Principles of Online Decision-Making (CS-303)

Problem Set 3 - Solutions

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 $V_t = V_0 + \sum_{s=1}^{t} A_s A_s^T$.

We simply find the minimum of the function $f(\theta) = \left(\sum_{s=1}^{t} (X_s - \langle \theta, A_s \rangle)^2 + \lambda \|\theta\|_2^2 \right)$ by setting the gradient to zero:

$$\nabla f(\theta) \quad = \quad -2 \sum_{s=1}^{t} A_s (X_s - A_s^T \theta) + 2\lambda\theta = 0. \tag{2}$$

Rearranging terms gives the result directly.

(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).

First, with $A_t \sim N(0, \Sigma)$ (i.i.d.), we note that the matrix $V_t/t \to \Sigma$, by the law of large numbers. For the remainder, we ignore the noise in V_t for finite t and simply pose $V_t = t\Sigma$.

Then we write out

$$\hat{\theta} \quad = \quad t^{-1} \sum_{s=1}^{t} \Sigma^{-1} A_s X_s \tag{3}$$

$$\quad = \quad t^{-1} \sum_{s=1}^{t} \left\{ \Sigma^{-1} A_s A_s^T \theta^* + \Sigma^{-1} \eta_s A_s \right\} \tag{4}$$

$$\quad = \quad \theta^* + t^{-1} \Sigma^{-1} \sum_{s=1}^{t} \eta_s A_s. \tag{5}$$

Call the second term above B. We compute the covariance matrix $K_{BB} = \mathbb{E} B B^T$ of B.

$$K_{BB} \quad = \quad t^{-2} \mathbb{E} \left\{ \Sigma^{-1} \left(\sum_{s=1}^{t} \eta_s A_s \right) \left(\sum_{s=1}^{t} \eta_s A_s^T \right) \Sigma^{-1} \right\} \tag{6}$$

$$\quad = \quad t^{-2} \Sigma^{-1} \mathbb{E} \left\{ \left(\sum_{s=1}^{t} \eta_s A_s \right) \left(\sum_{s=1}^{t} \eta_s A_s^T \right) \right\} \Sigma^{-1}. \tag{7}$$

The expectation can be written as

$$\sum_{s_1=1, s_2=1}^{t,t} \mathbb{E} \left\{ \eta_{s_1} \eta_{s_2} A_{s_1} A_{s_2}^T \right\} = \sum_{s=1}^{t} \mathbb{E} \left\{ \eta_s^2 A_s A_s^T \right\} = t \mathbb{E}[\eta^2] \Sigma, \tag{8}$$

because η_s and A_s are independent and η_s are zero-mean. Call $\sigma^2 = \mathbb{E}[\eta^2]$ the variance of η_s. Finally,

$$K_{BB} = t^{-1}\Sigma^{-1}\mathbb{E}[\eta^2]\Sigma\Sigma^{-1} = \sigma^2\Sigma^{-1}/t. \tag{9}$$

Thus we have shown (except for the approximation $V_t/t = \Sigma$ for finite t) that $\hat{\theta} \sim N(\theta^*, \sigma^2\Sigma^{-1}/t)$. This explains why the confidence regions are shrinking ellipsoids, whose axes lengths are inverted relative to the arm distribution $N(0, \Sigma)$.

Problem 2

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

$$(A_t, \tilde{\theta}_t) = \arg \max_{(a,\theta)\in\mathcal{A}_t\times\mathcal{C}_t} (\langle\theta, a\rangle) \tag{10}$$

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

$$\mathcal{C}_t = \left\{\theta \in \mathbb{R}^d \colon \|\theta - \hat{\theta}_t\|^2_{V_t} \le \beta_t\right\}, \tag{11}$$

then the arm chosen by `LinUCB` is given by

$$A_t = \arg\max_{a\in\mathcal{A}_t} \left(\langle\hat{\theta}_t, a\rangle + \sqrt{\beta_t}\|a\|_{V_t^{-1}}\right). \tag{12}$$

