Homework 2

CSE 541: Interactive Learning
Instructor: Kevin Jamieson
Due 11:59 PM on ~~May 11~~ May 18, 2025

Elimination algorithms with infinitely many arms $\mathcal{X}$

1.1 For $x \in \mathbb{R}^d$, Let $B(x, \epsilon)$ denote a Euclidean norm ball with radius ϵ centered at x. Then, given a compact set $\mathcal{X} \subset \mathbb{R}^d$, its ϵ-*covering* is a finite covering set $\mathcal{C} = \{x_1, \dots, x_{N(\epsilon)}\}$ such that $\mathcal{X} \subseteq \bigcup_{i=1}^{N(\epsilon)} B(x_i, \epsilon)$. It is known that if $\max_{x \in \mathcal{X}} \|x\|_2 \le L$ for some $L > 0$, then there exists a covering set for $\mathcal{X}$ such that $N(\epsilon) \le \left(\frac{3L}{\epsilon}\right)^d$. In practice, a greedy set cover works well.

In lecture, we showed that the elimination algorithm can achieve regret $\mathbb{E}[R_T] \le C\sqrt{dT \log(|\mathcal{X}|T)}$ for linear bandits with $|\mathcal{X}|$ arms. Now, suppose that the arm set $\mathcal{X}$ is a compact (bounded) but potentially uncountable set with a bounded 2-norm $L > 0$ and there is a black-box algorithm $\mathcal{A}$ such that the covering set can be obtained by $\mathcal{C} = \mathcal{A}(\mathcal{X}, \epsilon)$. Assume $\|\theta_\star\|_2 \le m$. Then, by using the elimination algorithm and algorithm $\mathcal{A}$ with appropriately chosen ϵ as subroutines, show that for linear bandits with arm set $\mathcal{X}$, it is possible with probability at least $1 - \delta$ to achieve regret

$$R_T \le C\sqrt{d^2 T \log(L/\delta)} + m.$$

Hint: On each stage of the algorithm build an ϵ'-cover so that for each $x \in \mathcal{X}$ we have that there exists an $x_0 \in \mathcal{C}$ such that $|\langle x, \widehat{\theta} - \theta_* \rangle| \le |\langle x_0, \widehat{\theta} - \theta_* \rangle| + |\langle x - x_0, \widehat{\theta} - \theta_* \rangle| \le \max_{x' \in \mathcal{C}} |\langle x', \widehat{\theta} - \theta_* \rangle| + \epsilon'$. Note that if we knew the time horizon T in advance (which you may assume) then you only need to build the cover once.

Linear regression and experimental design

2.1 Exercise 20.2 of [SzepesvariLattimore]

Pure-exploration Linear Bandits

The pure-exploration linear bandits game is as follows:

> **Input:** Finite set $\mathcal{X} \subset \mathbb{R}^d$, $\delta \in (0, 1)$
> **Initialize:** $t = 1$
> **while:** player does not exit
> Player chooses $x_t \in \mathcal{X}$
> Nature reveals $y_t = \langle \theta_*, x_t \rangle + \eta_t$ where $\eta_t \sim \mathcal{N}(0, 1)$
> $t \leftarrow t + 1$
> **Output:** a single arm $\widehat{x} \in \mathcal{X}$

Assume there exists an unique $x_* \in \mathcal{X}$ such that $\langle x_* - x, \theta_* \rangle > 0$ for all $x \in \mathcal{X} \setminus x_*$. There exists an algorithm that outputs an $\widehat{x} \in \mathcal{X}$ such that $\mathbb{P}(\widehat{x} = x_*) \ge 1 - \delta$ after a number of pulls that scales like $\rho_\star \log(1/\delta)$ up to constants and log factors, where

$$\rho_\star = \inf_{\lambda \in \triangle_{\mathcal{X}}} \max_{x \in \mathcal{X}} \frac{\|x - x_\star\|^2_{A(\lambda)^{-1}}}{\langle x_\star - x, \theta_\star \rangle^2},$$

$A(\lambda) = \sum_{x \in \mathcal{X}} \lambda_x x x^\top$, and the infimum is over probability distributions over $\mathcal{X}$ [1]. This is known to be optimal up to log factors.

