CSE 541: Interactive Learning - Homework 1

Abhishek Saini

1 Probability

Problem 1.1

(Markov's Inequality) Let X be a positive random variable. Prove that $\mathbb{P}(X > \lambda) \leq \frac{\mathbb{E}[X]}{\lambda}$.

Proof. Here's a proof for the case when X is a continuous random variable. A similar proof for the case when X is discrete can be easily constructed.

$$\begin{aligned}
\mathbb{E}[X] &= \int_{-\infty}^{\infty} x f(x)\, dx \\
&= \int_{0}^{\infty} x f(x)\, dx && X \text{ is a positive random variable} \\
&= \int_{0}^{\lambda} x f(x)\, dx + \int_{\lambda}^{\infty} x f(x)\, dx && \text{for any } \lambda > 0 \\
&\geq \int_{\lambda}^{\infty} x f(x)\, dx \\
&\geq \int_{\lambda}^{\infty} \lambda f(x)\, dx = \lambda\, \mathbb{P}(X > \lambda) \\
\implies \mathbb{P}(X > \lambda) &\leq \frac{\mathbb{E}[X]}{\lambda} && \text{Markov's Inequality}
\end{aligned}$$

□

Problem 1.2

(Jensen's Inequalty) Let X be a random vector in $\mathbb{R}^d$ and let $\phi : \mathbb{R}^d \to \mathbb{R}$ be convex. Then $\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(x)]$ Show this inequality for the special case when X has discrete support. That is, for $p_i \geq 0$ and $\sum_{i=1}^n p_i = 1$, and $(x_1, \ldots, x_n) \subset \mathbb{R}^d$ show that $\phi(\sum_{i=1}^n p_i x_i) \leq \sum_{i=1}^n p_i \phi(x_i)$

Proof. The proof follows by induction. Let $P(n)$ be the predicate that the discrete version of Jensen's inequality holds for any support of size n.

Base Case: $P(2)$ is true because ϕ is convex. For any $(x_1, x_2) \subset \mathbb{R}^d$, and for any p_1, p_2 such that $p_1 \geq 0, p_2 \geq 0$ and $p_1 + p_2 = 1$, since ϕ is convex,

$$\phi(p_1 x_1 + (1 - p_1)x_2) \leq p_1\phi(x_1) + (1 - p_1)\phi(x_2)$$

Thus, $P(2)$ is true. $P(1)$ is trivially true since $p_1 = 1$ in that case.
Inductive Step: Assume the predicate $P(n)$ holds true. Then let's deduce that $P(n+1)$ must also be true.

$$\phi(\sum_{i=1}^{n+1} p_i x_i) = \phi(p_1 x_1 + p_2 x_2 + \cdots + p_{n+1} x_{n+1})$$

If p_1 is 1, then $P(n+1)$ will be trivially true since $p_2, p_3, \ldots, p_{n+1}$ must all be zero. So let's consider the case when $p_1 < 1$. Let's define a new vector y as follows - $y = \frac{1}{1-p_1} \sum_{i=2}^{n+1} p_i x_i$. Since y is a linear combination of $x_2, x_3, ..., x_{n+1}$, $y \in \mathbb{R}^d$. Plugging y into the equation above, we get,

$$\begin{aligned} \phi(\sum_{i=1}^{n+1} p_i x_i) &= \phi(p_1 x_1 + (1 - p_1)y) \\ &\leq p_1\phi(x_1) + (1 - p_1)\phi(y) \qquad \text{since } \phi \text{ is convex} \end{aligned} \tag{1}$$

Next we get a bound on $\phi(y)$.

$$\phi(y) = \phi\left(\sum_{i=2}^{n+1} \frac{p_i x_i}{1 - p_1}\right)$$

Note, we are summing n vectors whose coefficients all sum to 1 and are all greater than or equal to zero. Hence, these coefficients can be valid support of size n for some random variable and we can apply induction hypothesis. Therefore,

$$\phi(y) = \phi\left(\sum_{i=2}^{n+1} \frac{p_i x_i}{1 - p_1}\right) \leq \sum_{i=2}^{n+1} \frac{p_i}{1 - p_1}\phi\left(x_i\right) \qquad \text{By inductive hypothesis}$$

Chugging the above inequality back into (1),

$$\begin{aligned} \phi(\sum_{i=1}^{n+1} p_i x_i) &\leq p_1\phi(x_1) + (1 - p_1)\sum_{i=2}^{n+1} \frac{p_i}{1 - p_1}\phi\left(x_i\right) \\ &= \sum_{i=1}^{n+1} p_i \phi(x_i) \end{aligned}$$

which means $P(n+1)$ is true.
So it follows by induction that $P(n)$ is true for all positive n. $\square$

Problem 1.3

(Sub-additivity of sub-Gaussian) For $i = 1, \ldots, n$ assume X_i is an independent random variable with $\mathbb{E}[\exp(\lambda(X_i - \mathbb{E}[X_i]))] \leq \exp\left(\lambda^2\sigma_i^2/2\right)$. If $Z = \sum_{i=1}^n X_i$ find $a \in \mathbb{R}$ and $b \geq 0$ such that $\mathbb{E}[\exp(\lambda(Z - a))] \leq \exp\left(\lambda^2 b/2\right)$. Consider the random variable $Z' = Z - \sum_{i=1}^n \mathbb{E}[X_i]$

$$\begin{aligned}
\mathbb{E}[\exp\left(\lambda Z'\right)] &= \mathbb{E}[\exp\left(\lambda(Z - \sum_{i=1}^n \mathbb{E}[X_i])\right)] \\
&= \mathbb{E}[\exp\left(\lambda(\sum_{i=1}^n X_i - \sum_{i=1}^n \mathbb{E}[X_i])\right)] \\
&= \mathbb{E}[\exp\left(\lambda \sum_{i=1}^n (X_i - \mathbb{E}[X_i])\right)] \\
&= \mathbb{E}[\prod_{i=1}^n \exp(\lambda(X_i - \mathbb{E}[X_i]))] \\
&= \prod_{i=1}^n \mathbb{E}[\exp(\lambda(X_i - \mathbb{E}[X_i]))] && \text{independence of } X_i \\
&\leq \prod_{i=1}^n \exp\left(\lambda^2\sigma_i^2/2\right) \\
&= \exp\left(\frac{\lambda^2(\sum_{i=1}^n \sigma_i^2)}{2}\right)
\end{aligned}$$

