Homework 1

6501 Reinforcement Learning (Spring 2024)

Submission deadline: 11:59pm, February 23

Rules

- Collaboration is allowed but must be stated in precision per sub-problem. For example, comment in the sub-problem or beside the equation which idea is a result of a discussion with whom.
- Leveraging large language models is allowed but must be stated. Describe how you use them or provide precise prompts that you use.
- The submission should be a zip file that contains a single pdf file typed with latex, and a single python file.
- For theoretical problems, rigorous proofs are required.

Notation $\tilde{O}(\cdot)$ denotes $O(\cdot)$ but ignoring logarithmic terms. For example, $\sqrt{T}\log^2 T = \tilde{O}(\sqrt{T})$. For any positive integer N, define $[N] = \{1, 2, \ldots, N\}$. Most of the notations used here follow those in the slides.

Appendix A provides theorems you may use in some problems.

1 $O(\log T)$ regret bound for UCB

In the class, we showed that the regret incurred in UCB is bounded by $\tilde{O}(\sqrt{T})$. In this problem, we will show that the same algorithm in fact ensures a more favorable $O(\log T)$ regret. We first define the following quantities: Let $R(a) \in [0, 1]$ be the true mean of the reward of arm a. Define $R^\star \triangleq \max_{a\in[A]} R(a)$ and $\Delta(a) = R^\star - R(a)$.

We consider the UCB algorithm described in Algorithm 1 (the same as presented in the class). Assume that the number of arms A is less or equal to the number of rounds T, and assume w_t is zero-mean and 1-sub-Gaussian.

Algorithm 1 UCB for multi-armed bandits

Input: A (number of arms), T (total number of rounds), δ (failure probability).
for $t = 1, \ldots, A$ **do**
  Draw $a_t = t$ and observe $r_t = R(a_t) + w_t$.
for $t = A + 1, \ldots, T$ **do**
  Define

$$N_t(a) = \sum_{s=1}^{t-1} \mathbb{I}\{a_s = a\}, \quad \hat{R}_t(a) = \frac{\sum_{s=1}^{t-1} \mathbb{I}\{a_s = a\} r_s}{N_t(a)}, \quad \mathit{conf}_t(a) = \sqrt{\frac{2\log(2/\delta)}{N_t(a)}}, \quad \tilde{R}_t(a) = \hat{R}_t(a) + \mathit{conf}_t(a).$$

  Draw $a_t = \operatorname{argmax}_a \tilde{R}_t(a)$ and observe $r_t = R(a_t) + w_t$.

Recall that the regret is defined as $\text{Regret} = TR^\star - \sum_{t=1}^{T} R(a_t)$.

(a) (3%) Argue that with probability at least $1 - AT\delta$, $|\hat{R}_t(a) - R(a)| \le \mathit{conf}_t(a)$ for all time t and arm a.

(b) (6%) Assume that the inequality in (a) holds for all t and a. Prove that if $N_t(a) > \frac{2\log(2/\delta)}{\Delta(a)^2}$, then $\tilde{R}_t(a) \neq \max_{a' \in [A]} \tilde{R}_t(a')$.

(c) (6%) Prove that with probability at least $1 - AT\delta$, Regret $\leq A + \sum_{a:\ R(a) \neq R^\star} \frac{2\log(2/\delta)}{\Delta(a)}$. Pick a δ and argue that this choice of δ gives $\mathbb{E}[\text{Regret}] = O\left(\sum_{a:\ R(a) \neq R^\star} \frac{\log(T)}{\Delta(a)}\right)$.

2 LinUCB with mis-specified reward

In the class, we provided a proof sketch for the regret bound of LinUCB assuming that the reward is *well-specified*, that is, $R(x,a) = \phi(x,a)^\top \theta^\star$ for some $\theta^\star \in \mathbb{R}^d$, where $R(x,a)$ is the true reward of (x,a), and $\phi(x,a) \in \mathbb{R}^d$ is the feature vector. In this problem, we will redo the analysis, but relaxing this assumption. Suppose $\theta^\star$ is such that $\phi(x,a)^\top \theta^\star$ is a good approximation for the true reward $R(x,a)$. Define the following *mis-specification* function:

$$\epsilon(x,a) = R(x,a) - \phi(x,a)^\top \theta^\star.$$

Consider the LinUCB algorithm Algorithm 3. It is same as the one presented in the lecture, though the β parameter is unspecified here. Later, we will set its value differently from that in the lecture.