3.1 Consider the standard multi-armed bandit game where $\mathcal{X} = \{\mathbf{e}_i : i \in [n]\}$. Assume that $\theta_{*,1} > \theta_{*,2} \ge \cdots \ge \theta_{*,n}$. Show that there exists universal positive constants c_1, c_2 such that $c_1 \rho_\star \le \sum_{i=2}^n (\theta_{*,1} - \theta_{*,i})^{-2} \le c_2 \rho_\star$.

Non-parametric bandits

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 $|f(y) - f(x)| \le L|y - x|$ 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 $R_T = \mathbb{E}\left[\sum_{t=1}^T f_*(x_\star) - f_*(x_t)\right]$ where $x_\star = \arg\max_{x \in [0,1]} f_*(x)$.

- 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).
- 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).

Experiments

5.1 Suppose we have random variables $Z_1, Z_2, \ldots$ that are iid with $\mathbb{E}[\exp(\lambda(Z_1 - \mathbb{E}[Z_1])] \leq \exp(\lambda^2/2)$. Then for any *fixed* $t \in \mathbb{N}$ we have the standard tail bound

$$\mathbb{P}\left(|\frac{1}{t}\sum_{s=1}^{t}(Z_s - \mathbb{E}[Z_s])| > \sqrt{\frac{2\log(2/\delta)}{t}}\right) \leq \delta. \tag{1}$$

On the other hand, as we showed in class, for any $\sigma > 0$ we have that

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

which is a confidence bound that holds *for all* $t \in \mathbb{N}$ simultaneously (i.e., not for just a fixed t) at the cost of a slightly inflated bound. Note that because σ must be chosen in advance, it cannot depend on t, and thus there is not a universally good choice. Let $\delta = 0.05$. Plot the *ratio* of the confidence bound of (2) to (1) as a function of t for values of $\sigma^2 \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$. What do you notice about where the ratio is smallest with respect to σ^2 (if the smallest value is at the edge, your t is not big enough–it should be in the many millions)?

Suppose you are observing a stream of iid Gaussian random variables $Z_1, Z_2, \ldots$ such that $Z_t \sim \mathcal{N}(\mu, 1)$. An oracle has told you that $|\mu| = 10^{-2}$, and your job is to design an estimator using (2) to determine the sign using as few observations as possible. How should you choose σ? If you're not sure, consider about how many samples you'd expect this test to take before outputting an answer. Given this guess of the stopping time, how should can you choose σ to potentially minimize this stopping time?

5.2 **G-optimal design** This problem addresses finding a G-optimal design. Fix $(x_1, \ldots, x_n) \subset \mathbb{R}^d$. For any matrix A, let $f(A) = \max_{i=1,\ldots,n} \|x_i\|^2_{A^{-1}}$ and $g(A) = -\log(|A|)$ (these are the G and D optimal designs, respectively). We wish to find a discrete allocation of size N for G-optimal design. Below are strategies to output an allocation $(I_1, \ldots, I_N) \in [n]$. Choose **one** strategy and **one** $h \in \{f, g\}$ to obtain samples from a G-optimal design

1. **Greedy** For $i = 1, \ldots, 2d$ select I_i uniformly at random with replacement from $[n]$. For $t = 2d+1, \ldots, N$ select $I_t = \arg\min_{k\in[n]} h(x_k x_k^\top + \sum_{j=1}^{t-1} x_{I_j} x_{I_j}^\top)$. Return $(I_1, \ldots, I_N)$.
2. **Frank Wolfe** For $i = 1, \ldots, 2d$ select I_i uniformly at random with replacement from $[n]$ and let $\lambda^{(2d)} = \frac{1}{2d}\sum_{i=1}^{2d} \mathbf{e}_{I_i}$. For $k = 2d+1, \ldots, N$ select $I_k = \arg\min_{j\in[n]} \frac{\partial h(\lambda)}{\partial \lambda_j}|_{\lambda=\lambda^{(k-1)}}$ and set $\lambda^{(k)} = (1-\eta_k)\lambda^{(k-1)} + \eta_k \mathbf{e}_{I_k}$ where $\eta_k = 2/(k+1)$. Return $(I_1, \ldots, I_N)$. You will need to derive the partial gradients $\frac{\partial h(\lambda)}{\partial \lambda_i}$. Hint[1].

