SI252 Reinforcement Learning 2025/04/03

Homework 4

Professor: Ziyu Shao *Due: 2025/04/20 11:59pm*

- **Part I: Analysis of Bandit Algorithms**

  (1) Reproduce the proof for Regret Decomposition Lemma

  (2) Given k actions, we define a preference function $H(\cdot) : \{1, \ldots, k\} \to \mathcal{R}$. Then for actions $x \in \{1, \ldots, k\}$ and $y \in \{1, \ldots, k\}$, we define a soft-max function

  $$\pi(x) = \frac{e^{H(x)}}{\sum_{y=1}^{k} e^{H(y)}}.$$

  Please show the following result: for any action $a \in \{1, \ldots, k\}$, we have

  $$\frac{\partial \pi(x)}{\partial H(a)} = \pi(x) \left(1_{\{x=a\}} - \pi(a) \right),$$

  where 1_A is an index function of events, being 1 when event A is true and being 0 otherwise.

  (3) Reproduce the proof for gradient bandit algorithm.

  (4) (Bonus): show the proof for the regret upper bound of UCB1 algorithm

  (5) (Bonus): show the proof for the regret upper bound of Thompson sampling algorithm (Beta-Bernoulli bandit only)

- **Part II: Performance Evaluation of Classical Bandit Algorithms**
  You are required to use the Jupyter Notebook (Formerly known as the IPython Notebook) to submit your work.

- **Basic Setting**
  We consider a time-slotted bandit system ($t = 0, 1, 2, \ldots$) with three arms. We denote the arm set as $\{1, 2, 3\}$. Pulling each arm j ($j \in \{1, 2, 3\}$) will obtain a reward r_j, which satisfies a Bernoulli distribution with mean θ_j (Bern(θ_j)), *i.e.*,

  $$r_j = \begin{cases} 1, & w.p.\ \theta_j, \\ 0, & w.p.\ 1 - \theta_j, \end{cases}$$

  where θ_j are parameters within $(0, 1)$ for $j \in \{1, 2, 3\}$.

Now we run this bandit system for N ($N \gg 3$) time slots. At each time slot t, we choose one and only one arm from these three arms, which we denote as $I(t) \in \{1, 2, 3\}$. Then we pull the arm $I(t)$ and obtain a reward $r_{I(t)}$. Our objective is to find an optimal policy to choose an arm $I(t)$ at each time slot t such that the expectation of the aggregated reward is maximized, *i.e.*,

$$\max_{I(t),t=1,\ldots,N} \quad \mathbb{E}\left[\sum_{t=1}^{N} r_{I(t)}\right].$$

If we know the values of $\theta_j, j \in \{1, 2, 3\}$, this problem is trivial. Since $r_{I(t)} \sim \mathrm{Bern}(\theta_{I(t)})$,

$$\mathbb{E}\left[\sum_{t=1}^{N} r_{I(t)}\right] = \sum_{t=1}^{N} \mathbb{E}[r_{I(t)}] = \sum_{t=1}^{N} \theta_{I(t)}.$$

Let $I(t) = I^* = \arg\max_{j\in\{1,2,3\}} \ \theta_j$ for $t = 1, 2, \ldots, N$, then

$$\max_{I(t),t=1,\ldots,N} \quad \mathbb{E}\left[\sum_{t=1}^{N} r_{I(t)}\right] = N \cdot \theta_{I^*}.$$

However, in reality, we do not know the values of $\theta_j, j \in \{1, 2, 3\}$. We need to estimate the values θ_j via empirical samples, and then make the decisions at each time slot.

Next we introduce three classical bandit algorithms: $\epsilon-$greedy, UCB and Thompson sampling.

- **Bandit Algorithms**

  1. $\epsilon-$greedy Algorithm ($0 \le \epsilon \le 1$)

Algorithm 1 $\epsilon-$greedy Algorithm

Initialize $\hat{\theta}(j) = 0, \mathrm{count}(j) = 0, j \in \{1, 2, 3\}$

for $t = 1, 2 \ldots, N$ **do**

$$I(t) \leftarrow \begin{cases} \arg\max_{j\in\{1,2,3\}} \ \hat{\theta}(j) & w.p. \ 1-\epsilon \\ \text{randomly chosen from}\{1,2,3\} & w.p. \ \epsilon \end{cases}$$

