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`.

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 $t \le 5000$ (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

```
1  Randomly initialize a reward network $R_\theta$ that takes the context as input and outputs the reward of each action.
2  Let $\theta_1$ be the initial weights for the reward network.
3  for $t = 1, \dots, T$ do
4      for $n = 1, \dots, N$ do
5          Receive context $x_{t,n}$.
6          Sample action $a_{t,n} \sim \pi(\cdot \mid x_{t,n})$ where $\pi(\cdot \mid x) = f\left(R_{\theta_t}(x, \cdot)\right)$
7          Receive reward $r_{t,n}$.
8      $\theta \leftarrow \theta_t$
9      for $m = 1, \dots, M$ do
           $$\theta \leftarrow \theta - \lambda \nabla_\theta \left( \frac{1}{N} \sum_{n=1}^{N} \left(R_\theta(x_{t,n}, a_{t,n}) - r_{t,n}\right)^2 \right). \quad (1)$$
10     $\theta_{t+1} \leftarrow \theta$
```

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.

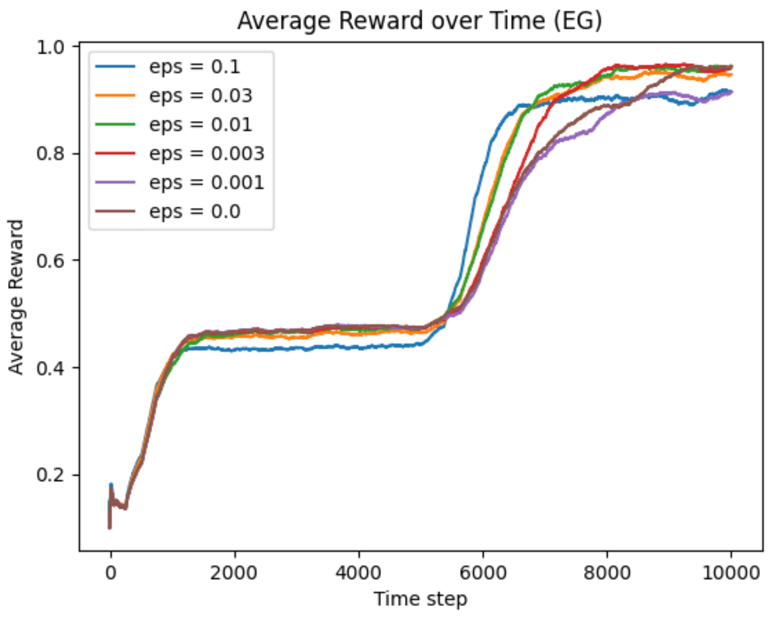
Please complete the following tasks in (a)-(g). Note that the starter code already calculate the average reward in Phase 1, Phase 2, and overall horizon, respectively. The starter code also already implement the regression oracle (1). A figure of running average is also generated for a single parameter. 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	0.4140	0.8480	0.6310
0.03	0.4327	0.8549	0.6438
0.01	0.4350	0.8615	0.6483
0.003	0.4382	0.8424	0.6403
0.001	0.4401	0.7917	0.6159
0	0.4401	0.8147	0.6274

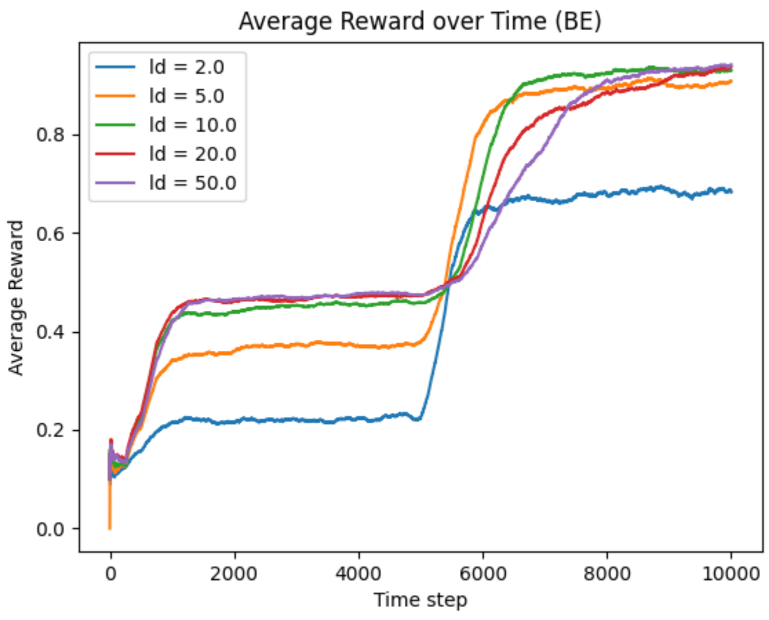
(b) (5%) Plot the running average reward over time in the same figure for your experiments in (a). See appendix for an example of such a figure.