Let $\widehat{\lambda} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{e}_{I_i}$. We are going to evaluate $f(\widehat{\lambda})$ in a variety of settings. For $a \geq 0$ and $n \in \mathbb{N}$ let $x_i \sim \mathcal{N}(0, \mathrm{diag}(\sigma^2))$ with $\sigma_j^2 = j^{-a}$ for $j = 1, \ldots, d$ with $d = 10$. On a single plot with $n \in \{10 + 2^i\}_{i=1}^{10}$ on the x-axis, plot your chosen method for $a \in \{0, 0.5, 1, 2\}$ as separate lines with $N = 1000$.

5.3 **Experimental design for function estimation**.

Let $f : [0,1]^d \to \mathbb{R}$ be some unknown function and fix some locations of interest $\mathcal{X} = \{x_i\}_{i=1}^n$. We wish to output an estimate $\widehat{f}$ of f uniformly well over $\mathcal{X}$ in the sense that $\max_{x\in\mathcal{X}} \mathbb{E}[(\widehat{f}(x) - f(x))^2]$ is small. To build such an estimate, we can make $N = 1000$ measurements. If we measure the function at location $X \in \mathcal{X}$ we assume we observe $Y = f(X) + \eta$ where $\eta \sim \mathcal{N}(0,1)$. While you can choose any N measurement

[1] $I = A(\lambda)A(\lambda)^{-1}$. Thus, $0 = \frac{\partial}{\partial\lambda_i} I = \left(\frac{\partial}{\partial\lambda_i} A(\lambda)\right) \cdot A(\lambda)^{-1} + A(\lambda) \cdot \left(\frac{\partial}{\partial\lambda_i} A(\lambda)^{-1}\right)$. $\frac{d}{dX}\log(|X|) = (A^{-1})^T$.

locations in X you'd like (with repeats) you have to choose all locations before the observations are revealed. This problem emphasizes that even though we're studying linear functions and linear bandits in class, this can encode incredibly rich functions using non-linear transformations.

1. **Linear functions** Assume $f(x) = \langle x, \theta_* \rangle$ for some $\theta_* \in \mathbb{R}^d$. Propose a strategy to select your N measurements and fit $\widehat{f}$. What guarantees can you make about $\max_{x \in \mathcal{X}} \mathbb{E}[(\widehat{f}(x) - f(x))^2]$? What about the random quantity $\max_{x \in \mathcal{X}} (\widehat{f}(x) - f(x))^2$?

2. **Sobolev functions** For simplicity, let $d = 1$ and assume $f(x)$ is absolutely continuous and $\int_0^1 |f'(x)|^2 dx < \infty$. This class of functions is known as the First-order Sobolev space[2]. Remarkably, this set of functions is equivalent to a reproducing kernel Hilbert space (RKHS) for the kernel $k(x, x') = 1 + \min\{x, x'\}$. While a discussion of all its properties are beyond the scope of this class (see [2, Ch.12] for an excellent treatment) it follows that there exists a sequence of orthogonal functions $\phi(x) := (\phi_1(x), \phi_2(x), \phi_3(x), \dots)$ such that each $\phi_k : [0, 1] \to \mathbb{R}$ and

   - for all $i, j \in \mathbb{N}$ we have $\int_0^1 \phi_i(x)\phi_j(x)dx = \beta_i \mathbf{1}\{i = j\}$ for a non-increasing sequence $\{\beta_i\}_i$
   - $k(x, x') = \langle \phi(x), \phi(x') \rangle := \sum_{k=1}^{\infty} \phi(x)_k \phi_k(x')$,
   - and any function f in this class can be written as $f(x) = \sum_{k=1}^{\infty} \alpha_k \phi_k(x)$ with $\sum_{k=1}^{\infty} \alpha_k^2/\beta_k < \infty$.

   Because $\phi(x)$ could be infinite dimensional, its not immediately clear why this is convenient. For finite datasets though (i.e., $n < \infty$) there always exists a finite dimensional $\phi : \mathcal{X} \to \mathbb{R}^{|\mathcal{X}|}$ where $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. Precisely, if one computes the matrix $[\mathbf{K}]_{i,j} = k(x_i, x_j)$ and $\mathbf{K} = \sum_{k=1}^{n} v_k v_k^\top e_k$ is the resulting eigenvalue decomposition where $\{e_k\}_k$ is a non-increasing sequence, then for all $i \in [n]$ we have $[\phi_i]_j := [\phi(x_i)]_j := v_{i,j}\sqrt{e_j}$ and there exists a $\theta \in \mathbb{R}^n$ such that

$$f(x_i) = \langle \theta, \phi_i \rangle = \sum_{j=1}^{n} \theta_j v_{i,j} \sqrt{e_j}.$$

