CSE 541: Interactive Learning - Homework 2

Abhishek Saini

Confidence Bounds

Problem 1.1

(Wald's Identity) Let X_1, X_2, ... be a sequence of iid random variables. For $j \in \{0, 1\}$, under $\mathbf{H}_j$ we have that $X_i \sim p_j$. Let $\mathbb{P}_j(.), \mathbb{E}_j()$ denote the probability and expectation under $\mathbf{H}_j$. Assume that the support of p_0 and p_1 are equal and furthermore, that $\sup_{x \in support(p_0)} \frac{p_1(x)}{p_0(x)} \leq \kappa$. Fix some $\delta \in (0, 1)$. If $L_t = \prod_{i=1}^{t} \frac{p_1(X_i)}{p_0(X_i)}$ and $\tau = \min\{t : L_t > 1/\delta\}$, we showed in class that the false alarm probability $\mathbb{P}_0(L_\tau > 1/\delta) \leq \delta$. Assume that $\mathbb{E}_1[\tau] < \infty$. Show that $\frac{\log(1/\delta)}{KL(p_1 \| p_0)} \leq \mathbb{E}_1[\tau] \leq \frac{\log(\kappa/\delta)}{KL(p_1 \| p_0)}$ where $KL(p_1 \parallel p_0) = \int p_1(x) \log\left(\frac{p_1(x)}{p_0(x)}\right) dx$ is the Kullback Leibler divergence between p_1 and p_0.

Proof. Let us define $S_t = \log(L_t)$ and $R_t = \log\left(\frac{p_1(X_t)}{p_0(X_t)}\right)$

$$\begin{aligned} S_t = \log(L_t) &= \log\left(\prod_{i=1}^{t} \frac{p_1(X_i)}{p_0(X_i)}\right) \\ &= \sum_{i=1}^{t} \log\left(\frac{p_1(X_i)}{p_0(X_i)}\right) \\ &= \left[\sum_{i=1}^{t-1} \log\left(\frac{p_1(X_i)}{p_0(X_i)}\right)\right] + \log\left(\frac{p_1(X_t)}{p_0(X_t)}\right) \\ &= S_{t-1} + R_t \end{aligned}$$

We know that $\tau = \min\{t : L_t > 1/\delta\}$. At time t, $L_1, L_2, \ldots, L_t$ is deterministic. The event $\mathbf{1}\{\tau \leq t\}$ is $\mathcal{F}_t$-measurable for all $t \in \mathbb{N}$. Hence, τ is a stopping time. Further, τ is assumed to have finite expectation. $R_t = \log\left(\frac{p_1(X_t)}{p_0(X_t)}\right)$ has a finite mean as follows.

$$\begin{aligned} \mathbb{E}_1[R_t] &= \int \log\left(\frac{p_1(x)}{p_0(x)}\right) p_1(x)\, dx \\ &\leq \int \log(\kappa) p_1(x)\, dx = \log(\kappa) \qquad \text{inequality from problem definition} \end{aligned}$$

Also, $\mathbb{E}_1[R_t]$ is the same for all $t \in \mathbb{N}$ because X_t are iid. Note, $\mathbb{E}_1[R_t] = KL(p_1 \parallel p_0)$.
Using Wald's equation, we get,

$$\begin{aligned} \mathbb{E}_1[S_\tau] &= \mathbb{E}_1[\tau]\, \mathbb{E}_1[R_1] \\ &= \mathbb{E}_1[\tau] KL(p_1 \parallel p_0) \qquad \text{(Wald's Identity)} \end{aligned}$$

From the stopping time condition, $S_\tau = \log(L_\tau) > \log(1/\delta)$. Therefore, $\mathbb{E}_1[S_\tau] > \log(1/\delta)$.
Also at time $\tau - 1$, $S_{\tau-1} \leq \log(1/\delta)$ (or it doesn't exist if $\tau = 1$), else $\tau - 1$ would be the stopping time which contradicts definition of τ.

$$\begin{aligned} S_\tau &= S_{\tau-1} + R_\tau \\ &\leq \log(1/\delta) + R_\tau && \text{since we don't stop at } \tau - 1 \\ &\leq \log(1/\delta) + \log(\kappa) = \log(\kappa/\delta) && \text{inequality from problem definition} \end{aligned}$$

Thus, $\mathbb{E}_1[S_\tau] \leq \log(\kappa/\delta)$
Plugging the upper and lower bounds on $\mathbb{E}_1[S_\tau]$ back in (Wald's Identity) and rearranging, we get,

$$\frac{\log(1/\delta)}{KL(p_1 \parallel p_0)} \leq \mathbb{E}_1[\tau] \leq \frac{\log(\kappa/\delta)}{KL(p_1 \parallel p_0)}$$

□

Problem 1.2

(Method of Mixtures) Let $X_1, X_1, \ldots$ be a sequence of iid random variables where $X_1 \sim \mathcal{N}(\mu, 1)$. Under $\mathbf{H}_0$ we have $\mu = 0$ and under $\mathbf{H}_1$ assume $\mu = \theta$.

- a) Define $L_t(\theta) := \exp\big(S_t\theta - t\theta^2/2\big)$ where $S_t = \sum_{s=1}^t X_s$. Show that $\prod_{i=1}^t \frac{p_1(X_i)}{p_0(X_i)} = L_t(\theta)$

Proof.

$$\begin{aligned} \prod_{i=1}^t \frac{p_1(X_i)}{p_0(X_i)} &= \prod_{i=1}^t \frac{\frac{1}{\sqrt{2\pi}} \exp\big(-(X_i - \theta)^2/2\big)}{\frac{1}{\sqrt{2\pi}} \exp(-(X_i)^2/2)} \\ &= \prod_{i=1}^t \exp\big(-(X_i - \theta)^2/2 + X_i^2/2\big) \\ &= \prod_{i=1}^t \exp\big(X_i\theta - \theta^2/2\big) \\ &= \exp\left(\sum_{i=1}^t \{X_i\theta - \theta^2/2\}\right) \\ &= \exp\big(S_t\theta - t\theta^2/2\big) = L_t(\theta) \end{aligned}$$

□

- b) Now suppose the sequence $X_1, X_2, \ldots$ is still iid and $\mathbb{E}[X_1] = \mu$ in the above notation, but now the distribution of X_1 is unknown other than the knowledge that $\mathbb{E}[\exp(\lambda(X_1 - \mu))] \leq \exp\big(\lambda^2/2\big)$. Show that $L_t(\theta)$ defined above is a super-martingale under $\mathbf{H}_0$

Proof. If X_t is a super-martingale, $\mathbb{E}[X_{t+1}|\mathcal{F}_t] \leq X_t$

$$
\begin{aligned}
\mathbb{E}[L_{t+1}(\theta)|\mathcal{F}_t] &= \mathbb{E}[\exp\big(S_{t+1}\theta - (t+1)\theta^2/2\big)|\mathcal{F}_t] && \text{by definition of } L_t(\theta)\\
&= \mathbb{E}[\exp\big((S_t + X_{t+1})\theta - (t+1)\theta^2/2\big)|\mathcal{F}_t]\\
&= \mathbb{E}[\exp\big(S_t\theta - t\theta^2/2 + X_{t+1}\theta - \theta^2/2\big)|\mathcal{F}_t]\\
&= \mathbb{E}[L_t(\theta)\exp\big(X_{t+1}\theta - \theta^2/2\big)|\mathcal{F}_t] && \text{by definition of } L_t(\theta)\\
&= L_t(\theta)\exp\big(-\theta^2/2\big)\,\mathbb{E}[\exp(X_{t+1}\theta)|\mathcal{F}_t] && L_t(\theta) \text{ is deterministic given } \mathcal{F}_t\\
&= L_t(\theta)\exp\big(-\theta^2/2\big)\,\mathbb{E}[\exp(X_{t+1}\theta)] && X_i\text{'s are independent rvs}\\
&\le L_t(\theta)\exp\big(-\theta^2/2\big)\exp\big(\theta^2/2\big) && \text{applying subgaussianity under } \mathbf{H}_0\\
&= L_t(\theta)
\end{aligned}
$$

Hence, $L_t(\theta)$ is a super-martingale under $\mathbf{H}_0$

□

- c) Assume the setting of the previous step. Define $\bar{L}_t = \int_\theta L_t(\theta)h(\theta)\,d\theta$ where $h(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\Big(-\frac{\theta^2}{2\sigma^2}\Big)$ and $\sigma > 0$. Show that $\bar{L}_t$ is a super-martingale under $\mathbf{H}_0$.

Proof.

$$
\begin{aligned}
\mathbb{E}[\bar{L}_{t+1}|\mathcal{F}_t] &= \mathbb{E}[\int_\theta L_{t+1}(\theta)h(\theta)\,d\theta|\mathcal{F}_t]\\
&= \int_\theta \mathbb{E}[L_{t+1}(\theta)h(\theta)|\mathcal{F}_t]\,d\theta && \text{Fubini's theorem}\\
&= \int_\theta \mathbb{E}[\exp\big(S_{t+1}\theta - (t+1)\theta^2/2\big)h(\theta)|\mathcal{F}_t]\,d\theta && \text{by definition of } L_t(\theta)\\
&= \int_\theta \mathbb{E}[\exp\big((S_t + X_{t+1})\theta - (t+1)\theta^2/2\big)h(\theta)|\mathcal{F}_t]\,d\theta\\
&= \int_\theta \mathbb{E}[L_t(\theta)\exp\big(X_{t+1}\theta - \theta^2/2\big)h(\theta)|\mathcal{F}_t]\,d\theta && \text{by definition of } L_t(\theta)\\
&= \int_\theta L_t(\theta)\exp\big(-\theta^2/2\big)h(\theta)\,\mathbb{E}[\exp(X_{t+1}\theta)|\mathcal{F}_t]\,d\theta && L_t(\theta) \text{ is deterministic given } \mathcal{F}_t\\
&= \int_\theta L_t(\theta)\exp\big(-\theta^2/2\big)h(\theta)\,\mathbb{E}[\exp(X_{t+1}\theta)]\,d\theta && X_i\text{'s are independent rvs}\\
&\le \int_\theta L_t(\theta)\exp\big(-\theta^2/2\big)h(\theta)\exp\big(\theta^2/2\big)\,d\theta && \text{applying subgaussianity under } \mathbf{H}_0\\
&= \int_\theta L_t(\theta)h(\theta)\,d\theta\\
&= \bar{L}_t && \text{by definition of } \bar{L}_t
\end{aligned}
$$