This gives $a = \sum_{i=1}^n \mathbb{E}[X_i]$ and $b = \sum_{i=1}^n \sigma_i^2$

Problem 1.4

(Maximal inequality) For $i = 1, \ldots, n$ let each X_i be an independent, random variable that satisfies $\mathbb{E}[\exp(\lambda X_i)] \leq \exp\left(\lambda^2\sigma_i^2/2\right)$ for all $\lambda > 0$. Show that $E[\max_{i=1\ldots n} X_i] \leq \sqrt{8 \max_{i=1\ldots n} \sigma_i^2 \log(n)}$. If $\sigma_1 \gg \sigma_2 = \cdots = \sigma_n$ how would you expect $E[\max_{i=1\ldots n} X_i]$ to behave (intuitive justification is enough)?

Proof. Applying Jensen's inequality on the identity provided in the hint,

$$
\begin{aligned}
\mathbb{E}[\max_i X_i] &= \frac{1}{\lambda}\log\left(\exp\left(\lambda\,\mathbb{E}[\max_i X_i]\right)\right) && \text{for all } \lambda > 0 \\
&\leq \frac{1}{\lambda}\log\left(\mathbb{E}[\exp\left(\lambda \max_i X_i\right)]\right) && \text{exponential function is convex} \\
& && \text{applying Jensen's inequality} \\
&\leq \frac{1}{\lambda}\log\left(\mathbb{E}\left[\sum_i \exp(\lambda X_i)\right]\right) \\
&= \frac{1}{\lambda}\log\left(\sum_i \mathbb{E}\left[\exp(\lambda X_i)\right]\right) \\
&\leq \frac{1}{\lambda}\log\left(\sum_i \exp\left(\lambda^2\sigma_i^2/2\right)\right) && \text{given in problem definition} \\
&\leq \frac{1}{\lambda}\log\left(\sum_i \exp\left(\lambda^2 \max_i \sigma_i^2/2\right)\right) && \text{bound by the sum of maximum} \\
&= \frac{1}{\lambda}\log\left(n\ \exp\left(\lambda^2 \max_i \sigma_i^2/2\right)\right) \\
&= \frac{1}{\lambda}\left[\log(n) + \log\left(\exp\left(\lambda^2 \max_i \sigma_i^2/2\right)\right)\right] \\
&= \frac{1}{\lambda}\left[\log(n) + \lambda^2 \max_i \sigma_i^2/2\right] \\
&= \frac{\log(n)}{\lambda} + \lambda \max_i \sigma_i^2/2
\end{aligned}
$$

Minimizing $\frac{\log(n)}{\lambda} + \lambda \max_i \sigma_i^2/2$ for any choice of λ by differentiating with respect to λ, we get,

$$
-\frac{\log(n)}{\lambda^2} + \max_i \sigma_i^2/2 = 0
$$

$$
\frac{\log(n)}{\lambda} = \lambda \max_i \sigma_i^2/2 \text{ and } \lambda = \sqrt{\frac{\log(n)}{\max_i \sigma_i^2/2}}
$$

Plugging λ in the inequality above,

$$
\begin{aligned}
\mathbb{E}[\max_i X_i] &\leq \frac{\log(n)}{\lambda} + \lambda \max_i \sigma_i^2/2 \\
&\leq 2\sqrt{\frac{\log(n)}{\max_i \sigma_i^2/2}}\max_i \sigma_i^2/2 \\
&= \sqrt{8\log(n)\max_{i=1\ldots n}\ \sigma_i^2}
\end{aligned}
$$

□

Since all X_i's have mean 0, if $\sigma_i \gg \sigma_2 = \cdots = \sigma_n$, $\max_i X_i$ would be equal to X_1 with high probability whenever $X_1 > 0$ and $\max_i X_i$ would take some small value relative to σ_1 whenever $X_1 \leq 0$. Hence,

$\mathbb{E}[\max_i X_i] \approx \mathbb{E}[X_1|X_1 > 0]\,\mathbb{P}(X_1 > 0)$

2 The Upper Confidence Bound Algorithm

Problem 2.1

Consider the event

$$\mathcal{E} = \bigcap_{i\in[n]} \bigcap_{s\le T} \left(|\hat{\mu}_{i,s} - \mu_i| \le \sqrt{\frac{2\log(2nT^2)}{s}} \right)$$

Show that $\mathbb{P}(\mathcal{E}) \ge 1 - \frac{1}{T}$.

Proof. Consider the event $\mathcal{E}^{\mathsf{c}}$

$$\begin{aligned}
\mathcal{E}^{\mathsf{c}} &= \bigcup_{i\in[n]} \bigcup_{s\le T} \left(|\hat{\mu}_{i,s} - \mu_i| \le \sqrt{\frac{2\log(2nT^2)}{s}} \right)^{\mathsf{c}} \\
&= \bigcup_{i\in[n]} \bigcup_{s\le T} \left(|\hat{\mu}_{i,s} - \mu_i| > \sqrt{\frac{2\log(2nT^2)}{s}} \right)
\end{aligned}$$

Bounding the probability of event $\mathcal{E}^{\mathsf{c}}$ with union bound, we get,

$$\begin{aligned}
\mathbb{P}(\mathcal{E}^{\mathsf{c}}) &= \mathbb{P}\left(\bigcup_{i\in[n]} \bigcup_{s\le T} \left(|\hat{\mu}_{i,s} - \mu_i| > \sqrt{\frac{2\log(2nT^2)}{s}} \right) \right) \\
&\le \sum_{i\in[n]} \sum_{s\le T} \mathbb{P}\left(|\hat{\mu}_{i,s} - \mu_i| > \sqrt{\frac{2\log(2nT^2)}{s}} \right)
\end{aligned}$$