   Because the e_j are rapidly going to zero, we usually do not include all n, but cut the sum off at some $d \ll n$ so that $\phi_i \in \mathbb{R}^d$ and $f(x_i) \approx \sum_{j=1}^{d} \theta_j v_{i,j} \sqrt{e_j}$. To summarize, for each $i \in [n]$ we have defined a map $x_i \to \phi_i$ with $\phi_i \in \mathbb{R}^d$ and $f(x) \approx \langle \phi_i, \theta_* \rangle$ for some unknown θ_*. Here is a piece of code that generates our set $\mathcal{X} = \{x_i\}_{i=1}^n \subset [0, 1]$ and these features $\Phi = \{\phi_i\}_{i=1}^n \subset \mathbb{R}^d$:

```
import numpy as np
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)::]
```

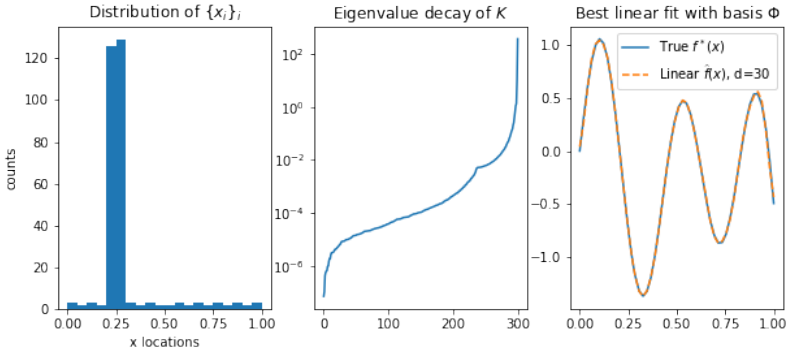
   Let's define some arbitrary function and see how well this works!

```
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
```

[2]This class is quite rich including functions as diverse as all polynomials $a+bx+cx^8$, trigonometric functions $\sin(ax)+\cos(bx)$, exponential functions $e^{ax} + b^x$, and even non-smooth functions like $\max\{1-x, 3x\}$. However, it does not include functions like $1/x$ or $\sqrt{x}$ because their derivatives blow up at $x = 0$



Observe that there is nearly no error in the reconstruction even though we're using a linear model in $\mathbb{R}^{30}$. This is because we defined a very good basis $\Phi \subset \mathbb{R}^d$. We could have achieved the same result by learning in kernel space and adding regularization (this is known as *Bayesian experimental design*). Instead of choosing d we would have to choose the amount of regularization, so there is still always a hyperparameter to choose.