Hence, $\bar{L}_t$ is a super-martingale under $\mathbf{H}_0$ □

- d) Fix $\delta \in (0,1)$. Show that

$$\mathbb{P}_0(\exists t \in \mathbb{N} : \bar{L}_t > 1/\delta) \le \delta \tag{1}$$

Proof. Since $\bar{L}_t$ is a super-martingale under $\mathbf{H}_0$, and $\bar{L}_t \geq 0$, by maximal inequality,

$$\begin{aligned}\mathbb{P}_0(\sup_{t\in\mathbb{N}} \bar{L}_t \geq 1/\delta) \leq \frac{\mathbb{E}_0[\bar{L}_0]}{1/\delta} &= \frac{\mathbb{E}_0[\int_\theta L_0(\theta)h(\theta)\,d\theta]}{1/\delta} \\ &= \frac{\mathbb{E}_0[\int_\theta \exp\big(S_0\theta - 0\times\theta^2/2\big)h(\theta)\,d\theta]}{1/\delta} \\ &= \frac{\mathbb{E}_0[\int_\theta h(\theta)\,d\theta]}{1/\delta} = \delta\end{aligned}$$

□

- e) Conclude that if $Z_1, Z_2, \ldots$ are iid random variables with $\mathbb{E}[\exp(\lambda(Z_1 - \mathbb{E}[Z_1]))] \leq \exp\big(\lambda^2/2\big)$, then for any $\sigma > 0$

$$\mathbb{P}\left(\exists t \in \mathbb{N} : \left|\frac{1}{t}\sum_{s=1}^{t}(Z_s - \mathbb{E}[Z_s])\right| > \sqrt{1+\frac{1}{t\sigma^2}}\sqrt{\frac{2\log(1/\delta)+\log(t\sigma^2+1)}{t}}\right) \leq \delta$$

Proof. Let's first calculate $\bar{L}_t$.

$$\begin{aligned}
\bar{L}_t &= \int_\theta L_t(\theta)h(\theta)\,d\theta \\
&= \int_\theta \exp\big(S_t\theta - t\theta^2/2\big)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\theta^2}{2\sigma^2}\right)d\theta \\
&= \int_\theta \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(S_t\theta - \frac{t\sigma^2+1}{2\sigma^2}\theta^2\right)d\theta \\
&= \int_\theta \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{t\sigma^2+1}{2\sigma^2}\left[\frac{2\sigma^2}{t\sigma^2+1}S_t\theta - \theta^2\right]\right)d\theta \\
&= \int_\theta \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{t\sigma^2+1}{2\sigma^2}\left[\left(\frac{\sigma^2 S_t}{t\sigma^2+1}\right)^2 - \left(\theta - \frac{\sigma^2 S_t}{t\sigma^2+1}\right)^2\right]\right)d\theta && \text{completing squares} \\
&= \int_{\theta'} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{t\sigma^2+1}{2\sigma^2}\left[\left(\frac{\sigma^2 S_t}{t\sigma^2+1}\right)^2 - \theta'^2\right]\right)d\theta' && \theta' = \theta - \frac{\sigma^2 S_t}{t\sigma^2+1} \\
&= \exp\left(\frac{t\sigma^2+1}{2\sigma^2}\left(\frac{\sigma^2 S_t}{t\sigma^2+1}\right)^2\right)\int_{\theta'} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{t\sigma^2+1}{2\sigma^2}\theta'^2\right)d\theta' \\
&= \exp\left(\frac{\sigma^2 S_t^2}{2(t\sigma^2+1)}\right)\int_{\theta'} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\theta'^2}{2\sigma'^2}\right)d\theta' && \sigma'^2 = \frac{\sigma^2}{t\sigma^2+1} \\
&= \exp\left(\frac{\sigma^2 S_t^2}{2(t\sigma^2+1)}\right)\frac{\sigma'}{\sigma}\int_{\theta'} \frac{1}{\sqrt{2\pi\sigma'^2}}\exp\left(-\frac{\theta'^2}{2\sigma'^2}\right)d\theta' \\
&= \exp\left(\frac{\sigma^2 S_t^2}{2(t\sigma^2+1)}\right)\frac{\sigma'}{\sigma} && \text{pdf integrates to 1} \\
&= \sqrt{\frac{1}{t\sigma^2+1}}\exp\left(\frac{\sigma^2 S_t^2}{2(t\sigma^2+1)}\right) && \sigma'^2 = \frac{\sigma^2}{t\sigma^2+1}
\end{aligned}$$

Substituting $\bar{L}_t$ back in (1),

$$\begin{aligned}
\delta &\geq \mathbb{P}_0(\exists t \in \mathbb{N} : \bar{L}_t > 1/\delta) \\
&= \mathbb{P}_0\left(\exists t \in \mathbb{N} : \sqrt{\frac{1}{t\sigma^2+1}}\exp\left(\frac{\sigma^2 S_t^2}{2(t\sigma^2+1)}\right) > 1/\delta\right) \\
&= \mathbb{P}_0\left(\exists t \in \mathbb{N} : \exp\left(\frac{\sigma^2 S_t^2}{2(t\sigma^2+1)}\right) > \sqrt{\frac{t\sigma^2+1}{\delta^2}}\right) \\
&= \mathbb{P}_0\left(\exists t \in \mathbb{N} : \frac{\sigma^2 S_t^2}{2(t\sigma^2+1)} > \log\left(\sqrt{\frac{t\sigma^2+1}{\delta^2}}\right)\right) \\
&= \mathbb{P}_0\left(\exists t \in \mathbb{N} : S_t^2 > \frac{t\sigma^2+1}{\sigma^2}\log\left(\frac{t\sigma^2+1}{\delta^2}\right)\right) \\
&= \mathbb{P}_0\left(\exists t \in \mathbb{N} : |S_t| > \sqrt{\frac{t\sigma^2+1}{\sigma^2}}\sqrt{2\log(1/\delta)+\log(t\sigma^2+1)}\right) \\
&= \mathbb{P}_0\left(\exists t \in \mathbb{N} : \frac{1}{t}|S_t| > \frac{1}{t}\sqrt{\frac{t\sigma^2+1}{\sigma^2}}\sqrt{2\log(1/\delta)+\log(t\sigma^2+1)}\right) \\
&= \mathbb{P}_0\left(\exists t \in \mathbb{N} : \left|\frac{1}{t}\sum_{s=1}^{t}(Z_s - \mathbb{E}[Z_s])\right| > \sqrt{1+\frac{1}{t\sigma^2}}\sqrt{\frac{2\log(1/\delta)+\log(t\sigma^2+1)}{t}}\right)
\end{aligned}$$

□

Problem 1.3

(Best arm identification) Consider an n-armed multi-armed bandit problem where the jth pull of the ith arm yields a random variable $X_{i,j} \sim \mathcal{N}(\theta_i, 1)$. The objective of the player is to strategically pull arms until getting to a point that they can predict the index of the arm with the highest mean, at which time they stop and output this estimated arm. Suppose an oracle told the player that θ, the true means they are playing against, is equal to $\Delta\mathbf{e}_j$ for some $j = 1, \ldots, n$ where $\mathbf{e}_j$ is a vector of all zeros except a 1 in the jth location. Note that while $\Delta > 0$ is known to the player, which is the true j is unknown.

- a) Due to the symmetry of the known problem setup, it is conceivable that the following algorithm is optimal: play every arm the same number of times (say, τ times), and then declare that the arm with the highest empirical mean is best. Provide a sufficient condition on τ such that such a procedure correctly identifies the location of the true j index with probability at least $1 - \delta$ (don't forget the union bound!).

Let Z_i be the random variable denoting the mean of arm i after τ pulls. $Z_i = \frac{1}{\tau}\sum_{j=1}^{\tau} X_{i,j}$. $Z_i \sim \mathcal{N}(\theta_i, 1/\tau)$ where $\theta_i = 0\ \forall i \in [n], i \neq j$ and $\theta_j = \Delta$
Let ξ be the event such that the correct arm is identified (i.e. arm j has the highest mean).

$$\xi = \bigcap_{i=1, i\neq j}^{n} \{Z_i < Z_j\}$$

ξ^c denotes the event that the incorrect arm is identified.

$$
\begin{aligned}
\mathbb{P}(\xi^c) &= \mathbb{P}(\{\bigcap_{i=1, i\neq j}^{n} \{Z_i < Z_j\}\}^c) \\
&= \mathbb{P}(\bigcup_{i=1, i\neq j}^{n} \{Z_i \geq Z_j\}) \\
&\leq \sum_{i=1, i\neq j}^{n} \mathbb{P}(\{Z_i \geq Z_j\}) && \text{Union Bound} \\
&= (n-1)\,\mathbb{P}(\{Z_i \geq Z_j\}) && \text{symmetric for all } i \neq j \\
&= (n-1)\,\mathbb{P}(\{Z_j - Z_i \leq 0\}) \\
&\leq (n-1)\exp\left(-\frac{\Delta^2}{2(2/\tau)}\right) && Z_j - Z_i \sim \mathcal{N}(\Delta, 2/\tau), \\
&&& \text{Cramer Chernoff Bound} \\
&< n\exp\left(-\frac{\tau\Delta^2}{4}\right)
\end{aligned}
$$

Let $\delta = n\exp\left(-\frac{\tau\Delta^2}{4}\right)$.
Thus, for $\tau \geq \lceil 4\Delta^{-2}\log(n/\delta)\rceil$, the correct arm will be chosen with probability at least $1-\delta$

- b) Argue that if each arm is pulled the same number of times this value of τ up to a constant factor (ignoring dependence on δ) is necessary.

Hint for this approach was provided by Lang Liu during office hours.