(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	0.2135	0.6588	0.4362
5	0.3489	0.8547	0.6018
10	0.4244	0.8565	0.6405
20	0.4406	0.8145	0.6276
50	0.4392	0.8040	0.6216

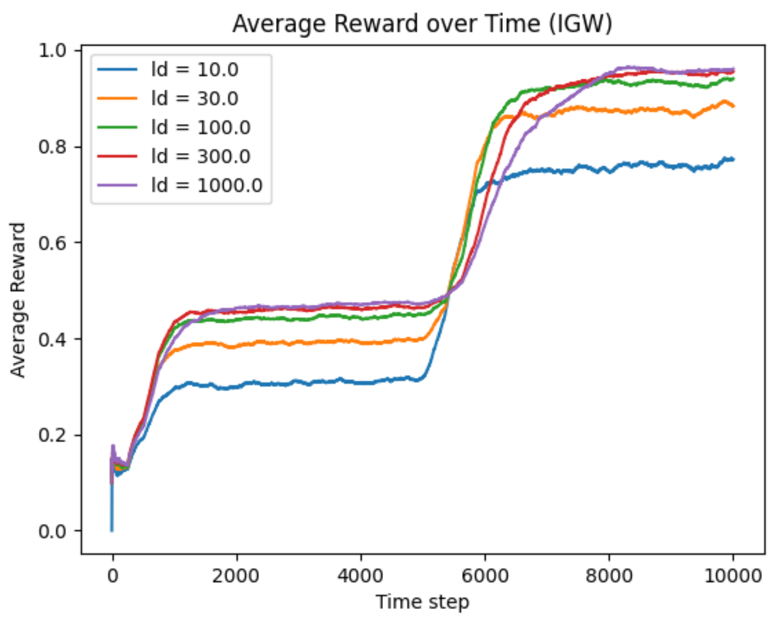
(d) (5%) Plot the running average reward over time in the same figure for your experiments in (c).



(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	0.2945	0.7304	0.5125
30	0.3726	0.8327	0.6027
100	0.4185	0.8717	0.6451
300	0.4346	0.8640	0.6493
1000	0.4339	0.8488	0.6414

(f) (5%) Plot the running average reward over time in the same figure for your experiments in (e).



(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?

Answer When decreasing ϵ in ϵ-Greedy, or increasing λ in Boltzmann exploration or inverse gap weighting, there will be *decreasing* amount of exploration and *increasing* amount of exploitation. In Phase 1, the more the exploration, the worse the average performance. This is because the environment is so simple and thus explicit exploration is almost not needed. However, higher exploration leads to *quicker discovery* of the change point around $t = 5000$ (observe the timing when the curves rise around $t = 5000$). Like in Phase 1, higher exploration leads to worse performance at the end of Phase 2. Therefore, there is a tradeoff between exploitation and exploration in Phase 2, and the best performed parameter is usually not the extreme ones.