Begin by expressing θ as $\theta = \hat{\theta}_t + (\theta - \hat{\theta}_t)$. Then we have

$$\langle\theta, a\rangle = \langle\hat{\theta}_t, a\rangle + \langle\theta - \hat{\theta}_t, a\rangle.$$

It suffices to show that

$$\max_{\theta\colon \|\theta-\hat{\theta}_t\|^2_{V_t}\le\beta_t} \langle\theta - \hat{\theta}_t, a\rangle = \sqrt{\beta_t}\|a\|_{V_t^{-1}}.$$

This is equivalent to showing that

$$\max_{x\colon \|x\|^2_{V_t}\le\beta_t} \langle x, a\rangle = \sqrt{\beta_t}\|a\|_{V_t^{-1}}.$$

Introducing the term $y = V_t^{1/2}x$, we see that

$$\|x\|^2_{V_t} = x^\top V_t x = (V_t^{1/2}x)^\top(V_t^{1/2}x) = y^\top y = \|y\|_2^2.$$

Therefore, the above optimisation problem is equivalent to showing

$$\max_{y\colon \|y\|_2^2\le\beta_t} \langle V_t^{-1/2}y, a\rangle = \sqrt{\beta_t}\|a\|_{V_t^{-1}}$$

or equivalently

$$\max_{y\colon \|y\|_2^2\le\beta_t} \langle y, V_t^{-1/2}a\rangle = \sqrt{\beta_t}\|a\|_{V_t^{-1}}.$$

Optimising over y on a circle of radius $\sqrt{\beta_t}$, we are free to choose the direction of y to be the same as that of $V_t^{-1/2}a$. Thus, we have

$$\max_{y\colon \|y\|_2^2\le\beta_t} \langle y, V_t^{-1/2}a\rangle = \sqrt{\beta_t}\|V_t^{-1/2}a\|_2 = \sqrt{\beta_t}\sqrt{a^\top V_t^{-1}a} = \sqrt{\beta_t}\|a\|_{V_t^{-1}}.$$

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:

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:

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

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

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.

The solution is in the solution code.

(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?

As the horizon goes to infinity, the posterior of each arm converges to a Dirac at the expected reward of the arm.

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

Yes, it is always the case because each arm will always be sampled an infinite number of times when

the time horizon goes to infinity. However the convergence is a lot slower for suboptimal arms, as they are played less and less often.

(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).

We can modify the algorithm so that it takes as parameters the prior of each arm. It is the same as assuming that we already observed one success and four failures for arm 1 and two successes and one failure for arm 2.

(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?

We can see that the posterior distributions are more concentrated for the modified version, especially in the first a few steps.

(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.

In (1) and (2), the algorithm successfully identifies the best arm. In (3) and (4), the algorithm often fails to identify the best arm (or takes significantly longer time to identify the right arm).

In (1), all the parameters are properly input. In the other situations, either prior means, prior variances, or both are misspecified.

Because of misspecification, the prior means (0.0) in (3) are very different from the true means (9, 10), in the first step because the reward actually obtained by pulling an arm is large, this change the estimated reward of pulled arm significantly. The arm that is pulled initially has significantly higher reward estimate (that should be close to 9 or 10) than the other arm (whose estimated reward is 0). Because of this, it can take significantly longer time to identify the best arm.

On top of the reason that (3) fails, the algorithm gets overconfident for the estimated means in (4), and explores the other arm not enough in the beginning.

In (2), the prior variance is small compared to the true means, but the separation between the two arms is large enough, so the algorithm can identify the best arm.

(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?

When the variance is misspecified to be 1, the algorithm occasionally fails to identify the best arm. This is because the algorithm becomes overconfident in its estimates after each observation, updating the posterior too aggressively. This can cause the algorithm to prematurely commit to a suboptimal arm based on early random fluctuations, resulting in higher regret and slower identification of the best arm.

Conversely, when the variance is misspecified to be 10000, the algorithm updates its beliefs very slowly, remaining uncertain about the true means for a long time. This leads to excessive exploration and delayed convergence to the optimal arm, also increasing regret.