$Z_i \sim \mathcal{N}(0, 1/\tau)$ for $i \neq j$ and $Z_j \sim \mathcal{N}(\Delta, 1/\tau)$ Assuming the hint provided holds true, $\mathbb{P}(\max_{i=1,\ldots,n} Z_i \geq \sqrt{\sigma^2\log(n)}) > c$ for some absolute constant c. Therefore, $\mathbb{P}(\max_{i=1,\ldots,n, i\neq j} Z_i \geq \sqrt{\frac{\log(n)}{\tau}}) > c$ Suppose there exists no $k > 0$ such that $\tau > k\Delta^{-2}\log(n/\delta)$. Thus,

$$
\begin{aligned}
\tau &< k\Delta^{-2}\log(n/\delta) && \forall k > 0 \\
\frac{\log(n)\Delta^2}{\log(n/\delta)k} &< \frac{\log(n)}{\tau} && \text{rearranging terms} \\
\Delta^2 &< \frac{\log(n)}{\tau} && \text{Choosing } k = \frac{\log(n)}{\log(n/\delta)}
\end{aligned}
$$

Using the above inequality, $\mathbb{P}(\max_{i=1,\ldots,n, i\neq j} Z_i \geq \sqrt{\Delta^2}) > \mathbb{P}(\max_{i=1,\ldots,n, i\neq j} Z_i \geq \sqrt{\frac{\log(n)}{\tau}}) > c$.
Also, $\mathbb{P}(Type\_I\_error) > \mathbb{P}(\max_{i=1,\ldots,n, i\neq j} Z_i \geq \Delta)\,\mathbb{P}(Z_j < \Delta) > c/2$. This means we have a constant Type I error probability that doesn't depend on δ, Δ or n which cannot be improved upon unless we relax our earlier restriction on τ.

Proof of Hint:
(attempted using the hints shared by Professor Jamieson on Ed)

$$
\begin{aligned}
\mathbb{P}(\max_i Z_i \geq \sqrt{\sigma^2 \log(n)}) &= \mathbb{P}(\bigcup_i \{Z_i \geq \sqrt{\sigma^2 \log(n)}\}) \\
&= 1 - \mathbb{P}(\bigcap_i \{Z_i < \sqrt{\sigma^2 \log(n)}\}) && \text{complementary event, De Morgan's} \\
&= 1 - \mathbb{P}(Z_i < \sqrt{\sigma^2 \log(n)})^n && \text{Independence} \\
&= 1 - (1 - \mathbb{P}(Z_i \geq \sqrt{\sigma^2 \log(n)}))^n \\
&\geq 1 - \left[1 - \exp\big(-\sigma^2 \log(n)/2\big) \left(\frac{\sigma^2 \log(n) - 1}{(\sigma^2 \log(n))^{3/2}}\right)\right]^n && \text{Lower bound on Gaussian tail}
\end{aligned}
$$

The sequence $\left[1 - \exp\big(-\sigma^2 \log(n)/2\big) \left(\frac{\sigma^2 \log(n) - 1}{(\sigma^2 \log(n))^{3/2}}\right)\right]^n$ converges to 0 as n becomes larger and larger so the event happens almost surely. For smaller n, the Gaussian tail bound used above isn't an accurate bound but since we are dealing with probabilities of Gaussian random variable, they lie between 0 and 1 and don't become 0 or 1 for small n. Hence they are bounded below by some constant c as right hand side term in the inequality above never becomes 0.

- c) The previous two parts suggest that any algorithm that pulls every arm the same number of times and identifies the best arm requires essentially $n\Delta^{-2} \log(n/\delta)$ total pulls. We will beat this with an adaptive procedure that requires just $O(n\Delta^{-2} \log(1/\delta))$ pulls.
  Algorithm: Initialize $S_1 = [n]$. At each around $t \geq 1$, while $|S_t| > 1$, pull every arm in S_t and then set $S_{t+1} = \{i \in S_t : \sum_{s=1}^t (X_{i,s} - \Delta) \geq -\Delta t/2 - \frac{\log(1/\delta)}{\Delta}\}$.
  We showed in class that if $Z_s \sim \mathcal{N}(0,1)$ then for any $\alpha > 0$ and $\rho \in (0,1)$ we have

$$
\max\left\{\mathbb{P}\left(\bigcup_{t=1}^{\infty}\{\sum_{s=1}^{t} Z_s < -\frac{\alpha t}{2} - \frac{\log(1/\rho)}{\alpha}\}\right), \mathbb{P}\left(\bigcup_{t=1}^{\infty}\{\sum_{s=1}^{t} Z_s > \frac{\alpha t}{2} + \frac{\log(1/\rho)}{\alpha}\}\right)\right\} \leq \rho \tag{2}
$$

  Conclude that if j is the index of the best-arm (so that $X_{j,s} \sim \mathcal{N}(\Delta, 1)$) then with probability at least $1 - \delta$ arm j remains in S_t for all $t \geq 1$.

Proof. Arm j gets removed in some round t if and only if $\sum_{s=1}^t (X_{j,s} - \Delta) \geq -\Delta t/2 - \frac{\log(1/\delta)}{\Delta}$.
Hence, the event that arm j gets discarded is given by

$$
\bigcup_{t=1}^{\infty}\{\sum_{s=1}^{t}(X_{j,s} - \Delta) < -\frac{\alpha t}{2} - \frac{\log(1/\delta)}{\alpha}\}
$$

Since, arm j has the best arm, $(X_{j,s} - \Delta) \sim \mathcal{N}(0,1)$. Hence using $\rho = \delta$ in (2), we conclude that arm j never gets discarded with probability at least $1 - \delta$

□

- d) For $i \neq j$ (so that $X_{i,s} \sim \mathcal{N}(0,1)$) define the random variables

$$
\rho_i := \sup\left\{\rho \in (0,1) : \bigcap_{t=1}^{\infty}\{\sum_{s=1}^{t} X_{i,s} \leq \frac{\Delta t}{4} + \frac{\log(1/\rho)}{\Delta/2}\}\right\}
$$

  If $T_i = \max t : i \in S_t$ show that $T_i \leq 4\Delta^{-2} \log(1/\delta) + 8\Delta^{-2} \log(1/\rho_i)$.

Proof. Since at time T_i, arm $i \in S_t$, else it would have been discarded which contradicts the definition of T_i.

$$\begin{aligned}
\sum_{s=1}^{T_i}(X_{i,s} - \Delta) &\geq -\Delta T_i/2 - \frac{\log(1/\delta)}{\Delta} \\
S_{i,T_i} - \Delta T_i &\geq -\Delta T_i/2 - \frac{\log(1/\delta)}{\Delta} \\
S_{i,T_i} &\geq \Delta T_i/2 - \frac{\log(1/\delta)}{\Delta} \qquad (3)
\end{aligned}$$

By definition of ρ_i, it satisfies, $\sum_{s=1}^{T_i} X_{i,s} = S_{i,T_i} \leq \frac{\Delta T_i}{4} + \frac{\log(1/\rho_i)}{\Delta/2}$. Combining this inequality with (3), we get

$$\begin{aligned}
\Delta T_i/2 - \frac{\log(1/\delta)}{\Delta} &\leq \frac{\Delta T_i}{4} + \frac{\log(1/\rho_i)}{\Delta/2} \\
\frac{\Delta T_i}{4} &\leq \frac{\log(1/\delta)}{\Delta} + \frac{\log(1/\rho_i)}{\Delta/2} \\
T_i &\leq 4\Delta^{-2}\log(1/\delta) + 8\Delta^{-2}\log(1/\rho_i)
\end{aligned}$$

□

- e) Note that by (2) we have $\mathbb{P}(\rho_i \leq \rho) \leq \rho$ for any $\rho \in (0,1)$. Use this fact to show that $\mathbb{E}[\sum_{i=1}^n T_i] \leq cn\Delta^{-2}\log(1/\delta)$ for some absolute constant $c > 0$.

Proof. We don't treat the jth arm separately as T_j will not be played more than $max_{i,i\neq j} T_i$ by the algorithm.

$$\begin{aligned}
\mathbb{E}[\sum_{i=1}^n T_i] &= \sum_{i=1}^n \mathbb{E}[T_i] && \text{linearity of expectation} \\
&\leq \sum_{i=1}^n \mathbb{E}[4\Delta^{-2}\log(1/\delta) + 8\Delta^{-2}\log(1/\rho_i)] && \text{from the previous result} \\
&= 4n\Delta^{-2}\log(1/\delta) + 8\Delta^{-2}\sum_{i=1}^n \mathbb{E}[\log(1/\rho_i)] \\
&= 4n\Delta^{-2}\log(1/\delta) + 8\Delta^{-2}\sum_{i=1}^n \int_{x>0} \mathbb{P}(\log(1/\rho_i) > x)\, dx && \text{if } X>0,\ \mathbb{E}[X] = \int_{x>0}\mathbb{P}(X>x)\,dx \\
&= 4n\Delta^{-2}\log(1/\delta) + 8\Delta^{-2}\sum_{i=1}^n \int_{x>0} \mathbb{P}(\rho_i < \exp(-x))\, dx \\
&\leq 4n\Delta^{-2}\log(1/\delta) + 8\Delta^{-2}\sum_{i=1}^n \int_{x>0} \exp(-x)\, dx && \mathbb{P}(\rho_i \leq \rho) \leq \rho\ \forall\ \rho \in (0,1) \\
&= 4n\Delta^{-2}\log(1/\delta) + 8\Delta^{-2}\sum_{i=1}^n 1 \\
&= 4n\Delta^{-2}\log(1/\delta) + 8n\Delta^{-2}
\end{aligned}$$

□

Linear regression and experimental design

Problem 2.1

Exercise 20.2 of [SzepesvariLattimore] Let $n \geq 2d$ and $(\eta_t)_{t=1}^n$ be a sequence of independent standard Gaussian random variables. Find a sequence of random vectors $(A_t)_{t=1}^n$, with $A_t \in \mathbb{R}^d$ such that $V_n^{-1} = \sum_{t=1}^n A^T A$ is invertible almost surely and A_t is $\sigma(A_1, \eta_1, \ldots, A_{t-1}, \eta_{t-1})$-measurable for all t and

$$\mathbb{E}[\langle \hat{\theta}_n, \mathbf{1} \rangle^2 / \|\mathbf{1}\|^2_{V_n^{-1}}] \geq cd,$$

where $c > 0$ is a universal constant and $S_n = \sum_{t=1}^n \eta_t A_t$ and $\hat{\theta}_n = V_n^{-1} S_n$.