Let's get back to estimating f from noisy samples. Now that we have made this a linear estimation problem, we are exactly in the setting of the first part of this problem. Consider the below listing:

```python
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 regularization 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

T = 1000

lbda = G_optimal(Phi)
f_G_Phi, A = sample_and_estimate(Phi, lbda, T)
conf_G = np.sqrt(np.sum(Phi @ np.linalg.inv(A) * Phi,axis=1))

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

Use your implementation of `G_optimal` from the previous problem and use the `sample_and_estimate` function given above. Your tasks (create a legend for all curves, label all axes, and provide a title for all plots):

(a) Plot 1: x-axis should be the x locations in $[0, 1]$. First line is the CDF of the uniform distribution over $\mathcal{X}$. The second line is the G-optimal allocation over $\mathcal{X}$. Comment on the relative shapes, and how this relates to the distribution over the x's shown in the left-most plot above.

(b) Plot 2: x-axis should be the x locations in $[0, 1]$. Plot `f_star`, `f_G_Phi`, `f_unif_Phi`.

(c) Plot 3: x-axis should be the x locations in $[0, 1]$. First line is the absolute value of `f_G_Phi` minus `f_star`, second line is `f_unif_Phi` minus `f_star`, third line is $\sqrt{d/n}$, fourth line is `conf_G`, fifth line is `conf_unif`. Comment on what these lines have to do with each other.

5.4 **Linear bandits** Implement the elimination algorithm (use your implementation of G-optimal design), UCB, and Thompson sampling (see listings below). Use the precise setup as the previous problem and set

$\mathcal{X} \leftarrow \{\phi_i\}_{i=1}^n$. For $T = 40,000$, on a single plot with $t \in [T]$ on the x-axis, plot $R_t = \max_{x\in\mathcal{X}} \sum_{s=1}^t f(x) - y_t$ with a line for each algorithm. Comment on the results–what algorithm would you recommend to minimize regret?

Elimination algorithm
Input: $T \in \mathbb{N}$
Initialize $\tau = 100$, $\delta = 1/T$, $\gamma = 1$, $U = 1$ (supposed to be an upper bound on $\|\theta_*\|_2$), $V_0 = \gamma I$, $S_0 = 0$, $\widehat{\mathcal{X}}_0 = \mathcal{X}$
for: $k = 1, \ldots, \lfloor T/\tau \rfloor$

$$\lambda^{(k)} = \arg \min_{\lambda \in \triangle_{\widehat{\mathcal{X}}_k}} \max_{x \in \widehat{\mathcal{X}}_k} x^\top \left(\sum_{x' \in \widehat{\mathcal{X}}_k} \lambda_{x'} x' x'^\top \right)^{-1} x$$

Draw $x_{(k-1)\tau+1}, \ldots, x_{k\tau} \sim \lambda^{(k)}$
For $t = (k-1)\tau + 1, \ldots, k\tau$, pull arm x_t and observe $y_t = f(x_t) + \eta_t$ where $\eta_t \sim \mathcal{N}(0,1)$
$V_k = V_{k-1} + \sum_{t=(k-1)\tau+1}^{k\tau} x_t x_t^\top$, $S_k = S_{k-1} + \sum_{t=(k-1)\tau+1}^{k\tau} x_t y_t, \theta_k = V_k^{-1} S_k$
$\beta_k = \sqrt{\gamma} U + \sqrt{2\log(1/\delta) + \log(|V_k|/|V_0|)}$
$\widehat{x}_k = \arg\max_{x' \in \widehat{\mathcal{X}}_k} \langle x', \theta_k \rangle$
$\widehat{\mathcal{X}}_{k+1} = \widehat{\mathcal{X}}_k - \{x \in \widehat{\mathcal{X}}_k : \langle \widehat{x}_k - x, \theta_k \rangle \geq \beta_k \|\widehat{x}_k - x\|_{V_k^{-1}}\}$

UCB algorithm
Input: $T \in \mathbb{N}$
Initialize $\delta = 1/T$, $\gamma = 1$, $U = 1$ (supposed to be an upper bound on $\|\theta_*\|_2$), $V_0 = \gamma I$, $S_0 = 0$
for: $t = 0, 1, 2, \ldots, T-1$
$\beta_t = \sqrt{\gamma} U + \sqrt{2\log(1/\delta) + \log(|V_t|/|V_0|)}$
$\theta_t = V_t^{-1} S_t$
$x_t = \arg\max_{x\in\mathcal{X}} \langle x, \theta_t \rangle + \|x\|_{V_t^{-1}} \beta_t$
Pull arm x_t and observe $y_t = f(x_t) + \eta_t$ where $\eta_t \sim \mathcal{N}(0,1)$
$V_{t+1} = V_t + x_t x_t^\top$, $S_{t+1} = S_t + x_t y_t$

Thompson sampling algorithm
Input: $T \in \mathbb{N}$
Initialize $\gamma = 1$, $V_0 = \gamma I$, $S_0 = 0$
for: $t = 0, 1, 2, \ldots, T-1$
$\theta_t = V_t^{-1} S_t$
$\widetilde{\theta}_t \sim \mathcal{N}(\theta_t, V_t^{-1})$
$x_t = \arg\max_{x\in\mathcal{X}} \langle x, \widetilde{\theta}_t \rangle$
Pull arm x_t and observe $y_t = f(x_t) + \eta_t$ where $\eta_t \sim \mathcal{N}(0,1)$
$V_{t+1} = V_t + x_t x_t^\top$, $S_{t+1} = S_t + x_t y_t$

References

[1] Tanner Fiez, Lalit Jain, Kevin G Jamieson, and Lillian Ratliff. Sequential experimental design for transductive linear bandits. *Advances in neural information processing systems*, 32, 2019.

[2] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.