Let's bound each term of the summation by using the two-sided Cramer-Chernoff bound for subgaussian random variables. Note that $\hat{\mu}_{i,s}$ is $\frac{1}{\sqrt{s}}$-subgaussian.

$$\begin{aligned}
\mathbb{P}(\mathcal{E}^{\mathsf{c}}) &\le \sum_{i\in[n]} \sum_{s\le T} 2\exp\left\{ -\frac{s\left(\sqrt{\frac{2\log(2nT^2)}{s}}\right)^2}{2} \right\} \\
&= \sum_{i\in[n]} \sum_{s\le T} 2\exp\{-\log(2nT^2)\} \\
&= \sum_{i\in[n]} \sum_{s\le T} \frac{2}{2nT^2} = \frac{1}{T} \\
\implies \mathbb{P}(\mathcal{E}) &= 1 - \mathbb{P}(\mathcal{E}^{\mathsf{c}}) \ge 1 - \frac{1}{T}
\end{aligned}$$

$\square$

Problem 2.2

On event $\mathcal{E}$ show that $T_i \leq 1 + \frac{8\log(2nT^2)}{\Delta_i^2}$ for $i \neq 1$.

Proof. We have assumed without loss of generality that arm 1 has the highest mean μ_1. Since $\mathcal{E}$ holds, each of the event $\left(|\hat{\mu}_{i,s} - \mu_i| \leq \sqrt{\frac{2\log(2nT^2)}{s}}\right)$ holds for $i \in [n]$ and $s \leq T$.

Lemma 1. *On event $\mathcal{E}$, the upper confidence bound for all arms is always greater than or equal to the mean of that arm.*

Proof. On event $\mathcal{E}$,

$$|\hat{\mu}_{i,s} - \mu_i| \leq \sqrt{\frac{2\log(2nT^2)}{s}}$$
$$\implies \mu_i \leq \hat{\mu}_{i,s} + \sqrt{\frac{2\log(2nT^2)}{s}} = UCB(i)$$

$\square$

Lemma 2. *On event $\mathcal{E}$, if arm $i \neq 1$ gets played at some time step s, then $UCB(i) \geq \mu_1$.*

Proof. Suppose arm $i \neq 1$ gets played but $UCB(i) < \mu_1$. From Lemma 1, $UCB(1) \geq \mu_1$ which would mean that arm i could not have been played in favour of arm 1. $\square$

It's possible that some arm i never gets played after initialization in which case the bound we are trying to prove holds. Suppose arm i does get played after initialization and Let $s = t$ be the last time arm i was played. Since arm i was played, from Lemma 2,

$$\begin{aligned}
UCB(i) &\geq \mu_1 \\
\hat{\mu}_{i,s} + \sqrt{\frac{2\log(2nT^2)}{s}} &\geq \mu_1 \\
\mu_i + \sqrt{\frac{2\log(2nT^2)}{s}} + \sqrt{\frac{2\log(2nT^2)}{s}} &\geq \mu_1 && \text{since event } \mathcal{E} \text{ holds} \\
2\sqrt{\frac{2\log(2nT^2)}{s}} &\geq \mu_1 - \mu_i = \Delta_i \\
\implies s &\leq \frac{8\log(2nT^2)}{\Delta_i^2} \\
\implies T_i = s + 1 &\leq \frac{8\log(2nT^2)}{\Delta_i^2} + 1
\end{aligned}$$

$\square$

Problem 2.3

Show that $\mathbb{E}[T_i] \leq \frac{8\log(2nT^2)}{\Delta_i^2} + 1$

Proof.

$$\begin{aligned}
\mathbb{E}[T_i] &= \mathbb{E}[T_i|\mathcal{E}]\,\mathbb{P}(\mathcal{E}) + \mathbb{E}[T_i|\mathcal{E}^{\mathsf{c}}]\,\mathbb{P}(\mathcal{E}^{\mathsf{c}}) && \text{Law of Total Expectation} \\
&\leq \mathbb{E}[T_i|\mathcal{E}] + \mathbb{E}[T_i|\mathcal{E}^{\mathsf{c}}]\,\mathbb{P}(\mathcal{E}^{\mathsf{c}}) && \mathbb{P}(\mathcal{E}) \leq 1 \\
&\leq \frac{8\log(2nT^2)}{\Delta_i^2} + 1 + \mathbb{E}[T_i|\mathcal{E}^{\mathsf{c}}]\,\mathbb{P}(\mathcal{E}^{\mathsf{c}}) && \text{Problem 2.2} \\
&\leq \frac{8\log(2nT^2)}{\Delta_i^2} + 1 + T\,\mathbb{P}(\mathcal{E}^{\mathsf{c}}) && \mathbb{E}[T_i] \leq T \\
&\leq \frac{8\log(2nT^2)}{\Delta_i^2} + 1 + T\frac{1}{T} && \text{Problem 2.1} \\
\implies \mathbb{E}[T_i] &\leq \frac{8\log(2nT^2)}{\Delta_i^2} + 2
\end{aligned}$$

□

When $n \leq T$, conclude by showing that $R_T \leq \sum_{i=2}^{n} \left(\frac{24\log(2T)}{\Delta_i} + \Delta_i\right)$

Proof.

$$\begin{aligned}
R_T &= \sum_{i=2}^{n} \Delta_i\,\mathbb{E}[T_i] \\
&\leq \sum_{i=2}^{n} \Delta_i \left(\frac{8\log(2nT^2)}{\Delta_i^2} + 2\right) \\
&= \sum_{i=2}^{n} \frac{8\log(2nT^2)}{\Delta_i} + 2\Delta_i \\
&\leq \sum_{i=2}^{n} \frac{8\log(8T^3)}{\Delta_i} + 2\Delta_i && n \leq 4T \\
&= \sum_{i=2}^{n} \frac{24\log(2T)}{\Delta_i} + 2\Delta_i
\end{aligned}$$

□

3 Thompson Sampling

Problem 3.1

On a given run of the algorithm, let $\hat{\theta}_{i,s}$ denote the empirical mean of the first s pulls from arm i, note that $\mathbb{E}[\hat{\theta}_{i,s}] = \theta_i^*$. Let the good event be

$$\mathcal{E} = \bigcap_{i\in[n]} \bigcap_{t\leq T} \left(\left| \hat{\theta}_{i,t} - \theta_i^* \right| \leq \sqrt{\frac{2\log(2/\delta)}{t}} \right)$$

Show that $\mathbb{P}(\mathcal{E}^{\mathsf{c}}) \leq nT\delta$.