We use the following sequence of random vectors. Consider $\mathbf{e}_i$ to be the standard basis vectors of $\mathbb{R}^d$. For $t = 1, \ldots, d$, we play arm A_t given by the vector $\mathbf{e}_t$. This allows the covariance matrix to be invertible almost surely. From time $t = d + 1$ onwards, we play the arm given by the standard basis vector $A_t = \mathbf{e}_j$, where $j = (t-1) \,\%\, d + 1$, if the random variable $\eta_t > 0$.

Thus the matrix A^T is given as $\left[\begin{array}{ccc|ccc|c} 1 & 0 & \ldots & \mathbf{1}\{\eta_{d+1} > 0\} & 0 & \ldots & \\ 0 & 1 & & 0 & \mathbf{1}\{\eta_{d+2} > 0\} & & \ldots \\ \vdots & 0 & \ddots & \vdots & 0 & \ddots & \end{array}\right] = \left[\begin{array}{c|c|c|c} I & V_1 & \ldots & V_{\lceil n/d \rceil} \end{array}\right]$

Thus, the covariance matrix $A^T A = \left[\begin{array}{c|c|c|c} I & V_1 & \ldots & V_{\lceil n/d \rceil} \end{array}\right] \begin{bmatrix} I \\ V_1^T \\ \vdots \\ V_{\lceil n/d \rceil}^T \end{bmatrix} = I + V_1 + \cdots + V_{\lceil n/d \rceil}$

Therefore $V_n = I + V_1 + \cdots + V_{\lceil n/d \rceil} = \begin{bmatrix} X_1 + 1 & 0 & \ldots & \ldots \\ 0 & X_2 + 1 & 0 & \ldots \\ \vdots & 0 & \ddots & 0 \\ \vdots & \vdots & 0 & X_d + 1 \end{bmatrix}$. where X_i is the random variable denoting the number of times arm i is played after round d.

Inverting, we get $V_n^{-1} = \begin{bmatrix} 1/(X_1+1) & 0 & \ldots & \ldots \\ 0 & 1/(X_2+1) & 0 & \ldots \\ \vdots & 0 & \ddots & 0 \\ \vdots & \vdots & 0 & 1/(X_d+1) \end{bmatrix}$.

Since the diagonal elements of V_n^{-1} are all less than or equal to 1, $\mathbf{x}^T(V_n^{-1} - I)\mathbf{x}$ will always be less than or equal to 0 for any vector $\mathbf{x}$. Thus,

$$\begin{aligned} \mathbf{x}^T(V_n^{-1} - I)\mathbf{x} &\leq 0 \\ \mathbf{x}^T V_n^{-1} \mathbf{x} &\leq \mathbf{x}^T I \mathbf{x} \\ \|\mathbf{x}\|^2_{V_n^{-1}} &\leq \|\mathbf{x}\|_2^2 \\ \|\mathbf{1}\|^2_{V_n^{-1}} &\leq \|\mathbf{1}\|_2^2 = d \end{aligned}$$

Using this inequality, we get,

$$
\begin{aligned}
\mathbb{E}[\langle\hat{\theta}_n, \mathbf{1}\rangle^2/\|\mathbf{1}\|^2_{V_n^{-1}}] &\geq \mathbb{E}[\langle\hat{\theta}_n, \mathbf{1}\rangle^2/d] \\
&\geq \frac{1}{d}\,\mathbb{E}[\langle\hat{\theta}_n, \mathbf{1}\rangle]^2 && \text{Jensen's Inequality} \\
&= \frac{1}{d}\,\mathbb{E}[\sum_{i=1}^{d}\hat{\theta}_{n,i}]^2 \\
&= \frac{1}{d}\,\mathbb{E}\left[\sum_{i=1}^{d}\frac{\eta_i + \sum_{t=d+1}^{n}\eta_t\mathbf{1}\{i=(t-1)\,\%\,d+1\}\mathbf{1}\{\eta_t>0\}}{1+X_i}\right]^2 && \text{from } \hat{\theta}_n = V_n^{-1}S_n \\
&\geq \frac{1}{d}\,\mathbb{E}\left[\sum_{i=1}^{d}\frac{\eta_i + \sum_{t=d+1}^{n}\eta_t\mathbf{1}\{i=(t-1)\,\%\,d+1\}\mathbf{1}\{\eta_t>0\}}{1}\right]^2 \\
&= \frac{1}{d}\left[\sum_{i=1}^{d}\mathbb{E}[\eta_i] + \mathbb{E}[\sum_{i}\eta_t\mathbf{1}\{\eta_t>0\}]\right]^2 && \text{sum over number of times that arm i was played which itself is a r.v.} \\
&= \frac{1}{d}\left[\sum_{i=1}^{d}\mathbb{E}[X_i]\,\mathbb{E}[\eta_t\mathbf{1}\{\eta_t>0\}]\right]^2 && \text{law of iterated expectations} \\
&= \frac{1}{d}\left[\sum_{i=1}^{d}\frac{\lceil n/d-1\rceil}{2}\,\mathbb{E}[\eta_t\mathbf{1}\{\eta_t>0\}]\right]^2 && \mathbb{E}[X_i] = \frac{\lceil n/d-1\rceil}{2} \\
&\geq \frac{1}{d}\left[\sum_{i=1}^{d}\frac{1}{2}\,\mathbb{E}[\eta_t\mathbf{1}\{\eta_t>0\}]\right]^2 && n \geq 2d \\
&= \frac{1}{d}\left[\sum_{i=1}^{d}c\right]^2 && \text{Let } \frac{1}{2}\,\mathbb{E}[\eta_t\mathbf{1}\{\eta_t>0\}] = c \\
&= \frac{1}{d}c^2d^2 = c^2d
\end{aligned}
$$

Non-parametric bandits

Problem 4.1

Let $\mathcal{F}_{Lip}$ be a set of functions defined over $[0,1]$ such that for each $f \in \mathcal{F}_{Lip}$ we have $f : [0,1] \to [0,1]$ and for every $x, y \in [0,1]$ we have $|f(y) - f(x)| \leq L|y - x|$ for some known $L > 0$. At each round t the player chooses an $x_t \in [0,1]$ and observes a random variable $y_t \in [0,1]$ such that $\mathbb{E}[y_t] = f_*(x_t)$ where $f_* \in \mathcal{F}_{Lip}$. Define the regret of an algorithm after T steps as $R_T = \mathbb{E}[\sum_{t=1}^{T} f_*(x_\star) - f_*(x_t)]$ where $x_\star = \arg\max_{x\in[0,1]} f_*(x)$.

- a) Propose an algorithm, that perhaps uses knowledge of the time horizon T, that achieves $R_T \leq O(T^{2/3})$ regret (Okay to ignore constant, log factors).

From the definition of $\mathcal{F}_{Lip}$, we note that these are always continuous and their derivates (if they exist) are lesser than L. So we should expect them to not vary too much in any small neighbourhood.
I have taken hints from Slivkins lectures [1] to approach this problem. Discretize $[0,1]$ interval into k distinct intervals to convert this into a k-armed bandit problem and then apply UCB algorithm to minimize the regret. This algorithms regret would consist of the regret due to UCB as well as the regret due to discretization. The regret can be computed as

$$
\begin{aligned}
R_T &= \mathbb{E}[\sum_{t=1}^{T} f_*(x_\star) - f_*(x_t)] \\
&= \mathbb{E}[\sum_{t=1}^{T} f_*(x_\star) - f_{D.opt}(x') + f_{D.opt}(x') - f_*(x_t)] && f_{D.opt} \text{ denotes playing the arm that contains } x\star \\
&= \mathbb{E}[\sum_{t=1}^{T} f_*(x_\star) - f_{D.opt}(x')] + \sqrt{kT\log(T)} && \text{UCB finds } f_{D.opt} \text{ with this regret} \\
&= T\frac{L}{k} + \sqrt{kT\log(T)} && \text{regret due to discretization.}
\end{aligned}
$$

Differentiating with respect to k to minimize worst case regret,

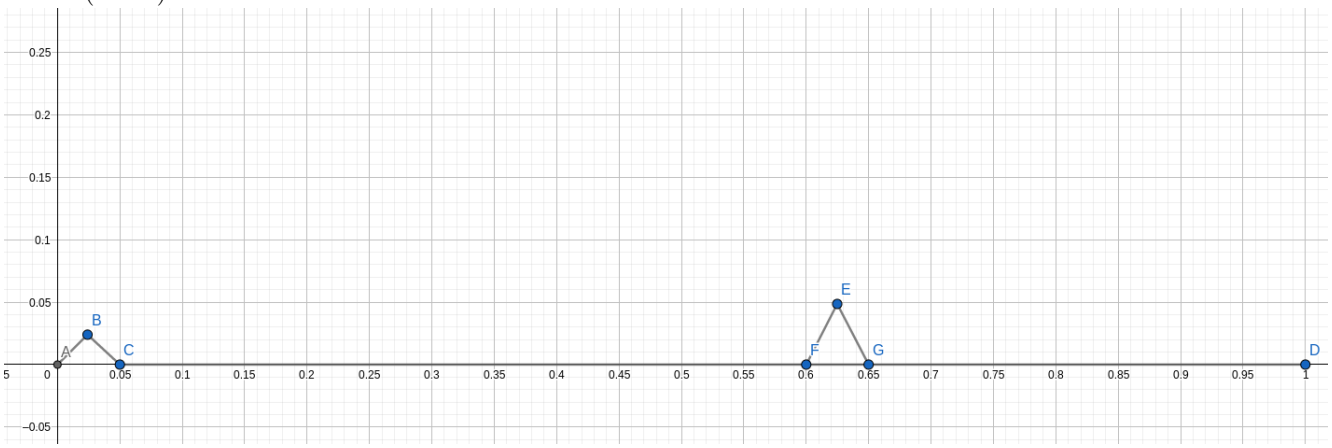
$$\frac{dR_T}{dk} = \sqrt{T\log(T)}\frac{1}{2}k^{-1/2} + TL(-1)k^{-2} = 0$$

Rearranging we get $k = 2L^{2/3}\left(\frac{T}{\log(T)}\right)^{1/3}$.
Substituting this in R_T, we get,

$$
\begin{aligned}
R_T &= \sqrt{\frac{T^{1/3}T\log(T)}{\log(T)^{1/3}}} + \frac{TL}{2L^{2/3}(\frac{T}{\log(T)})^{1/3}} \\
&= T^{2/3}\log(T)^{1/3} + \frac{T^{2/3}L^{1/3}\log(T)^{1/3}}{2} \\
&= O(T^{2/3})
\end{aligned}
$$

- b) Argue that this is minimax optimal (i.e., unimprovable in general through the use of an explicit example, with math, but no formal proof necessary).

