Homework 1

4771 Reinforcement Learning (Spring 2026)

Deadline: 11:59pm, February 4, 2026

Instructions:

- Submit one .pdf file and one .py file (not .ipynb) to Gradescope. The .pdf should contain your answers to the questions, and the .py file should contain your code.
- You may use LaTeX or another editor (e.g., Microsoft Word) to generate the .pdf, as long as it includes the required information. You may also handwrite your answers and scan them as a .pdf. A LaTeX template is available at https://www.overleaf.com/read/xxqcmjgptysq#aa4bb6.

1 Contextual Bandits with Regression

In this homework, we will implement two exploration strategies for value-based contextual bandits, including ϵ-greedy (EG) and Boltzmann exploration (BE). The starter code can be accessed at http://bahh723.github.io/rl2026sp_files/bandits.py.

The homework relies on coding with PyTorch. If you have not used it before, a nice tutorial is in https://www.youtube.com/watch?v=c36lUUr864M. It is recommended to watch it at least up to Tutorial #07 (Linear Regression) to understand everything in the starter code.

1.1 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.

1.2 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 x_t (context), and the learner has to pick one of the classes a_t (action). The learner interacts with the environment for $T = 10000$ rounds. We artificially make the reward function switch once in the middle: 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 a noisy reward $r_t = R(x_t, a_t) + w_t$, where $w_t \sim \mathcal{N}(0, 0.5^2)$. That is, r_t follows normal distribution with mean $= R(x_t, a_t)$ and standard deviation $= 0.5$.

This conversion is already implemented in the starter code, so you don't need to code up anything here. However, you should read the code and verify that the code is indeed implementing the above.

1.3 Algorithm

In this part, we will implement the simple value-based algorithm based on regression on the reward function. In the class, we introduced two most common exploration mechanisms: ϵ-Greedy and Boltzmann Exploration. They share the same pseudo-code outlined in Algorithm 1.

Algorithm 1 Value-based contextual bandit algorithm 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 $\pi(\cdot \mid x) = f(R_{\theta_t}(x, \cdot))$
7          Receive reward $r_{t,n}$.
8      $\theta \leftarrow \theta_t$
9      for $m = 1, \dots, M$ do
           $$\theta \leftarrow \theta - \lambda \nabla_\theta \left( \frac{1}{N} \sum_{n=1}^{N} (R_\theta(x_{t,n}, a_{t,n}) - r_{t,n})^2 \right). \quad (1)$$
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 two exploration mechanisms correspond to the following two choices of f:

- ϵ-Greedy: $[f(v)]_a = \frac{\epsilon}{A} + (1 - \epsilon)\mathbb{I}\{a = \operatorname{argmax}_{a'} v(a')\}$.
- Boltzmann Exploration: $[f(v)]_a = \frac{e^{\lambda v(a)}}{\sum_{a'} e^{\lambda v(a')}}$.

Complete the tasks and answer the questions in (a)-(f) below. Note that the starter code already calculates the average reward in Phase 1, Phase 2, overall reward, and prints them after execution. The starter code also already implements the regression procedure (1). A figure of running average will be generated. The only TODO is to code up the two link functions above (marked with `TODO` in the code).

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

The starter code already implements two baseline methods: one uniformly randomly chooses an action, and the other greedily chooses an action with the highest estimated reward. Run them with the command line:

```
python bandits.py --algorithm Rand
python bandits.py --algorithm Greedy
```

The code will run five times with different random seeds and average up the performance. For each method you implement, a correct implementation must achieve an "overall reward" of at least 0.55 under the *best choice* of the hyperparameter. Meeting this requirement is necessary to get full credit.

(a) (5%) Implement the ϵ-Greedy decision rule (the first `TODO` in the code) and, for different values of ϵ (specify the list of ϵ in `eps_list`), record in the table below the average reward in Phase 1, Phase 2, and over the entire horizon.

ϵ	Phase 1	Phase 2	Overall
0.3			
0.1			
0.03			
0.01			
0.003			
0.001			
0			

(b) (5%) Paste the running average reward plot generated by the code for all parameters in (a), following the example provided in the appendix.

(c) (5%) Implement Boltzmann Exploration decision rule (the second `TODO` in the code) and, for different values of λ (specify the list of λ in `lam_list`), 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%) Paste the running average reward plot generated by the code for all parameters in (c), following the example provided in the appendix.

(e) (5%) How does the performance in **Phase 1** change as ϵ decreases in ϵ-Greedy, or as λ increases in Boltzmann Exploration? Why?

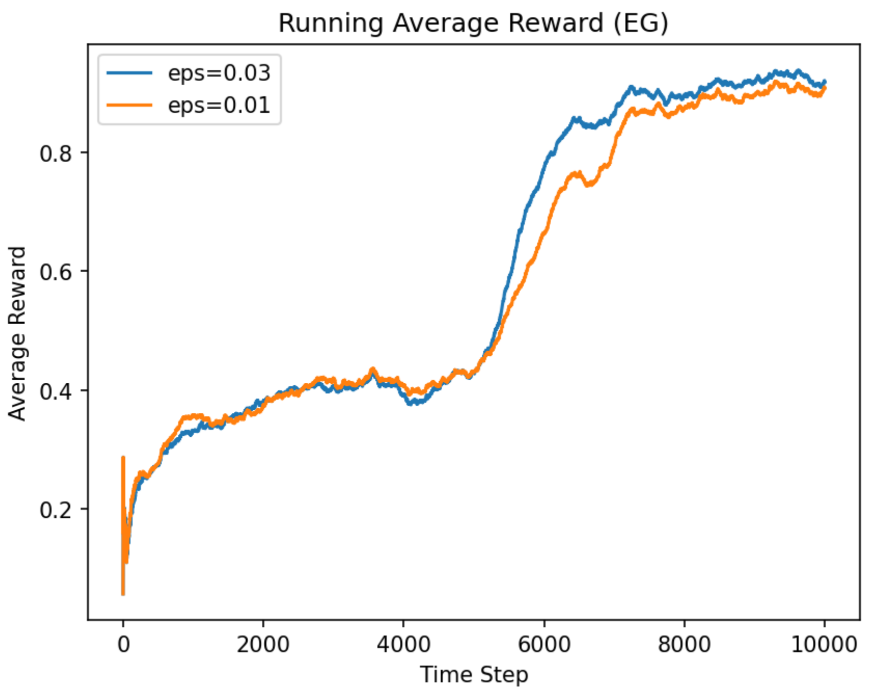
(f) (5%) How does the performance in **Phase 2** change as ϵ decreases in ϵ-Greedy, or as λ increases in Boltzmann Exploration? Why?

2 Survey

(5%) How much time did you use to complete this homework? How difficult is this homework to you? Do you have any suggestion for the course?

Appendix

The code will generate the learning curves for a list of parameters. You need the specify the list of parameters in `eps_list` and `lam_list` in the code. Below is an example of the figure that will be generated when running ϵ-Greedy with `eps_list=[0.1, 0.03]`. In (b) above, this list should include all values specified in the table in (a), and similar for `lam_list`.


Running Average Reward (EG)
eps=0.03
eps=0.01
Average Reward
Time Step