Algorithm 2 LinUCB

Given: action set $\mathcal{A}$, feature mapping ϕ.
Parameter: β
for $t = 1, 2, \ldots, T$ **do**
  Receive x_t, and choose action

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} \phi(x_t,a)^\top \hat{\theta}_t + \sqrt{\beta} \|\phi(x_t,a)\|_{\Lambda_t^{-1}},$$

  where $\Lambda_t = I + \sum_{i=1}^{t-1} \phi(x_i,a_i)\phi(x_i,a_i)^\top$ and $\hat{\theta}_t = \Lambda_t^{-1} \sum_{i=1}^{t-1} \phi(x_i,a_i) r_i$.
  Receive $r_t = R(x_t,a_t) + w_t$, for some zero-mean, 1-sub-Gaussian w_t.

Define $\epsilon_t = \epsilon(x_t,a_t)$ and $\phi_t = \phi(x_t,a_t)$. Assume that $\|\phi(x,a)\|_2 \leq 1$ for any x, a, and assume $\|\theta^\star\|_2 \leq \sqrt{d}$.

(a) (5%) Prove that for any $u \in \mathbb{R}^d$,

$$\left|u^\top(\hat{\theta}_t - \theta^\star)\right| \leq \|u\|_{\Lambda_t^{-1}} \left(\left\| \sum_{i=1}^{t-1} \phi_i w_i \right\|_{\Lambda_t^{-1}} + \left\| \sum_{i=1}^{t-1} \phi_i \epsilon_i \right\|_{\Lambda_t^{-1}} + \|\theta^\star\|_{\Lambda_t^{-1}} \right).$$

(b) (5%) Continuing from (a), prove that with probability at least $1 - \delta$, the following hold for all $t \in [T+1]$ and all $u \in \mathbb{R}^d$:

$$\left|u^\top(\hat{\theta}_t - \theta^\star)\right| \leq \|u\|_{\Lambda_t^{-1}} \left(\sqrt{d \log\left(1 + \frac{T}{d}\right) + 2\log(1/\delta)} + \sqrt{d \sum_{i=1}^{t-1} \epsilon_i^2} + \sqrt{d} \right).$$

(c) (5%) Suppose that the learner know in advance that the following condition on the "total amount of mis-specification" always holds:

$$\sum_{t=1}^{T} \max_a \epsilon(x_t,a)^2 \leq \mathcal{E} \tag{1}$$

no matter what choices of a_t's are. What would be a reasonable choice for the value of β? With this choice of β, what regret bound can we obtain for Algorithm 3?

Remark Unfortunately, in order to well deal with mis-specification in LinUCB, one has to set β according to some prior knowledge on the "total amount of mis-specification" $\mathcal{E}$ defined in Eq. (1). If we just use the original β as in the well-specified case, the regret can be much worse (theoretically), as show on Page 12 of Lattimore et al. (2020)

3 ϵ-greedy for general contextual bandits

In the class, we have shown that SquareCB (a.k.a. Inverse Gap Weighting) can reduce contextual bandit to regression, and ensure a $\tilde{O}(\sqrt{T})$ regret. In this problem, we will show that ϵ-Greedy also serves as an easy way to reduce CB to regression, albeit with a slightly worse $\tilde{O}(T^{2/3})$ regret bound.

Algorithm 3 ϵ-Greedy

Parameter: $\epsilon \in [0,1]$, A (number of actions)
Given: A regression procedure
for $t = 1, 2, \ldots, T$ **do**
  Receive x_t, and obtain $\hat{R}_t$ from the regression procedure.
  Define

$$p_t(a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{A} & \text{if } a = \operatorname{argmax}_{a'} \hat{R}_t(x_t, a') \\ \frac{\epsilon}{A} & \text{otherwise} \end{cases}$$

  Sample $a_t \sim p_t$, and receive $r_t = R(x_t, a_t) + w_t$.

Define $a_t^\star = \operatorname{argmax} R(x_t, a)$.