  Consider two bandit instances with their functions as shown in the figure below such that $Area(EFG) = 2.Area(ABC)$

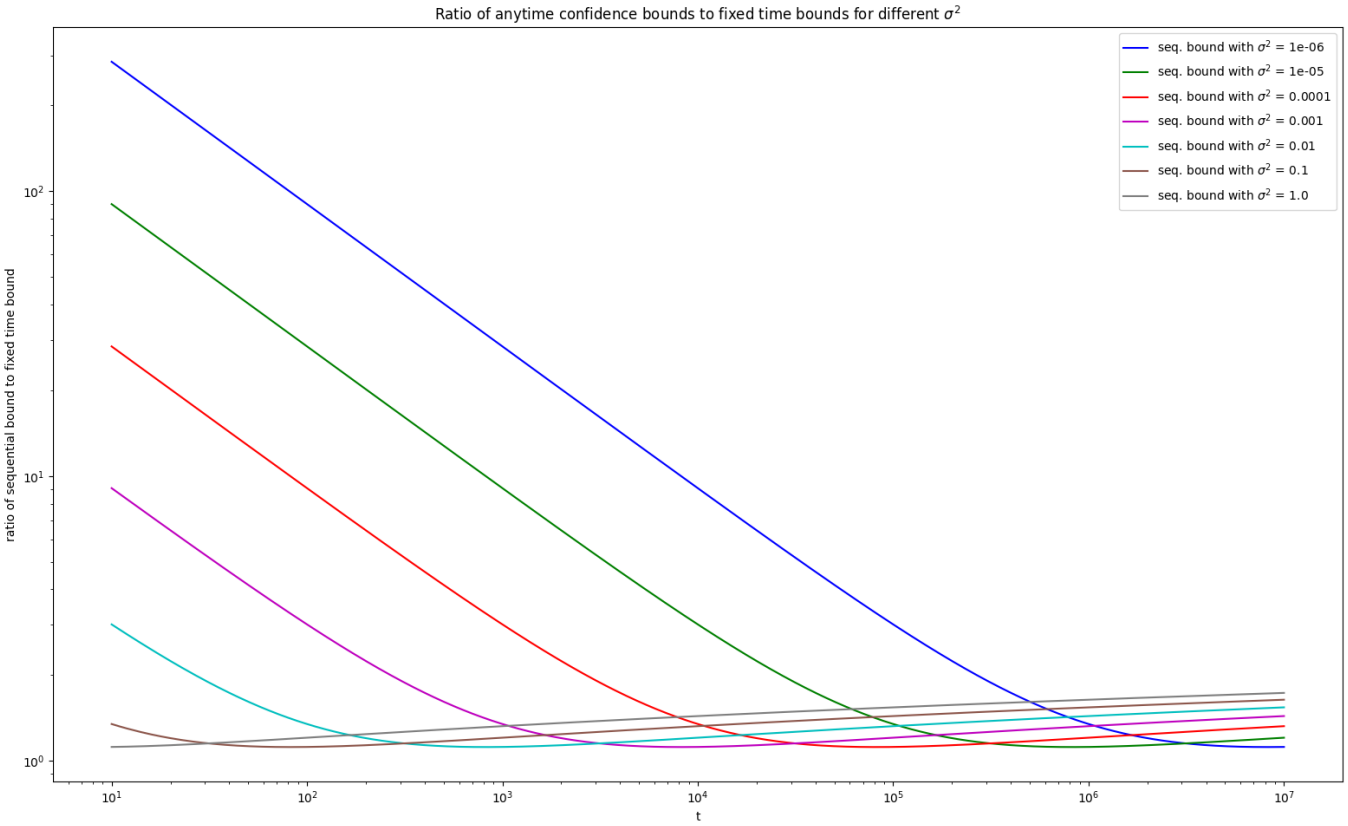


  The first bandit instance has a bump near 0 while the second has a bump near 0.6. In the discretized space, expected value of each arm would be proportional to the area under the bump. For the first bandit instance, reward would be $c.Area(ABC)$ for interval (0, 0.05) and 0 elsewhere. For the second bandit instance, reward would be $2.c.Area(ABC)$ for interval (0.6, 0.65) and 0 elsewhere. The k-armed bandit approach on discretized space has a lower bound on regret of $\Omega(\sqrt{kT})$. Substituting $k = 2L^{2/3}\left(\frac{T}{\log(T)}\right)^{1/3}$ to minimize worst case regret gives a lower bound of $\Omega(T^{2/3})$

Experiments

Experiment 5.1

Here is the plot of the ratio of anytime bound to fixed time bound.



The ratio is smallest with respect to σ^2 at a point $k\sigma^{-2}$ where k is a constant between 5 and 10.
To detect a difference in mean of 0.01, I would choose $\sigma^2 = \Delta^2 = 10^{-4}$ as this is the value of σ that minimizes $n \approx \frac{\Delta^{-2}+\sigma^{-2}}{\sigma^{-2}} \log\left((\frac{\Delta^{-2}+\sigma^{-2}}{\sigma^{-2}})/\delta\right)$.

Experiment 5.2

Strategy chosen is Frank Wolfe. Gradient is calculated for $h = g$. Here's a plot of the mean of 10 samples of $f(\hat{\lambda})$ for each configuration (n, a). The error bars denote 1 standard deviation around the mean.

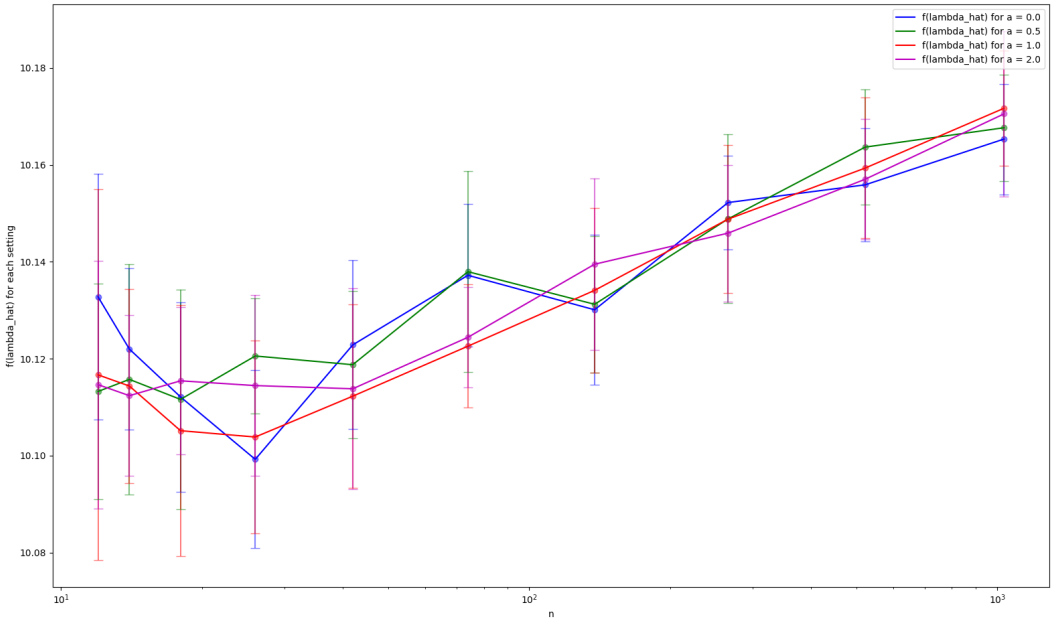
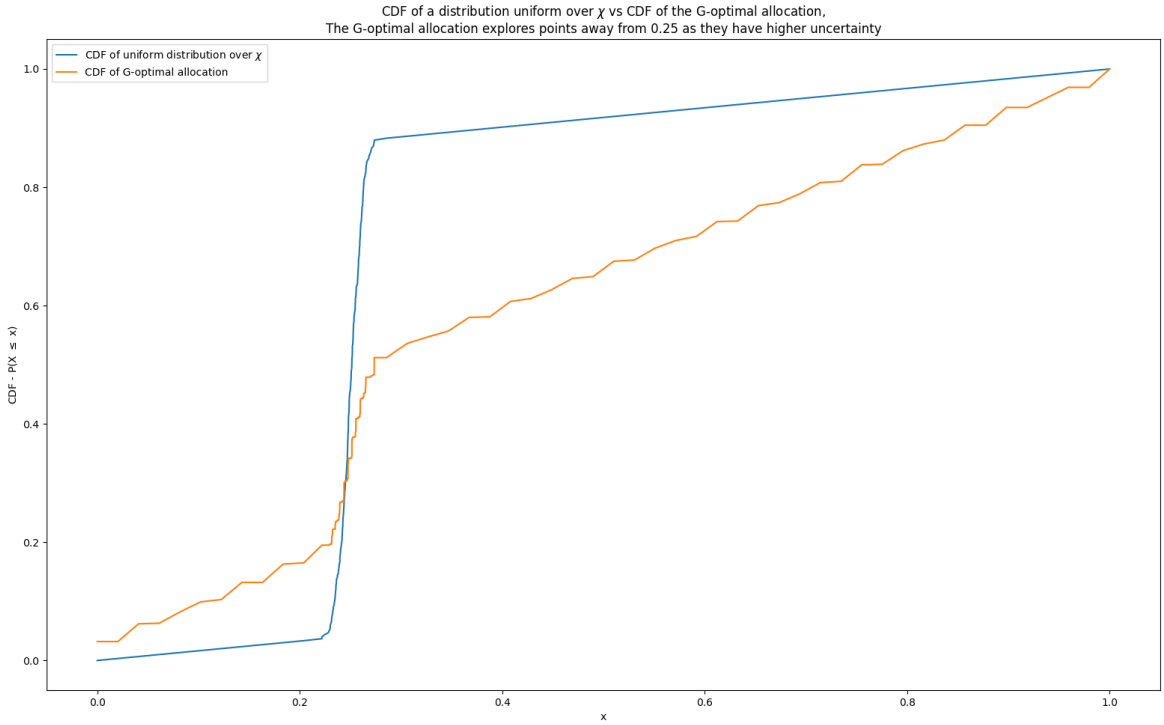