Proof. Following a similar strategy as in Problem 2.1

$$\begin{aligned}
\mathbb{P}(\mathcal{E}^{\mathsf{c}}) &= \mathbb{P}\left(\bigcup_{i\in[n]} \bigcup_{t\leq T} \left(\left| \hat{\theta}_{i,t} - \theta_i^* \right| > \sqrt{\frac{2\log(2/\delta)}{t}} \right) \right) \\
&\leq \sum_{i\in[n]} \sum_{t\leq T} \mathbb{P}\left(\left| \hat{\theta}_{i,t} - \theta_i^* \right| > \sqrt{\frac{2\log(2/\delta)}{t}} \right) && \text{Union Bound} \\
&\leq \sum_{i\in[n]} \sum_{t\leq T} 2\exp\left\{ -\frac{t\left(\sqrt{\frac{2\log(2/\delta)}{t}}\right)^2}{2} \right\} && \hat{\theta}_{i,t} \text{ is } 1/\sqrt{t}\text{-subgaussian,} \\
& && \text{two-sided Cramer-Chernoff bound} \\
&= \sum_{i\in[n]} \sum_{t\leq T} 2\exp\{-\log(2/\delta)\} \\
&= \sum_{i\in[n]} \sum_{t\leq T} \delta = nT\delta
\end{aligned}$$

□

Problem 3.2

(Key idea.) Argue that $\mathbb{P}(i^* = \cdot|\mathcal{F}_{t-1}) = \mathbb{P}(I_t = \cdot|\mathcal{F}_{t-1})$

I_t denotes the index of the arm that was pulled at time t. At each time step t, a sample $\theta^{(t)}$ is sampled from the posterior distribution at time $t-1$ denoted by p_{t-1}, except at the first time step when it is sampled from the prior distribution p_0. Hence, $\theta^{(t)}|\mathcal{F}_{t-1}$ is an n-dimensional random vector coming from the distribution p_{t-1}. I_t is the index of the maximum element of this random vector and therefore, $I_t = \arg\max_{i\leq n} \theta^{(t)}$.
i^* denotes the index of the arm that has the highest mean amongst all the arms and is formally denoted as $i^* = \arg\max_i \theta_i^*$. i^* depends on what θ^* was initialized to at the start of the game which the algorithm has no way of knowing since θ^* is a random sample of the n-dimensional prior distribution p_0. The only way the algorithm infers information about θ^* is by observing X_{I_t} and recomputing

the posterior distribution p_t. The most updated belief the algorithm has about θ^* before an arm is pulled at time t is given by p_{t-1}. Formally, this belief is updated by the algorithm after observing the reward using Bayes' theorem

$$\mathbb{P}(\theta^*_{i,t}|X_{I_t}, \mathcal{F}_{t-1}) = \frac{\mathbb{P}(X_{I_t}|\theta^*_{i,t-1}, \mathcal{F}_{t-1})\ \mathbb{P}(\theta^*_{i,t-1}|\mathcal{F}_{t-1})}{\mathbb{P}(X_{I_t}|\mathcal{F}_{t-1})}$$

Thus, the most updated belief the algorithm has about θ^* right before an arm is pulled at time t is distributed as $\theta^*|\mathcal{F}_{t-1} \sim p_{t-1}$. Since both $\theta^{(t)}|\mathcal{F}_{t-1}$ and $\theta^*|\mathcal{F}_{t-1}$ have the same distribution p_{t-1}, $\mathbb{P}(i^* = \cdot|\mathcal{F}_{t-1}) = \mathbb{P}(I_t = \cdot|\mathcal{F}_{t-1})$.

Problem 3.3

Define $U_t(i) = \min\{1, \hat{\theta}_{i,T_i(t)} + \sqrt{\frac{2\log(2/\delta)}{T_i(t)}}\}$. If $i^* = \arg\max_i \theta^*_i$, show that $\mathbb{E}_{\theta^*\sim p_0}[\mathbb{E}_{I_t}[\theta^*_{i^*} - \theta^*_{I_t}|\mathcal{F}_{t-1}]] = \mathbb{E}_{\theta^*\sim p_0}[\theta^*_{i^*} - U_t(i^*)] + \mathbb{E}_{\theta^*\sim p_0}[\mathbb{E}_{I_t}[U_t(I_t) - \theta^*_{I_t}|\mathcal{F}_{t-1}]]$.

Proof. The key idea from Problem 3.2 implies that for any well defined function f,

$$\mathbb{E}[f(i^*)|\mathcal{F}_{t-1}] = \mathbb{E}[f(I_t)|\mathcal{F}_{t-1}] \tag{1}$$

$$\begin{aligned}
\mathbb{E}_{\theta^*\sim p_0}[\mathbb{E}_{I_t}[\theta^*_{i^*} - \theta^*_{I_t}|\mathcal{F}_{t-1}]] &= \mathbb{E}_{\theta^*\sim p_0}[\mathbb{E}_{I_t}[\theta^*_{i^*} - U_t(I_t) + U_t(I_t) - \theta^*_{I_t}|\mathcal{F}_{t-1}]] \\
&= \mathbb{E}_{\theta^*\sim p_0}[\mathbb{E}_{I_t}[\theta^*_{i^*} - U_t(i^*) + U_t(I_t) - \theta^*_{I_t}|\mathcal{F}_{t-1}]] && \text{From (1)} \\
&= \mathbb{E}_{\theta^*\sim p_0}[\mathbb{E}_{I_t}[\theta^*_{i^*} - U_t(i^*)|\mathcal{F}_{t-1}]] + \mathbb{E}_{\theta^*\sim p_0}[\mathbb{E}_{I_t}[U_t(I_t) - \theta^*_{I_t}|\mathcal{F}_{t-1}]] \\
&= \mathbb{E}_{\theta^*\sim p_0}[\theta^*_{i^*} - U_t(i^*)] + \mathbb{E}_{\theta^*\sim p_0}[\mathbb{E}_{I_t}[U_t(I_t) - \theta^*_{I_t}|\mathcal{F}_{t-1}]]
\end{aligned}$$