(a) (5%) Show that the one-step expected regret can be upper bounded as follows:

$$R(x_t, a_t^\star) - \mathbb{E}_{a_t \sim p_t}\left[R(x_t, a_t)\right] \leq \epsilon + R(x_t, a_t^\star) - \hat{R}_t(x_t, a_t^\star) + \mathbb{E}_{a_t \sim p_t}\left[\hat{R}_t(x_t, a_t) - R(x_t, a_t)\right].$$

(b) (5%) Argue that for any $\alpha > 0$, the following two inequalities hold:

$$\mathbb{E}_{a_t \sim p_t}\left[\hat{R}_t(x_t, a_t) - R(x_t, a_t)\right] \leq \frac{1}{2\alpha} + \frac{\alpha}{2}\mathbb{E}_{a_t \sim p_t}\left[\left(\hat{R}_t(x_t, a_t) - R(x_t, a_t)\right)^2\right],$$

$$R(x_t, a_t^\star) - \hat{R}_t(x_t, a_t^\star) \leq \frac{1}{2\alpha p_t(a_t^\star)} + \frac{\alpha}{2}\mathbb{E}_{a_t \sim p_t}\left[\left(\hat{R}_t(x_t, a_t) - R(x_t, a_t)\right)^2\right].$$

(c) (5%) Combining (a) and (b), show that the one-step expected regret can be upper bounded as

$$R(x_t, a_t^\star) - \mathbb{E}_{a_t \sim p_t}\left[R(x_t, a_t)\right] \leq \epsilon + \frac{A}{\alpha\epsilon} + \alpha\mathbb{E}_{a_t \sim p_t}\left[\left(\hat{R}_t(x_t, a_t) - R(x_t, a_t)\right)^2\right],$$

for any $\alpha > 0$. Conclude that the expected regret $\mathbb{E}\left[\sum_{t=1}^T \left(R(x_t, a_t^\star) - R(x_t, a_t)\right)\right]$ can be upper bounded by

$$O\left(\left(A\mathbb{E}\left[\sum_{t=1}^T \left(\hat{R}_t(x_t, a_t) - R(x_t, a_t)\right)^2\right]\right)^{\frac{1}{3}} T^{\frac{2}{3}}\right).$$

by properly choosing ϵ. (Hint: you may find AM-GM $\frac{x+y+z}{3} \geq (xyz)^{\frac{1}{3}}$ useful)

4 Implementing contextual bandits algorithms

In this problem, we will implement exploration mechanisms for contextual bandits discussed in the class. The starter code is in `https://bahh723.github.io/course/HW/HW1.py`.

4.1 Data

To simulate a contextual bandit environment, we use existing classification dataset for supervised learning. Specifically, in this homework, we mainly use the digit dataset, which has 1797 data samples (i.e.g, feature-label pairs). Each feature vector has dimension 64 because it is a 8×8 gray-scale image, and each label belongs to $\{0, 1, \ldots, 9\}$.

Based on the relation between supervised classification and contextual bandits, we can simulate a contextual bandit environment using this dataset. In round t, the learner sees a context $x_t \in \mathbb{R}^{64}$ and takes an action $a_t \in \{0, 1, \ldots, 9\}$. The learner receives a reward of 1 if a_t equals the true label, and 0 if not.

4.2 Linear Function Approximation

In this problem, you will implement all exploration schemes for linear function approximation that we discussed in the class. This includes 1) LinUCB, 2) Thompson Sampling, 3) ϵ-Greedy, 4) Inverse Gap Weighting, 5) Boltzmann Exploration. Notice that all these algorithms share the same "linear regression" component; the only difference is just how they generate a_t based on $\hat{\theta}_t$ and Λ_t. Therefore, it is easy to implement them together at once. Besides these five, you are also required to implement two baseline methods: one is 6) Greedy, which is just ϵ-Greedy with $\epsilon = 0$. This is a strong baseline according to Bietti et al. (2021). The other is 7) Full-Information, that is, assuming the learner can see all actions' reward in each round. The performance in the full-information setting should serve as a performance upper bound for the bandit setting.

Below are some things to notice in the implementation.