4 PPO

Algorithm 2 PPO without clipping

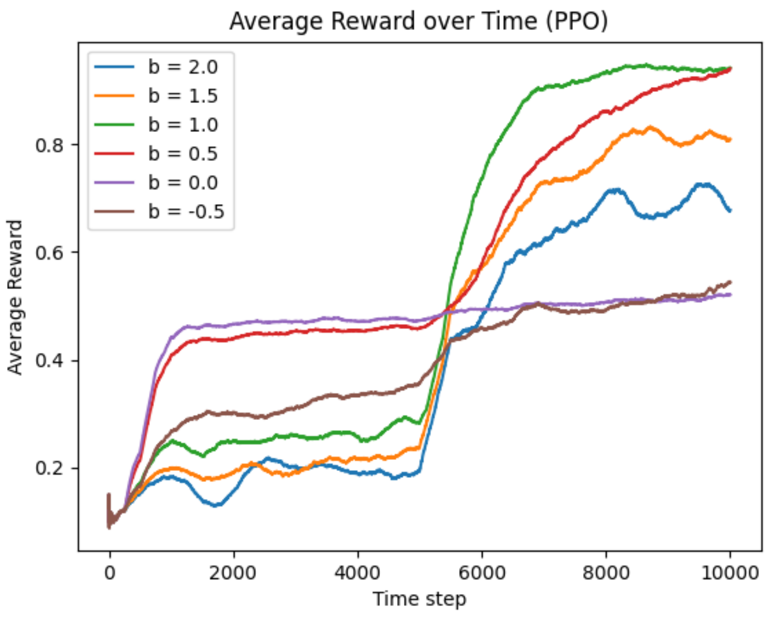
```
11 Randomly initialize a policy network $\pi_\theta$ that takes contexts as input and outputs an action distribution.
12 Let $\theta_1$ be the initial weights for the policy network.
13 If using adaptive baseline, additionally initialize a baseline network $b_\phi$.
14 for $t = 1, \dots, T$ do
15     for $n = 1, \dots, N$ do
16         Receive context $x_{t,n}$.
17         Sample action $a_{t,n} \sim \pi_{\theta_t}(\cdot|x_{t,n})$
18         Receive reward $r_{t,n}$.
19     $\theta \leftarrow \theta_t$
20     for $m = 1, \dots, M$ do
           $$\theta \leftarrow \theta + \lambda \nabla_\theta \left\{ \frac{1}{N} \sum_{n=1}^{N} \left[ \frac{\pi_\theta(a_{t,n}|x_{t,n})}{\pi_{\theta_t}(a_{t,n}|x_{t,n})} (r_{t,n} - b_{t,n}) - \frac{1}{\eta} \left( \frac{\pi_\theta(a_{t,n}|x_{t,n})}{\pi_{\theta_t}(a_{t,n}|x_{t,n})} - 1 - \log \frac{\pi_\theta(a_{t,n}|x_{t,n})}{\pi_{\theta_t}(a_{t,n}|x_{t,n})} \right) \right] \right\}, \quad (2)$$
           where
           $$b_{t,n} = \begin{cases} b & \text{for fixed baseline} \\ b_\phi(x_{t,n}) + b' & \text{for adaptive baseline} \end{cases} \quad (3)$$
           If using adaptive baseline, update
           $$\phi \leftarrow \phi - \lambda' \nabla_\phi \left[ \frac{1}{N} \sum_{n=1}^{N} \left( b_\phi(x_{t,n}) - r_{t,n} \right)^2 \right]. \quad (4)$$
21     $\theta_{t+1} \leftarrow \theta$
```

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	0.1817	0.6228	0.4022
1.5	0.2001	0.7221	0.4611
1	0.2471	0.8636	0.5554
0.5	0.4226	0.7889	0.6057
0	0.4432	0.5046	0.4739
-0.5	0.2997	0.4937	0.3967

(b) (5%) Plot the running average reward over time in the same figure for your experiments in (a).



(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?