□

Conclude that $BR_T = \mathbb{E}_{\theta^*\sim p_0}[\sum_{t=1}^T \theta^*_{i^*} - U_t(i^*) + \sum_{t=1}^T \mathbb{E}_{I_t}[U_t(I_t) - \theta^*_{I_t}|\mathcal{F}_{t-1}]]$

Proof.

$$\begin{aligned}
BR_T &= \mathbb{E}_{\theta^*\sim p_0}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t}\right] \\
&= \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}_{I_t}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t} \mid \mathcal{F}_{t-1}\right]\right] \\
&= \mathbb{E}_{\theta^*\sim p_0}[\sum_{t=1}^T \theta^*_{i^*} - U_t(i^*) + \sum_{t=1}^T \mathbb{E}_{I_t}[U_t(I_t) - \theta^*_{I_t}|\mathcal{F}_{t-1}]] && \text{from previous result}
\end{aligned}$$

□

Problem 3.4

Show that $BR_T \le 4\delta n T^2 + \mathbb{E}[\mathbb{E}[\mathbf{1}\{\mathcal{E}\}(\sum_{t=1}^T U_t(I_t) - \theta^*_{I_t}) \mid \theta^*]] \le O(\delta n T^2 + \sqrt{Tn\log(1/\delta)})$

Proof.

$$
\begin{aligned}
BR_T &= \mathbb{E}_{\theta^*\sim p_0}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t}\right] \\
&= \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t} \mid \theta^*\right]\right] \\
&= \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t} \mid \theta^*, \mathcal{E}^{\mathsf{c}}\right]\mathbb{P}(\mathcal{E}^{\mathsf{c}}|\theta^*) + \mathbb{E}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t} \mid \theta^*, \mathcal{E}\right]\mathbb{P}(\mathcal{E}|\theta^*)\right] && \text{Law of total expectation} \\
&\le \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}\left[\sum_{t=1}^T 2 \mid \theta^*, \mathcal{E}^{\mathsf{c}}\right]\mathbb{P}(\mathcal{E}^{\mathsf{c}}|\theta^*) + \mathbb{E}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t} \mid \theta^*, \mathcal{E}\right]\mathbb{P}(\mathcal{E}|\theta^*)\right] && \theta^*_{i^*}, \theta^*_{I_t} \in [-1,1] \\
&= \mathbb{E}_{\theta^*\sim p_0}\left[2T\,\mathbb{P}(\mathcal{E}^{\mathsf{c}}|\theta^*) + \mathbb{E}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t} \mid \theta^*, \mathcal{E}\right]\mathbb{P}(\mathcal{E}|\theta^*)\right] \\
&\le \mathbb{E}_{\theta^*\sim p_0}\left[2\delta n T^2 + \mathbb{E}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t} \mid \theta^*, \mathcal{E}\right]\mathbb{P}(\mathcal{E}|\theta^*)\right] && \text{Problem 3.1} \\
&= 2\delta n T^2 + \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}\left[\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t} \mid \theta^*, \mathcal{E}\right]\mathbb{P}(\mathcal{E}|\theta^*)\right] \\
&= 2\delta n T^2 + \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}\left[\mathbf{1}\{\mathcal{E}\}\left(\sum_{t=1}^T \theta^*_{i^*} - \theta^*_{I_t}\right) \mid \theta^*\right]\right] && \mathbb{E}[\mathbf{1}\{A\}X] = \mathbb{E}[X|A]\,\mathbb{P}(A) \\
&= 2\delta n T^2 + \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}\left[\mathbf{1}\{\mathcal{E}\}\left(\sum_{t=1}^T \theta^*_{i^*} - U_t(I_t) + U_t(I_t) - \theta^*_{I_t}\right) \mid \theta^*\right]\right] \\
&= 2\delta n T^2 + \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}\left[\mathbf{1}\{\mathcal{E}\}\left(\sum_{t=1}^T \theta^*_{i^*} - U_t(i^*) + U_t(I_t) - \theta^*_{I_t}\right) \mid \theta^*\right]\right] && \text{Problem 3.2}
\end{aligned}
$$

Since only the constants differ from Problem 2, applying Problem 2.2 Lemma 1 implies $\theta^*_{i^*} - U_t(i^*) \le 0$. Therefore,

$$BR_T \le 2\delta n T^2 + \mathbb{E}_{\theta^*\sim p_0}\left[\mathbb{E}\left[\mathbf{1}\{\mathcal{E}\}\left(\sum_{t=1}^T U_t(I_t) - \theta^*_{I_t}\right) \mid \theta^*\right]\right]$$

On event $\mathcal{E}$, $\hat{\theta}_{i,T_i(t)} - \theta^*_i \le \sqrt{\frac{2\log(2/\delta)}{T_i(t)}}$. Also by definition, $U_t(I_t) \le \hat{\theta}_{i,T_i(t)} + \sqrt{\frac{2\log(2/\delta)}{T_i(t)}}$ Combining the two gives,

$$U_t(I_t) - \theta^*_{I_t} \le 2\sqrt{\frac{2\log(2/\delta)}{T_i(t)}}$$

Plugging this in the bound for BR_T gives

$$\begin{aligned} BR_T &\leq 2\delta n T^2 + \mathbb{E}_{\theta^* \sim p_0}\left[\mathbb{E}\left[\mathbf{1}\{\mathcal{E}\}\left(\sum_{t=1}^{T}\sqrt{\frac{8\log(2/\delta)}{T_i(t)}}\right) \mid \theta^*\right]\right] \\ &\leq 2\delta n T^2 + \mathbb{E}_{\theta^* \sim p_0}\left[\mathbb{E}\left[\mathbf{1}\{\mathcal{E}\}\left(\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbf{1}\{I_t = i\}\sqrt{\frac{8\log(2/\delta)}{T_i(t)}}\right) \mid \theta^*\right]\right] \end{aligned}$$

