%) Implement Algorithm 2 with fixed baseline (so Eq. (4) can be omitted) and, for different values of b, record in the table below the average reward in Phase 1, Phase 2, and over the entire horizon.

b	Phase 1	Phase 2	Overall
2			
1.5			
1			
0.5			
0			
-0.5			

(b) (5%) Plot the running average reward over time for every parameter setting from part (a) on the same figure.

(c) (5%) Have you noticed any trends in the experiments from (a)–(b)? How does the baseline affect the average reward in Phase 1 and Phase 2, respectively? What is the underlying explanation for this effect?

4.2 Adaptive Baseline

(a) (5%) Implement Algorithm 2 with adaptive baseline and, for different values of the extra baseline b' in (3), record in the table below the average reward in Phase 1, Phase 2, and over the entire horizon.

b'	Phase 1	Phase 2	Overall
0.2			
0.1			
0			
-0.1			
-0.2			

(b) (5%) Plot the running average reward over time for every parameter setting from part (a) on the same figure.

(c) (5%) Is there any difference between adaptive baseline and fixed baseline in Section 4.1? What are the potential advantages or disadvantages of using adaptive baseline?

(d) (5%) What is the effect of the extra baseline parameter b'?

5 Survey

(a) (5%) How much time did you use to complete each of the problems in this homework? Do you have any suggestion for the course? (e.g., the pace of the lecture, the length of the homework)

A Appendix

In the starter code, we plot the learning curve for a single parameter. You need to plot the learning curves for multiple parameter values on the same figure. Below is an example that shows how to plot two curves, although the problems above require you to plot 5-6 curves in the same figure.