Figure 1: Plot of the mean of 10 samples of $f(\hat{\lambda})$, error bars denote 1 standard deviation around the mean

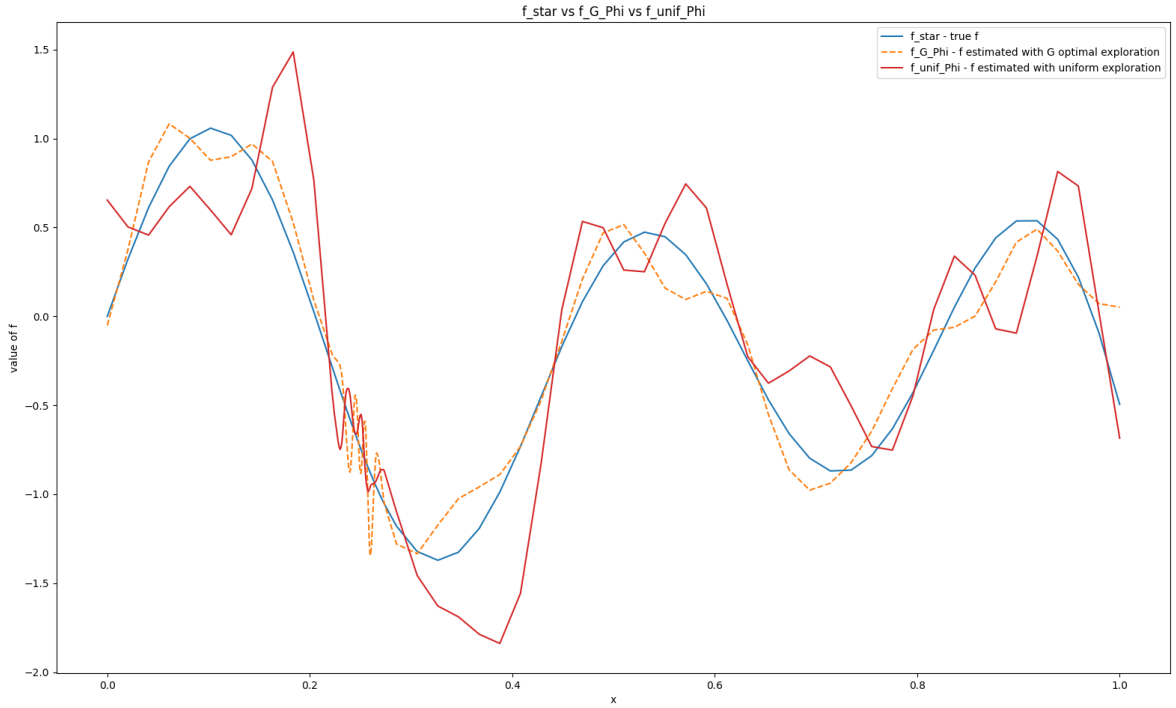
Experiment 5.3

(a) CDF of distribution uniform over χ shows a sharp jump at 0.25 as there are many points in the support near 0.25. Plot of G-optimal allocation shows that it explores points away form 0.25. This should be expected because the higher density of points around 0.25 would convey similar information while those away from 0.25 would carry more information about θ_*



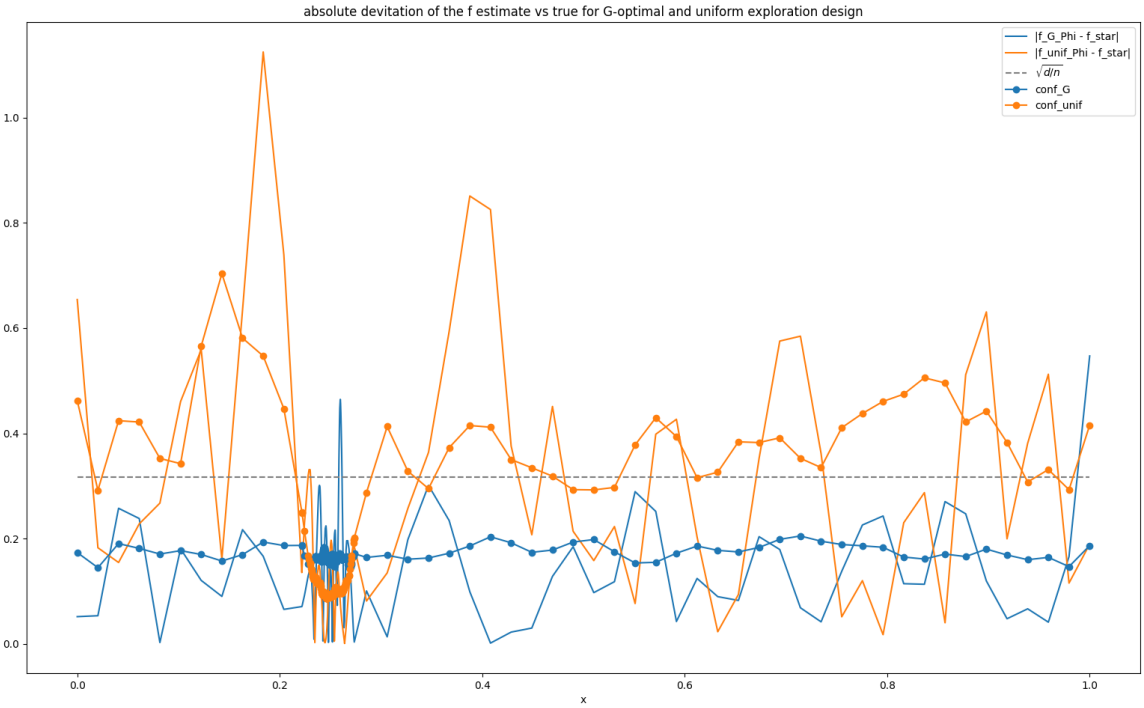
...

(b) The second plot containing f_star, f_G_Phi, f_unif_Phi is shown below.



...

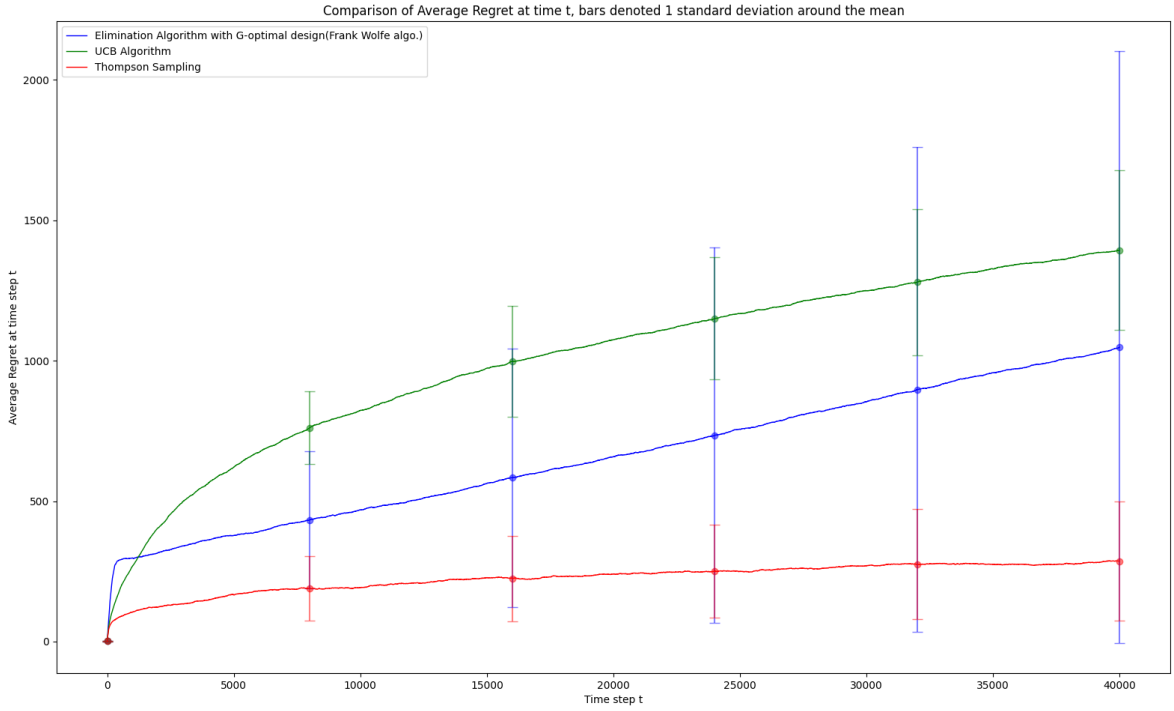
(c) $\sqrt{d/n} = \sqrt{\max_{x \in \chi} \mathbb{E}[\langle x, \hat{\theta} - \theta_* \rangle^2]}$ is where we would expect the maximum deviation to be for G-optimal design. conf is proportional to the uncertainty in the direction of that x. G-optimal has a similar uncertainty in all direction whereas uniform exploration has lower uncertainty near $x = 0.25$ as those points have been explored more often. Correspondingly points further away from 0.25 have higher uncertainty for uniform exploration when compared to G-optimal design.


absolute devitation of the f estimate vs true for G-optimal and uniform exploration design
|f_G_Phi - f_star|
|f_unif_Phi - f_star|
$\sqrt{d/n}$
conf_G
conf_unif
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
x

...