Let's try to bound the sum

$$\begin{aligned} \sum_{t=1}^{T}\sum_{i=1}^{n}\sqrt{\frac{1}{T_i(t)}} &\leq \sum_{i=1}^{n}\int_0^{T_i(T)}\sqrt{\frac{1}{T_i(t)}}\,dT_i(t) \\ &= \sum_{i=1}^{n} 2\sqrt{T_i(T)} \\ &\leq 2\sqrt{\sum_{i=1}^{n} T_i(T)\sum_{i=1}^{n} 1} && \text{Cauchy–Schwarz with } x = (\sqrt{T_1}, \ldots, \sqrt{T_k}), y = (1, \ldots, 1) \\ &= 2\sqrt{Tn} \end{aligned}$$

Using this to bound the regret,

$$\begin{aligned} BR_T &\leq 2\delta n T^2 + \mathbb{E}_{\theta^* \sim p_0}\left[\mathbb{E}\left[\sqrt{32Tn\log(2/\delta)} \mid \theta^*\right]\right] \\ &= 2\delta n T^2 + \sqrt{32Tn\log(2/\delta)} \\ &\leq O(\delta n T^2 + \sqrt{Tn\log(1/\delta)}) \end{aligned}$$

□

Problem 3.5

Make an appropriate choice of δ and state a final regret bound.

Choosing $\delta = 1/T^2$,

$$\begin{aligned} BR_T &\leq 2n + \sqrt{32Tn\log(2T^2)} \\ &\leq O(n + \sqrt{Tn\log(T)}) \end{aligned}$$

4 Empirical Experiments

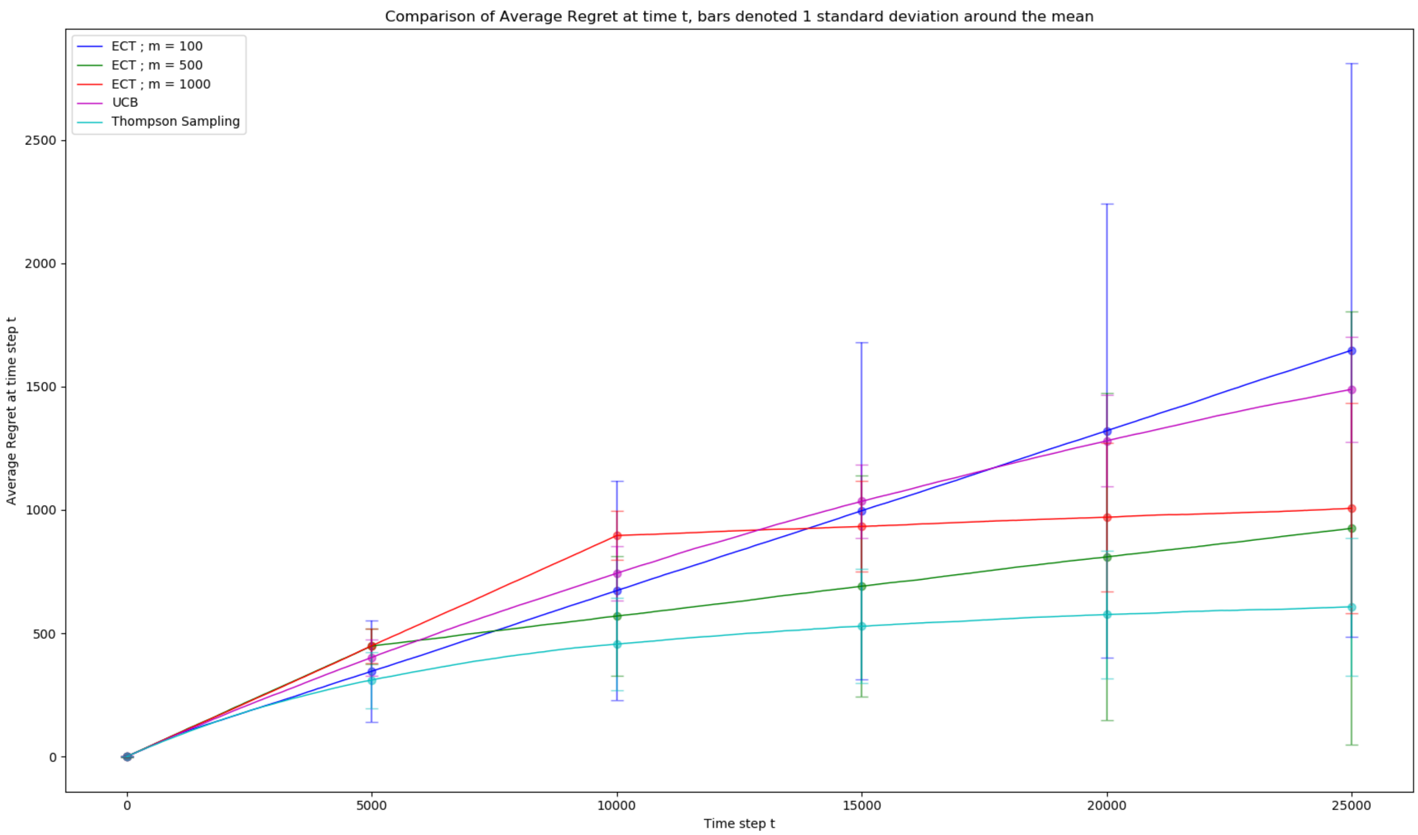
NOTE: Code for section 4 is provided at the end of this assignment.

Section 4.1

Let $n = 10$ and $\mu_1 = 0.1$ and $\mu_i = 0$ for $i > 1$. On a single plot, for an appropriately large T to see expected effects, plot the regret for the UCB, TS, and ETC for several values of m.