  $\mathrm{count}(I(t)) \leftarrow \mathrm{count}(I(t)) + 1$

  $\hat{\theta}(I(t)) \leftarrow \hat{\theta}(I(t)) + \frac{1}{\mathrm{count}(I(t))}\left[r_{I(t)} - \hat{\theta}(I(t))\right]$

end for

2. UCB (Upper Confidence Bound) Algorithm

Algorithm 2 UCB Algorithm

for $t = 1, 2, 3$ **do**
  $I(t) \leftarrow t$
  $\text{count}(I(t)) \leftarrow 1$
  $\hat{\theta}(I(t)) \leftarrow r_{I(t)}$
end for
for $t = 4, \dots, N$ **do**

$$I(t) \leftarrow \underset{j \in \{1,2,3\}}{\arg\max} \left(\hat{\theta}(j) + c \cdot \sqrt{\frac{2 \log t}{\text{count}(j)}} \right)$$

  $\text{count}(I(t)) \leftarrow \text{count}(I(t)) + 1$
  $\hat{\theta}(I(t)) \leftarrow \hat{\theta}(I(t)) + \frac{1}{\text{count}(I(t))} \left[r_{I(t)} - \hat{\theta}(I(t)) \right]$
end for

Note: c is a positive constant with a default value of 1.

3. Thompson sampling (TS) Algorithm

Recall that $\theta_j, j \in \{1, 2, 3\}$, are unknown parameters over $(0, 1)$. From the Bayesian perspective, we assume their priors are Beta distributions with given parameters (α_j, β_j).

Algorithm 3 Thompson sampling Algorithm

Initialize Beta parameter $(\alpha_j, \beta_j), j \in \{1, 2, 3\}$
for $t = 1, 2 \dots, N$ **do**
  # Sample model
  for $j \in \{1, 2, 3\}$ **do**
    Sample $\hat{\theta}(j) \sim \text{Beta}(\alpha_j, \beta_j)$
  end for
  # Select and pull the arm

$$I(t) \leftarrow \underset{j \in \{1,2,3\}}{\arg\max} \, \hat{\theta}(j)$$

  # Update the distribution

$$\alpha_{I(t)} \leftarrow \alpha_{I(t)} + r_{I(t)}$$
$$\beta_{I(t)} \leftarrow \beta_{I(t)} + 1 - r_{I(t)}$$

end for

- **Simulation**

(1) Now suppose we obtain the Bernoulli distribution parameters from an oracle, which are shown in the following table below. Choose $N = 10000$ and compute the theoretically maximized expectation of aggregate rewards over N time slots. We call it the oracle value. Note that these parameters $\theta_j, j = 1, 2, 3$ and oracle values are unknown to all bandit algorithms.

Arm j	1	2	3
θ_j	0.9	0.8	0.7

(2) Implement classical bandit algorithms with following settings:

  - $N = 5000$
  - ϵ-greedy with $\epsilon = 0.1, 0.5, 0.9$.
  - UCB with $c = 1, 5, 10$.
  - Thompson Sampling with $\{(\alpha_1, \beta_1) = (1, 1), (\alpha_2, \beta_2) = (1, 1), (\alpha_3, \beta_3) = (1, 1)\}$ and $\{(\alpha_1, \beta_1) = (601, 401), (\alpha_2, \beta_2) = (401, 601), (\alpha_3, \beta_3) = (2, 3)\}$
  - Gradient bandit with baseline $b = 0, 0.8, 5, 20$.
  - Parameterized gradient bandit with constant parameter $\beta = 0.2, 1, 2, 5$
  - Parameterized gradient bandit with time-varying parameters (you need to design a time-varying rule)

(3) Each experiment lasts for $N = 5000$ turns, and we run each experiment 1000 times. Results are averaged over these 1000 independent runs.

(4) Please report three performance metrics

  - The total regret accumulated over the experiment.
  - The regret as a function of time.
  - The percentage of plays in which the optimal arm is pulled.

(5) Compute the gaps between the algorithm outputs and the oracle value. Compare the numerical results of ϵ-greedy, UCB, Thompson Sampling and gradient bandit. Which one is the best? Then discuss the impacts of ϵ, C, and α_j, β_j, b, and β respectively.

(6) What is the role of baseline in gradient bandit algorithm? Show your answer with simulation result.

(7) Give your understanding of the exploration-exploitation trade-off in bandit algorithms.

(8) Give your understanding of the adoption of sublinear regret as the performance threshold between good bandit algorithms and bad bandit algorithms.

- **Part III: Design for Modern Bandit Algorithms**
  Read papers from the selective reading list, then

  (1) design a UCB style algorithm for graph bandits in the format of pseudocode, and explain how you utilize the additional structure information.

  (2) design a UCB style algorithm for dueling bandits in the format of pseudocode, and explain how you utilize the additional structure information.

  (3) design a UCB style algorithm for combinatorial bandits in the format of pseudocode, and explain how you utilize the additional structure information.

  (4) design a UCB style algorithm for neural bandits in the format of pseudocode, and explain how you utilize the additional structure information.