Principles of Online Decision-Making (CS-303)

Problem Set 3

Problem 1

(a) For the linear bandit model, show that the solution to

$$\hat{\theta}_t = \arg\min_{\theta \in \mathbb{R}^d} \left(\sum_{s=1}^{t} (X_s - \langle \theta, A_s \rangle)^2 + \lambda \|\theta\|_2^2 \right) \tag{1}$$

is indeed $\hat{\theta}_t = V_t^{-1} \sum_{s=1}^{t} A_s X_s$, with $V_0 = \lambda I$ and $V_t = V_0 + \sum_{s=1}^{t} A_s A_s^T$.

(b) In this exercise, we aim to understand why the region of uncertainty is an ellipse with short axes in directions where the arm distribution has a lot of variation, and vice versa. For this, suppose we generate arm vectors A_t from a Gaussian distribution $N(0, \Sigma)$. Find the distribution of $\hat{\theta}_t$ as a function of Σ, θ^*, and t when t grows large (so, you can use approximations that are valid when t is large such as the law of large numbers).

Problem 2

The `LinUCB` algorithm prescribes that the arm to be played, A_t, be chosen as

$$(A_t, \tilde{\theta}_t) = \arg \max_{(a,\theta) \in \mathcal{A}_t \times \mathcal{C}_t} (\langle \theta, a \rangle) \tag{2}$$

Show that if the confidence set $\mathcal{C}_t$ is given by an ellipsoid:

$$\mathcal{C}_t = \left\{ \theta \in \mathbb{R}^d \colon \ \|\theta - \hat{\theta}_t\|_{V_t}^2 \leq \beta_t \right\}, \tag{3}$$

then the arm chosen by `LinUCB` is given by

$$A_t = \arg\max_{a \in \mathcal{A}_t} \left(\langle \hat{\theta}_t, a \rangle + \sqrt{\beta_t} \|a\|_{V_t^{-1}} \right). \tag{4}$$

Problem 3

In the lecture, we have seen the Thompson sampling for the Bernoulli bandit setting. In this homework we use Bernoulli Thompson sampling, but we also see how to use it in a Gaussian bandit setting. For this, we assume that the distribution of the arm i is given the form of $\mathcal{N}(\mu_i, s)$, where s is known to us. We further assume that we somehow know that μ_i follows a Gaussian distribution $\mathcal{N}(\mu, \sigma_\mu)$ (These can be just hyper parameters). The pseudo code for the Gaussian Thompson sampling is as follows:

We provide incomplete algorithms for Thompson sampling for the Bernoulli and Gaussian case in bandit_algorithm.py.

(a) Complete the code marked by #PODMexercise in bandit_algorithm.py.

(b1) Experiment with two Bernoulli arms of expected rewards 0.6 and 0.7., and plot the evolution of the posteriors for both arms (you can use the `plot_thompson_3d()` function we provide in exp_structure.py). What do the posteriors converge to as the horizon goes to infinity?

(b2) Is this always the case if we change the expected rewards of the arms? Why is that?

Algorithm 1: Thompson Sampling for Gaussian Bandits

Let the counter for arm pulls be $N[i] = 0$ for any $i \in [k]$

Let average rewards be $\bar{X}[i] = 0$ for any $i \in [k]$

Let $\hat{\mu}_i(0) = \mu$ and $\hat{\sigma}_i(0) = \sigma_\mu$

For $t = 1, 2, \cdots$ do:

For $i = 1, 2, \cdots, k$ do:

Sample $\tilde{\mu}_i(t) \sim N(\hat{\mu}_i(t), \hat{\sigma}_i(t))$

Choose $A_t = \arg\max_i \tilde{\mu}_i(t)$

Observe X_t,update $N[A_t] += 1$ and $\bar{X}[A_t] = \frac{X_t - (N[A_t]-1)\bar{X}[A_t]}{N[A_t]}$

update $\hat{\mu}_i(t), \hat{\sigma}_i(t)$ by

For every arm $i = 1, 2, \cdots, k$ do: $\hat{\sigma}_i(t+1) = \hat{\sigma}_i(t)$ and $\hat{\mu}_i(t+1) = \hat{\mu}_i(t)$

For arm A_t:

$$\hat{\sigma}_{A_t}(t+1) = \left(\frac{1}{\sigma_\mu^2} + \frac{N[A_t]}{s^2}\right)^{-1}$$

$$\hat{\mu}_{A_t}(t+1) = \hat{\sigma}_{A_t}(t+1)\left(\frac{\mu}{\sigma_\mu^2} + \frac{N[A_t]\bar{X}[A_t]}{s^2}\right)$$

(c1) The basic Thompson Sampling algorithm for Bernoulli rewards is implicitly designed for a case where the parameters of each arms are sampled uniformly in $[0, 1]$. How should you modify the Thompson Sampling algorithm if the means of arm 1 and arm 2 are sampled respectively from a Beta(2, 5) and a Beta(3, 2)? Confirm experimentally that your modification performs better than the original version (for this, plot the cumulative regret averaged over different seeds).

(c2) For one seed, use the `plot_thompson_3d()` function to plot the evolution of the posterior of the arms for both versions of the algorithm. How does the difference between the two versions of the algorithm appear on these visualisations?

(d) Use Thompson sampling for Gaussian Bandits (Algorithm 1). Experiment with two Gaussian arms of different means (specified later), and variance 1. Assume that the prior of each arm is $\mathcal{N}(0, 1)$ (*i.e.,* $\mu = 0, \sigma_\mu = 1$). Assume further that the variance of reward is known to the algorithm (so, input $s = 1$ to Algorithm 1). Plot the evolution of the posteriors and regrets for the following situations where means μ_1, μ_2 are:

1. -1 and 1
2. -10 and 10
3. 9 and 10
4. 10 and 20.

In which situation do you observe the algorithm properly functions and in which case does it not? Infer why such a phenomena occurs.

(e) Again experiment with Algorithm 1. Set the Gaussian bandit environment to have two Gaussian arms of means -1 and 1, and variance 100 (*i.e.,* $\mu_1 = -1, \mu_2 = 1$, $s = 1$. Note here, these parameters are for the bandit environment. Not the input for the algorithm). Assume that the prior of each arm is $\mathcal{N}(0, 1)$ (*i.e.,* $\mu = 0, \sigma_\mu = 1$). In question (d), we assumed that the variance of reward is known to the algorithm. However, in this question, we misspecify the variance of reward. Specifically, assume that the variance of reward (s as the input for Algorithm 1) is

1. 1
2. 10^4

Plot the evolution of the posteriors and regrets for both arms. Did it work well? How does this misspecification affect the performance of the algorithm?