Experiment details: Each algorithm was run for 1000 simulations. Each simulation ran for a time horizon $T = 25000$. Plot shows the mean regret at each time step $t \leq T$ and error bars indicate points $\pm 1\sigma$ away from the mean.

Observations: UCB takes a long time to learn optimal arm whereas Thompson Sampling learns much faster than the other algorithms. ETC with $m = 100$ doesn't really learn the optimal strategy whereas with $m = 1000$ it usually recognizes the optimal arm.

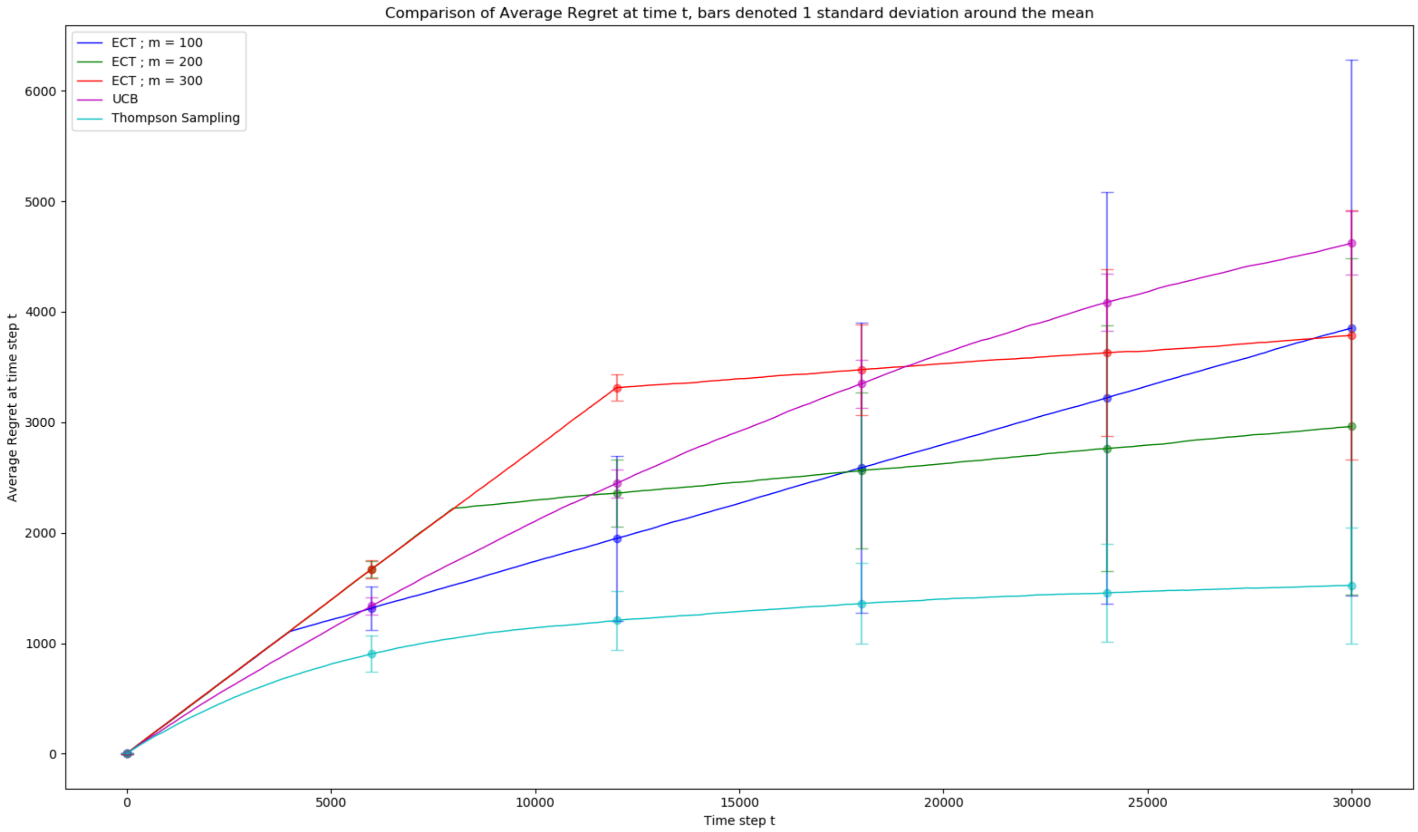


Section 4.2

Let $n = 40$ and $\mu_1 = 1$ and $\mu_i = 1 - 1/\sqrt{i-1}$ for $i > 1$. On a single plot, for an appropriately large T to see expected effects, plot the regret for the UCB, TS, and ETC for several values of m.

Experiment details: Each algorithm was run for 100 simulations. Each simulation ran for a time horizon $T = 25000$. Plot shows the mean regret at each time step $t \leq T$ and error bars indicate

points $\pm 1\sigma$ away from the mean.
Observations: UCB takes a long time to learn optimal arm whereas Thompson Sampling learns much faster than the other algorithms. ETC with $m = 100$ doesn't really learn the optimal strategy whereas with $m = 300$ it usually recognizes the optimal arm.



5 Lower Bounds on Hypothesis Testing

Problem 5.1

Show $\inf_\phi \max \mathbb{P}_0(\phi = 1), \mathbb{P}_1(\phi = 0) \geq \frac{1}{2} \int_{\mathbb{R}^n} \min(p_0(x), p_1(x))\, dx$.