Experiment 5.4

Thompson sampling looks the most suitable if the objective is to minimize regret.



...

References

[1] A. Slivkins. "CMSC 858G: Bandits, Experts and Games". In: (2016). URL: https://www.cs.umd.edu/~slivkins/CMSC858G-fall16/lecture6.pdf.

Code for Experiments

exercise-5-1.py

```
import numpy as np
import matplotlib.pyplot as plt

def fixed_time_tail_bound(delta, t):
    num = 2*np.log(2/delta)
    den = t
    return np.sqrt(num/den)

def sequential_tail_bound(delta, t, var):
```

```
    temp = (1+1/t/var)*(2*np.log(1/delta) + np.log(t*var + 1))/t
    return np.sqrt(temp)

if __name__ == "__main__":
    delta = 0.05  # confidence level
    variances = [1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1e0]

    T_start = 10
    T_end = 10000000
    t_arr = np.arange(T_start, T_end+1, step=5)

    fixed_time_bound = fixed_time_tail_bound(delta, t_arr)

    colors = ['b', 'g', 'r', 'm', 'c', 'tab:brown', 'tab:grey', 'black']

    plt.yscale("log")
    plt.xscale("log")
    #plt.plot(standard_bound, color=colors[0], label='standard bound')
    for i, var in enumerate(variances):
        sequential_bound = sequential_tail_bound(delta, t_arr, var)
        seq_to_standard_ratio = sequential_bound / fixed_time_bound
        plt.plot(t_arr, seq_to_standard_ratio, color=colors[i], label=r"seq. bound with $\sigma^2$"+f" =

    plt.legend(loc='upper right')
    plt.xlabel('t')
    plt.ylabel('ratio of sequential bound to fixed time bound')
    plt.title(r'Ratio of anytime confidence bounds to fixed time bounds for different $\sigma^2$')
    plt.show()
```

g_optimal_design.py

```
import numpy as np
import matplotlib.pyplot as plt

def basis(i, d):
    """
    returns standard basis vector e_i in R^d with 1 at idx i and 0 everywh
    """
    e = np.zeros(d)
    e[i] = 1
    return e

def G_optimal(aaT, lambda_t, X):
    """
    return G-optimal arm - argmax_{aj in X} ||aj||^2_{A(lambda)-1}
```

```
    """
    n, d = X.shape
    A_t = np.sum(np.reshape(lambda_t, (n, 1, 1))*aaT, axis=0)
    A_t_inv = np.linalg.pinv(A_t)
    a2_A_t_inv = np.diagonal(np.matmul(np.matmul(X, A_t_inv), X.T))
    return np.argmax(a2_A_t_inv)

def f(aaT, lambda_hat, X):
    """
    returns value of f(lambda) - max_{aj in X} ||aj||^2_{A(lambda)-1}
    """
    n, d = X.shape
    A_t = np.sum(np.reshape(lambda_hat, (n, 1, 1))*aaT, axis=0)
    A_t_inv = np.linalg.inv(A_t)
    a2_A_t_inv = np.diagonal(np.matmul(np.matmul(X, A_t_inv), X.T))
    return np.max(a2_A_t_inv)

def allocate_frank_wolfe(X, N):
    """
    returns a discrete allocation of size N for G-optimal design using greedy strategy
    """
    n, d = X.shape
    allocation = []

    # play first 2d arms uniformly at random
    #np.random.seed(0)
    initial_I = np.random.randint(0, n, size=2*d)
    for alloc in initial_I: allocation.append(alloc)

    lambda_t = np.zeros(n)
    I_counter = np.zeros(n)
    for idx in initial_I:
        lambda_t[idx]+=1
        I_counter[idx]+=1
    lambda_t = lambda_t/2/d

    aaT = np.zeros((n, d, d))
    for idx in range(n): aaT[idx,:,:] = np.matmul(np.reshape(X[idx], (d, 1)), np.reshape(X[idx], (1, d)))

    for t in range(2*d+1, N+1):
        I_t = G_optimal(aaT, lambda_t, X)
        eta_t = 2/(t+1)
        lambda_t = (1-eta_t)*lambda_t + eta_t*basis(I_t, n)
```

```
            I_counter[I_t]+=1
            allocation.append(I_t)

        lambda_hat = I_counter/N
        return f(aaT, lambda_hat, X), allocation, lambda_hat


if __name__ == "__main__":
    d = 10
    N = 1000
    samples = 10
    a_vals = np.array([0, 0.5, 1, 2])
    j = np.arange(1, d+1)
    n_vals = 10+np.power(2, j)
    variance_vec = np.power(j[np.newaxis,:], -a_vals[:,np.newaxis])
    cov = np.diag(variance_vec[3])
    mean = np.zeros(d)

    f_lambda_hat_mean_array = np.zeros((len(a_vals), len(n_vals)))
    f_lambda_hat_stdev_array = np.zeros((len(a_vals), len(n_vals)))

    for idx_a, a in enumerate(a_vals):
        for idx_n, n in enumerate(n_vals):
            cov = np.diag(variance_vec[idx_a])
            #np.random.seed(0)

            f_lambda_hats = [0]*samples
            for j in range(samples):
                X = np.random.multivariate_normal(mean, cov, size=n)
                f_lambda_hat, allocation, lbda = allocate_frank_wolfe(X, N)
                f_lambda_hats[j] = f_lambda_hat

            f_lambda_hat_mean_array[idx_a, idx_n] = np.mean(f_lambda_hats)
            f_lambda_hat_stdev_array[idx_a, idx_n] = np.std(f_lambda_hats)

    colors = ['b', 'g', 'r', 'm', 'c', 'tab:brown', 'tab:grey', 'black']

    #plt.plot(standard_bound, color=colors[0], label='standard bound')
    plt.xscale("log")
    for i, a_val in enumerate(a_vals):
        plt.plot(n_vals, f_lambda_hat_mean_array[i], color=colors[i], label=f"f(lambda_hat) for a = {a_
        plt.errorbar(n_vals, f_lambda_hat_mean_array[i], f_lambda_hat_stdev_array[i], linestyle='None',
```

```
    plt.legend(loc='upper right')
    plt.xlabel('n')
    plt.ylabel('f(lambda_hat) for each setting')
    plt.show()
```

function_estimation.py

```
import numpy as np
import matplotlib.pyplot as plt
from g_optimal_design import allocate_frank_wolfe

def homework_plot(X, f_star, f_hat, show=True):
    if not show: return
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3)

    # plot 1
    ax1.hist(X)

    # plot 2
    ax2.set_yscale("log")
    ax2.plot(e)

    # plot 3
    ax3.plot(X, f_star)
    ax3.plot(X, f_hat)

    plt.show()

n =300
X = np.concatenate( ( np . linspace (0 ,1 ,50) , 0.25+ 0.01* np . random . randn (250) ) , 0)
X = np.sort( X )

K = np.zeros (( n , n ) )
for i in range( n ):
    for j in range( n ) :
        K[i , j] = 1+ min( X[i], X[j])
e , v = np . linalg . eigh ( K ) # eigenvalues are increasing in order
d = 30
Phi = np.real(v @ np.diag(np.sqrt(np.abs(e))) )[:,(n-d)::]

def f ( x ) :
    return -x **2 + x*np.cos(8*x) + np.sin(15*x)

f_star = f ( X )
```

```
theta = np.linalg.lstsq( Phi, f_star, rcond = None )[0]
f_hat = Phi @ theta

homework_plot(X, f_star, f_hat, show=False)

def observe ( idx ) :
    return f( X[idx]) + np.random.randn(len(idx))

def sample_and_estimate(X, lbda, tau):
    n, d = X.shape
    reg = 1e-6 # we can add a bit of regulari zation to avoid divide by 0
    idx = np.random.choice(np.arange(n), size=tau, p=lbda)
    y = observe(idx)

    XtX = X [ idx ]. T @ X [ idx ]
    XtY = X [ idx ]. T @ y

    theta = np.linalg.lstsq( XtX + reg*np.eye(d), XtY, rcond = None )[0]
    return Phi @ theta, XtX

def plot_func_estimate_1():
    p = 1. * np.arange(len(X)) / (len(X) - 1)
    plt.plot(X, p, color='tab:blue', label=r'CDF of uniform distribution over $\chi$')

    plt.plot(X, np.cumsum(lbda_G), color='tab:orange', label='CDF of G-optimal allocation')

    plt.legend(loc='upper left')
    plt.xlabel('x')
    plt.ylabel(r'CDF - P(X $\leq$ x)')
    plt.title(r'CDF of a distribution uniform over $\chi$ vs CDF of the G-optimal allocation,' + \
        '\nThe G-optimal allocation explores points away from 0.25 as they have higher uncertainty')
    plt.show()
    pass

def plot_func_estimate_2():
    plt.plot(X, f_star, color='tab:blue', label=r'f_star - true f')
    plt.plot(X, f_G_Phi, color='tab:orange', linestyle='dashed', label=r'f_G_Phi - f estimated with G o
    plt.plot(X, f_unif_Phi, color='tab:red', label='f_unif_Phi - f estimated with uniform exploration')

    plt.legend(loc='upper right')
    plt.xlabel('x')
    plt.ylabel(r'value of f')
    plt.title(r'f_star vs f_G_Phi vs f_unif_Phi')
```

```
    plt.show()
    pass

def plot_func_estimate_3():
    plt.plot(X, np.abs(f_G_Phi - f_star), color='tab:blue', label=r'|f_G_Phi - f_star|')
    plt.plot(X, np.abs(f_unif_Phi - f_star), color='tab:orange', label=r'|f_unif_Phi - f_star|')
    plt.plot(X, np.sqrt(d/n)*np.ones_like(X), color='tab:grey', linestyle='dashed', label=r'$\sqrt{d/n}$
    plt.plot(X, conf_G, color='tab:blue', marker='o', label='conf_G')
    plt.plot(X, conf_unif, color='tab:orange', marker='o', label='conf_unif')

    plt.legend(loc='upper right')
    plt.xlabel('x')
    plt.title(r'absolute devitation of the f estimate vs true for G-optimal and uniform exploration des
    plt.show()
    pass


T = 1000

_, allocation, lbda_G = allocate_frank_wolfe( Phi, T )
f_G_Phi , A = sample_and_estimate(Phi, lbda_G, T)
conf_G = np.sqrt(np.sum(Phi @ np.linalg.inv(A) * Phi, axis =1) )

lbda_U = np.ones(n)/n
f_unif_Phi, A = sample_and_estimate(Phi, lbda_U, T)
conf_unif = np.sqrt(np.sum(Phi @ np.linalg.inv(A) * Phi , axis =1) )

plot_func_estimate_1()
plot_func_estimate_2()
plot_func_estimate_3()
```