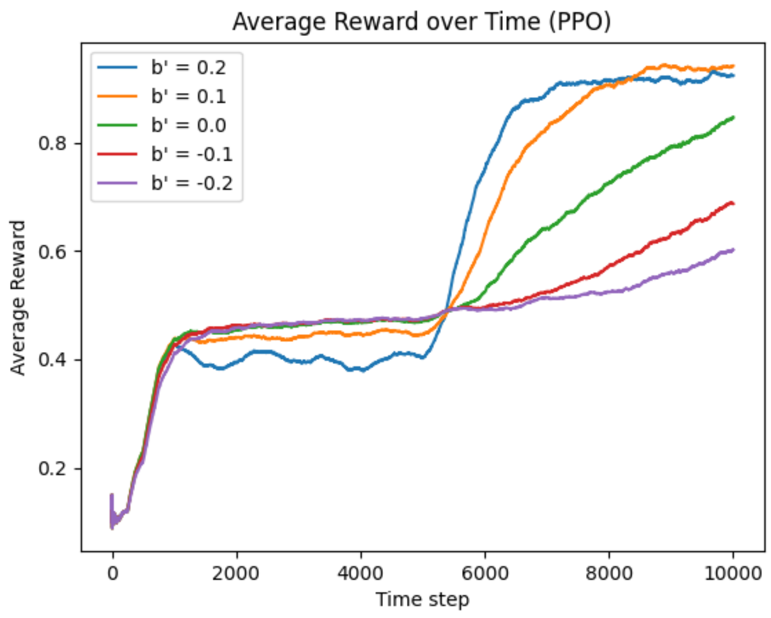
Answer The reasoning is quite similar to Question 3(g), just that now higher baseline leads to more exploration. But unlike in EG/BE/IGW where pure exploitation leads to satisfying performance, here, when we set b to be very low (-0.5), the performance is quite bad in both phases. This is probably because setting $b = -0.5$ has a very bad effect: after choosing an action, no matter what reward we get, the algorithm will tend to increase the probability of choosing that action in the next round. Setting $b = 0$ or $b = -0.5$ will make the algorithm totally unable to discover the change point at $t = 5000$.

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	0.3837	0.8557	0.6197
0.1	0.4212	0.8241	0.6226
0	0.4370	0.6867	0.5619
-0.1	0.4380	0.5708	0.5044
-0.2	0.4344	0.5308	0.4826

(b) (5%) Plot the running average reward over time in the same figure for your experiments in (a).



(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?

Answer The best-performed curve under adaptive baseline seems to be close to the combination of the best-performed ones in Phase 1 and Phase 2 under fixed baselines. The adaptive baseline is able to track the average performance of the algorithm itself and tune baseline on it.

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

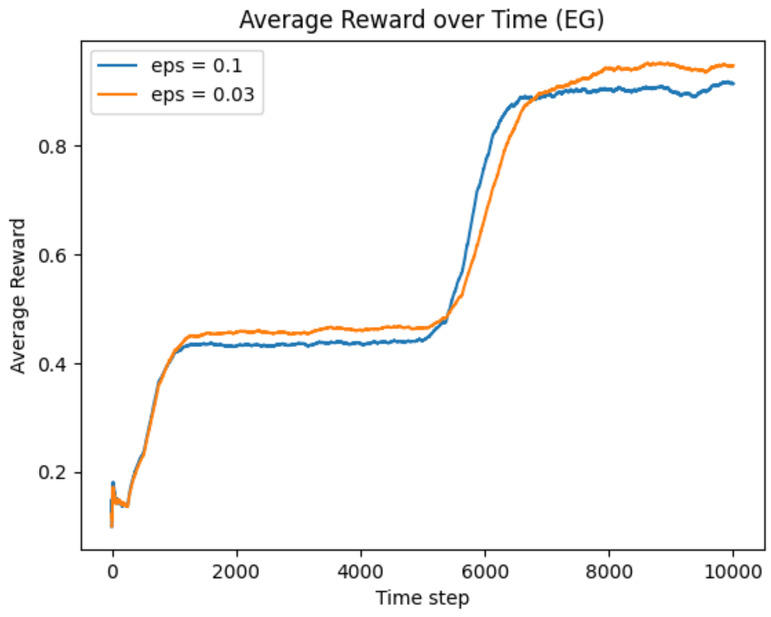
Answer Intuitively, the learner tends to increase the probability of a particular action if its reward is higher than the baseline, and decrease if it is lower than the baseline. With a positive extra baseline b', the learner will more aggressively seek actions that can improve over its current performance. Therefore, a positive extra baseline allows the algorithm to more quickly converge to the new optimal action after the change point.

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 in usually need to plot 5-6 curves in the problems above.


Average Reward over Time (EG)
eps = 0.1
eps = 0.03
Average Reward
Time step