- In the class, our formulation for linear contextual bandit is that $r \approx \phi(x, a)^\top \theta$. Notice that the feature vector $\phi(x, a)$ depends both on the context and the action. However, now our feature is just the 64-dimensional vector, which is shared among all actions. How should we handle this case? In fact, these are two common formulations of linear contextual bandits, and one can convert one from another. See Appendix B for how to convert the latter to the former. We provide the conversion in the appendix. Based on the idea there, together with some algebraic manipulation, it turns out the linear regression problem you will implement is the following. Say $\Phi(x_t) \in \mathbb{R}^{64}$ is the image feature received at round t. Define for every $a \in \{0, 1, \ldots, 9\}$,

$$\Lambda_{t,a} = \lambda I + \sum_{i=1}^{t-1} \mathbb{I}\{a_i = a\} \Phi(x_i) \Phi(x_i)^\top, \qquad \hat{\theta}_{t,a} = \Lambda_{t,a}^{-1} \sum_{i=1}^{t-1} \mathbb{I}\{a_i = a\} \Phi(x_i) r_i. \tag{2}$$

  The prediction for the reward of action a would then be $\Phi(x_t)^\top \hat{\theta}_{t,a}$, and the bonus in LinUCB would be $\sqrt{\beta} \|\Phi(x_t)\|_{\Lambda_{t,a}^{-1}}$.

- All constants like β in LinUCB, γ in Inverse Gap Weighting, or the λ in Eq. (2) should be viewed as hyperparameters to tune. Do not follow the theoretically optimal value derived in the class.

- In the class, Inverse Gap Weighting (or SquareCB) is expressed as

$$p_t(a) = \frac{1}{\lambda + \gamma \text{Gap}_t(a)}$$

  with the normalization factor λ that makes p_t a distribution. The value of λ has to be found through binary search, which adds some complexity to the implementation. A simpler version is

$$p_t(a) = \begin{cases} \frac{1}{A + \gamma \text{Gap}_t(a)} & \text{if } a \neq \text{argmax}_{a'} \hat{R}_t(x_t, a') \\ 1 - \sum_{a' \neq a} p_t(a') & \text{otherwise} \end{cases}$$

  which is also theoretically sound as mentioned in the original SquareCB paper (Foster and Rakhlin, 2020).

- Notice that the updates of $\Lambda_{t,a}$ every time is by adding a rank-one matrix. If you want to accelerate the calculation of $\Lambda_{t,a}^{-1}$, take a look at Vieira (2021).

4.3 General Function Approximation

For general function approximation, you are only required to implement 1) ϵ-Greedy, 2) Inverse Gap Weighting, 3) Boltzmann Exploration, 4) Greedy, 5) Full-Information. You do not need to implement the RegCB algorithm since it involves a slightly more complicated binary search procedure. However, you can earn some bonus points if you implement it.

To find the best approximated reward function, one can do

$$\hat{\theta}_t = \operatorname*{argmin}_{\theta} \sum_{i=1}^{t-1} \left(R_\theta(x_i, a_i) - r_i\right)^2 \tag{3}$$

where $R_\theta(x, a)$ is a parameterized function with parameter θ. This is implemented as a two-layer neural network in the starter code. But performing Eq. (3) in every round requires going over the dataset for $T = 1797$ times. This may be okay for our toy example, but would not be scalable when the number of rounds become larger. Notice that this does not happen in the linear case, because in there Eq. (3) has a closed form that can be obtained through the succinct expressions of $\sum_i \phi(x_i, a_i)\phi(x_i, a_i)^\top$ and $\sum_i \phi(x_i, a_i) r_i$.

One option is to only perform Eq. (3) once in a while. Another approach is to use *replay buffer*. That is, each time, after obtaining the tuple (x_t, a_t, r_t), we add it to a set $\mathcal{S}$. When we would like to update θ, we sample a batch $\mathcal{S}'$ from $\mathcal{S}$ and perform a gradient descent step on θ by minimizing $\sum_{(x,a,r)\in\mathcal{S}'} \left(R_\theta(x, a) - r\right)^2$. A simple replay buffer approach is implemented in the starter code. You can decide the relative frequency between obtaining new samples and updating θ.

Below are the tasks. In (b) to (d), there is flexibility to represent the results in a way you feel clear, as long as it fulfills the requirement.

(a) (25%) Read and understand the starter code. Implement the TODO's based on the instructions above. You can make any modifications to the starter code, but try to place all codes in a single file.

(b) (10%) Pick an algorithm that you like (e.g., linear function approximation + ϵ-greedy), and perform hyper-parameter tuning. Record in a table the values of the hyper-parameter you have tried, and their corresponding performance. Describe how you conduct the experiments so that others can reproduce the results.