linear_bandit_algorithms.py

```
import numpy as np
from g_optimal_design import allocate_frank_wolfe
import math

def f(x):
    return -x **2 + x*np.cos(8*x) + np.sin(15*x)

def observe(X, idx) :
    return f(X[idx]) + np.random.randn(len(idx), 1)


def Elimination_algorithm(T, Phi, X):
```

```
tau = 100
delta = 1/T
gamma = 1
U = 1
n, d = Phi.shape
V_0, S_0 = gamma * np.eye(d), np.zeros((d, 1))
Phi_k_hat = Phi
X_k_hat = X
N = 1000

Y = np.zeros(T)
I = np.zeros(T)

V_k, S_k = V_0, S_0
det_V_0 = np.linalg.det(V_0)
for k in range(1, math.floor(T/tau)):
    if Phi_k_hat.shape[0] == 1: break
    # get G-optimal allocation
    _, allocation, lbda_k = allocate_frank_wolfe(Phi_k_hat, N)

    # sample and observe based on the G-optimal allocation
    idx = np.random.choice(np.arange(Phi_k_hat.shape[0]), size=tau, p=lbda_k)
    y = observe(X_k_hat, idx)
    Y[(k-1)*tau:(k-1)*tau+tau] = np.squeeze(y)
    I[(k-1)*tau:(k-1)*tau+tau] = len(Phi_k_hat)

    # update V_k, S_k, theta_k
    V_k = V_k + Phi_k_hat[idx].T @ Phi_k_hat[idx]
    S_k = S_k + np.sum((y.T)*(Phi_k_hat[idx].T), axis=1, keepdims=True)
    V_k_inv = np.linalg.inv(V_k)
    theta_k = V_k_inv @ S_k
    beta_k = np.sqrt(gamma)*U + np.sqrt(2*np.log(1/delta) + np.log( np.abs(np.linalg.det(V_k)/det_V

    # Eliminate arms
    i_max = np.argmax(Phi_k_hat @ theta_k)
    keep_indices = np.squeeze(((Phi_k_hat[[i_max]] - Phi_k_hat) @ theta_k) <= beta_k*\
        np.sum(((Phi_k_hat[[i_max]] - Phi_k_hat) @ V_k_inv) * (Phi_k_hat[[i_max]] - Phi_k_hat), axis
    Phi_k_hat = Phi_k_hat[keep_indices]
    X_k_hat = X_k_hat[keep_indices]

# play remaining arms
rem = T-(k-1)*tau
if rem > 0:
```

```
        I[-rem:] = 1
        y = np.random.randn(rem) + f(X_k_hat[0, 0])
        Y[-rem:] = y
        I[-rem:] = 1

    regret = max(f(X)) - Y
    return np.cumsum(regret), I


def UCB_algorithm(T, Phi, X):
    delta = 1/T
    gamma = 1
    U = 1
    n, d = Phi.shape
    V_0, S_0 = gamma * np.eye(d), np.zeros((d, 1))
    det_V_0 = np.linalg.det(V_0)

    Y = np.zeros(T)
    I = np.zeros(T)

    V_t, S_t = V_0, S_0
    for t in range(T):
        #print(f"iteration {t} of UCB")
        beta_t = np.sqrt(gamma)*U + np.sqrt(2*np.log(1/delta) + np.log( np.abs(np.linalg.det(V_t)/det_V
        V_t_inv = np.linalg.inv(V_t)
        theta_t = V_t_inv @ S_t
        i_t = np.argmax( Phi @ theta_t + beta_t*np.sqrt(np.sum(Phi @ V_t_inv * Phi, axis=1, keepdims=Tr

        # pull arm and observe
        Y[t] = observe(X, [i_t])
        I[t] = i_t

        # update V_t, S_t
        V_t = V_t + Phi[[i_t]].T @ Phi[[i_t]]
        S_t = S_t + Y[t]*(Phi[[i_t]].T)

    regret = max(f(X)) - Y
    return np.cumsum(regret), I


def Thompson_sampling(T, Phi, X):
    gamma = 1
    n, d = Phi.shape
```

```
    V_0, S_0 = gamma * np.eye(d), np.zeros((d, 1))

    Y = np.zeros(T)
    I = np.zeros(T)

    V_t, S_t = V_0, S_0
    for t in range(T):
        V_t_inv = np.linalg.inv(V_t)
        theta_t = V_t_inv @ S_t
        theta_sample = np.random.multivariate_normal(np.squeeze(theta_t), V_t_inv)
        i_t = np.argmax(Phi @ theta_sample)

        # pull arm and observe
        Y[t] = observe(X, [i_t])
        I[t] = i_t

        # update V_t, S_t
        V_t = V_t + Phi[[i_t]].T @ Phi[[i_t]]
        S_t = S_t + Y[t]*(Phi[[i_t]].T)

    regret = max(f(X)) - Y
    return np.cumsum(regret), I
```

simulation.py

```
import numpy as np
from utils import aggregate_regrets, finalize_regrets, plot
from linear_bandit_algorithms import Elimination_algorithm, UCB_algorithm, Thompson_sampling

def generate_features(n, d):
    X = np.concatenate( ( np.linspace(0, 1, 50), 0.25+0.01*np.random.randn(250) ) , 0)
    X = np.sort(X)

    K = np.zeros((n, n))
    for i in range(n):
        for j in range(n) :
            K[i, j] = 1+ min(X[i], X[j])
    e, v = np.linalg.eigh(K) # eigenvalues are increasing in order

    Phi = np.real(v@np.diag(np.sqrt(np.abs(e))))[:,(n-d)::]

    return Phi, np.expand_dims(X, axis=1)

def simulation():
    n_sims = 100
```

```
    T = 40000
    n = 300
    d = 30

    sim_types = [
        ('ELIM', "Elimination Algorithm with G-optimal design(Frank Wolfe algo.)"),
        ('UCB', "UCB Algorithm"),
        ('TS', "Thompson Sampling"),
    ]

    Phi, X = generate_features(n, d)

    mean_result, var_result, labels = [[0 for i in range(len(sim_types))] for _ in range(3)]

    for i, (key, label) in enumerate(sim_types):
        labels[i] = label
        count_aggregate, mean_aggregate, M2_aggregate = [[0 for i in range(T)] for _ in range(3)]
        for n_sim in range(n_sims):
            if n_sim%(n_sims/10)==0: print(f"Running simulation for {key}: {n_sim}/{n_sims}")
            if key == 'ELIM':
                regret, I = Elimination_algorithm(T, Phi, X)
            if key == 'UCB':
                regret, I = UCB_algorithm(T, Phi, X)
            if key == 'TS':
                regret, I = Thompson_sampling(T, Phi, X)
            count_aggregate, mean_aggregate, M2_aggregate = aggregate_regrets(regret, count_aggregate, m

        mean_result[i], var_result[i] = finalize_regrets(count_aggregate, mean_aggregate, M2_aggregate)

    plot(mean_result, var_result, labels, T)

if __name__=="__main__":
    simulation()
```

utils.py

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np


def update(existingAggregate, newValue):
    (count, mean, M2) = existingAggregate
    count += 1
    delta = newValue - mean
```

```
    mean += delta / count
    delta2 = newValue - mean
    M2 += delta * delta2
    return (count, mean, M2)

# Retrieve the mean, variance and sample variance from an aggregate
def finalize(existingAggregate):
    (count, mean, M2) = existingAggregate
    if count < 2:
        return float("nan")
    else:
        (mean, variance, sampleVariance) = (mean, M2 / count, M2 / (count - 1))
        return (mean, variance, sampleVariance)


def aggregate_regrets(regrets, count_aggregate, mean_aggregate, M2_aggregate):
    for i in range(len(regrets)):
        count_aggregate[i], mean_aggregate[i], M2_aggregate[i] = \
            update((count_aggregate[i], mean_aggregate[i], M2_aggregate[i]), regrets[i])

    return count_aggregate, mean_aggregate, M2_aggregate

def finalize_regrets(count_aggregate, mean_aggregate, M2_aggregate):
    variance_aggregate = [0 for i in mean_aggregate]
    for i in range(len(mean_aggregate)):
        mean_aggregate[i], variance_aggregate[i], _ = finalize((count_aggregate[i], mean_aggregate[i],

    return mean_aggregate, variance_aggregate


def plot(mean_aggregate, var_aggregate, labels, T):
    mean_aggregate = [np.array(item) for item in mean_aggregate]
    std_aggregate = [np.sqrt(item) for item in var_aggregate]

    x_error = np.arange(0, T+1, T/5, dtype=np.int32)
    x_error[-1] = x_error[-1]-1

    for i in range(len(mean_aggregate)):
        colors = ['b', 'g', 'r', 'm', 'c', 'k', 'tab:grey']
        y_error = mean_aggregate[i][x_error]
        e = std_aggregate[i][x_error]

        time_series_df = pd.DataFrame(mean_aggregate[i])
```

```
    plt.plot(time_series_df, linewidth=1, label=labels[i], color=colors[i]) #mean curve.
    plt.errorbar(x_error, y_error, e, linestyle='None', fmt='o', color=colors[i], capsize=5, alpha=
plt.legend(loc='upper left')
plt.ylabel("Average Regret at time step t")
plt.xlabel("Time step t")
plt.title("Comparison of Average Regret at time t, bars denoted 1 standard deviation around the mea
plt.rcParams["figure.figsize"] = (30,6)
plt.savefig('results/plot.png')
plt.show()
```