(c) (10%) Generate two figures, one for linear function approximation, and the other for general function approximation. In each figure, compare the learning curves of different methods (x-axis: time t, y-axis: the learner's average reward in $[1, t]$). For each method, just plot one curve based the best hyper-parameters you have found. Similarly, describe how you conduct the experiments.

(d) (8%) Elaborate on any interesting observations or tricks you have found in the experiments.

(e) [optional] (15%) Implement the RegCB method, and include it in (b)-(d).

(f) [optional] (15%) Find another (preferably larger-scale) dataset and repeat (b)-(d).

(g) (2%) Did you leverage large language models to finish this problem? If so, how did you use it?

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)

Appendix

A Useful theorems

Theorem 1 (Hoeffding's Inequality)**.** *Let $X_1, X_2, \ldots, X_N$ be i.i.d. σ-sub-Gaussian random variables with mean μ. Then with probability at least $1-\delta$,*

$$\left|\frac{1}{N}\sum_{i=1}^{N} X_i - \mu\right| \leq \sqrt{\frac{2\log(2/\delta)}{N}}.$$

Theorem 2 (Concentration Inequality for Self-Normalized Process)**.** *Let $(\phi_1, w_1, \phi_2, w_2 \ldots, \phi_T, w_T)$ be a sequence of random variables where $\phi_i \in \mathbb{R}^d$ and $w_i \in \mathbb{R}$. Define $\mathcal{H}_t = (\phi_1, w_1, \ldots, \phi_{t-1}, w_{t-1})$. Suppose that ϕ_t may be arbitrarily depend on $\mathcal{H}_t$, but conditioned on $\mathcal{H}_t$, w_t is zero-mean and 1-sub-Gaussian. Then with probability at least $1-\delta$, for all $t \in [T+1]$,*

$$\left\|\sum_{i=1}^{t-1}\phi_i w_i\right\|^2_{\Lambda_t^{-1}} \leq d\log\left(1+\frac{T}{d}\right) + 2\log(1/\delta),$$

where $\Lambda_t := I + \sum_{i=1}^{t-1}\phi_i\phi_i^\top$.

Theorem 3 (Elliptical Potential Lemma)**.** *Let $\phi_i \in \mathbb{R}^d$ and $\|\phi_i\|_2 \leq 1$. Define $\Lambda_t = I + \sum_{i=1}^{t-1}\phi_i\phi_i^\top$. Then*

$$\sum_{t=1}^{T}\|\phi_t\|^2_{\Lambda_t^{-1}} \leq d\log\left(1+\frac{T}{d}\right).$$

B Converting action-independent context to action-dependent context

Define

$$\phi(x,a) = \begin{bmatrix} | & | & & | & & | \\ \mathbf{0} & \mathbf{0} & \cdots & \Phi(x) & \cdots & \mathbf{0} \\ | & | & & | & & | \end{bmatrix} \in \mathbb{R}^{64\times 10}, \qquad \theta^\star = \begin{bmatrix} | & | & & | & & | \\ \theta_1^\star & \theta_2^\star & \cdots & \theta_a^\star & \cdots & \theta_A^\star \\ | & | & & | & & | \end{bmatrix} \in \mathbb{R}^{64\times 10},$$

where in $\phi(x,a)$, the only non-zero column is the a-th column. In this way, we can model the reward of action a when seeing context x as $\langle\phi(x,a),\theta^\star\rangle = \langle\Phi(x),\theta_a^\star\rangle$.

References

Bietti, A., Agarwal, A., and Langford, J. (2021). A contextual bandit bake-off. *The Journal of Machine Learning Research*, 22(1):5928–5976.

Foster, D. and Rakhlin, A. (2020). Beyond ucb: Optimal and efficient contextual bandits with regression oracles. In *International Conference on Machine Learning*, pages 3199–3210. PMLR.

Lattimore, T., Szepesvari, C., and Weisz, G. (2020). Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, pages 5662–5670. PMLR. `https://arxiv.org/pdf/1911.07676.pdf`.

Vieira, T. (2021). Fast rank-one updates to matrix inverse? `https://timvieira.github.io/blog/post/2021/03/25/fast-rank-one-updates-to-matrix-inverse/`.