Proof.

$$\begin{aligned}
\inf_\phi \max \mathbb{P}_0(\phi = 1), \mathbb{P}_1(\phi = 0) &\geq \inf_\phi \frac{\mathbb{P}_0(\phi = 1) + \mathbb{P}_1(\phi = 0)}{2} && \text{max is greater than average} \\
&= \inf_\phi \frac{\int_{\mathbb{R}^n} \mathbf{1}\{\phi = 1\}\, d\,\mathbb{P}_0 + \int_{\mathbb{R}^n} \mathbf{1}\{\phi = 0\}\, d\,\mathbb{P}_1}{2} \\
&= \inf_\phi \frac{\int_{\mathbb{R}^n} \mathbf{1}\{\phi = 1\} p_0(x)\, dx + \int_{\mathbb{R}^n} \mathbf{1}\{\phi = 0\} p_1(x)\, dx}{2} && \text{Assuming independence, joint pdf is product of the marginals} \\
&= \inf_\phi \frac{\int_{\mathbb{R}^n} (\mathbf{1}\{\phi = 1\} p_0(x) + \mathbf{1}\{\phi = 0\} p_1(x))\, dx}{2} \\
&\geq \frac{\int_{\mathbb{R}^n} \min(p_0(x), p_1(x))\, dx}{2} && \text{minimized when } \phi \text{ is such that it chooses the min of } p_0(x), p_1(x)
\end{aligned}$$

□

Problem 5.2

Let's continue on. Show $\frac{1}{2} \int_{x \in \mathbb{R}^n} \min(p_0(x), p_1(x))\, dx \geq \frac{1}{4} \left(\int_{x \in \mathbb{R}^n} \sqrt{p_0(x) p_1(x)}\, dx \right)^2$

Proof. Since $ab = \min(a, b) \max(a, b)$, we can write

$$\begin{aligned}
\frac{1}{4} \left(\int_{x \in \mathbb{R}^n} \sqrt{p_0(x) p_1(x)}\, dx \right)^2 &= \frac{1}{4} \left(\int_{x \in \mathbb{R}^n} \sqrt{\min(p_0(x) p_1(x)), \max(p_0(x) p_1(x))}\, dx \right)^2 \\
&\leq \frac{1}{4} \int_{x \in \mathbb{R}^n} \min(p_0(x) p_1(x))\, dx \int_{x \in \mathbb{R}^n} \max(p_0(x) p_1(x))\, dx && \text{Cauchy-Schwarz inequality} \\
&\leq \frac{1}{4} \int_{x \in \mathbb{R}^n} \min(p_0(x) p_1(x))\, dx \int_{x \in \mathbb{R}^n} p_0(x) + p_1(x)\, dx && \text{max is less than sum for +ve terms} \\
&= \frac{1}{2} \int_{x \in \mathbb{R}^n} \min(p_0(x) p_1(x))\, dx && \text{joint pdfs integrate to 1}
\end{aligned}$$

□

Problem 5.3

One more step. Show $\left(\int_{x \in \mathbb{R}^n} \sqrt{p_0(x) p_1(x)}\, dx \right)^2 \geq \exp\left\{ - \int_{x \in \mathbb{R}^n} \log\left(\frac{p_1(x)}{p_0(x)} \right) p_1(x)\, dx \right\}$

Proof.

$$\begin{aligned}
\left(\int_{x\in\mathbb{R}^n} \sqrt{p_0(x)p_1(x)}\,dx\right)^2 &= \exp\left\{\log\left(\int_{x\in\mathbb{R}^n} \sqrt{p_0(x)p_1(x)}\,dx\right)^2\right\} \\
&= \exp\left\{2\log\left(\int_{x\in\mathbb{R}^n} \sqrt{p_0(x)p_1(x)}\,dx\right)\right\} \\
&= \exp\left\{2\log\left(\int_{x\in\mathbb{R}^n} \sqrt{\frac{p_0(x)}{p_1(x)}}p_1(x)\,dx\right)\right\} \\
&\geq \exp\left\{2\int_{x\in\mathbb{R}^n} \log\left(\sqrt{\frac{p_0(x)}{p_1(x)}}\right)p_1(x)\,dx\right\} && \text{Jensen's inequality} \\
&\geq \exp\left\{-\int_{x\in\mathbb{R}^n} \log\left(\frac{p_1(x)}{p_0(x)}\right)p_1(x)\,dx\right\}
\end{aligned}$$

□

Problem 5.4

The final quantity is known as the KL-Divergence between distributions. Now assume that $P_0 = N(\mu_0\mathbf{1}_n, I_n)$ and $P_1 = N(\mu_1\mathbf{1}_n, I_n)$ where I_n is the $n\times n$ identity matrix and $\mathbf{1}_n \in \mathbb{R}^n$ is the all ones vector. Show (or look up) $KL(P_0||P_1)$.

$$
\begin{aligned}
KL(P_0||P_1) &= \int_{x\in\mathbb{R}^n} \log\left(\frac{p_1(x)}{p_0(x)}\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \log\left(\frac{\prod_{j=1}^n \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{(x_i-\mu_1)^2}{2}\right\}}{\prod_{j=1}^n \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{(x_i-\mu_0)^2}{2}\right\}}\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \log\left(\frac{\exp\left\{\sum_{j=1}^n -\frac{(x_i-\mu_1)^2}{2}\right\}}{\exp\left\{\sum_{j=1}^n -\frac{(x_i-\mu_0)^2}{2}\right\}}\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \log\left(\exp\left\{\sum_{j=1}^n \frac{(x_i-\mu_0)^2}{2} - \sum_{j=1}^n \frac{(x_i-\mu_1)^2}{2}\right\}\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \left(\sum_{j=1}^n \frac{(x_i-\mu_0)^2}{2} - \sum_{j=1}^n \frac{(x_i-\mu_1)^2}{2}\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \frac{1}{2}\left(\sum_{j=1}^n (x_i-\mu_0)^2 - (x_i-\mu_1)^2\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \frac{1}{2}\left(\sum_{j=1}^n (x_i-\mu_1+\mu_1-\mu_0)^2 - (x_i-\mu_1)^2\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \frac{1}{2}\left(\sum_{j=1}^n (x_i-\mu_1)^2 + (\mu_1-\mu_0)^2 + 2(x_i-\mu_1)(\mu_1-\mu_0) - (x_i-\mu_1)^2\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \frac{1}{2}\left(\sum_{j=1}^n (\mu_1-\mu_0)^2 + 2(x_i-\mu_1)(\mu_1-\mu_0)\right) p_1(x)\,dx \\
&= \int_{x\in\mathbb{R}^n} \frac{1}{2}\left(n(\mu_1-\mu_0)^2\right) = \frac{n\Delta^2}{2}
\end{aligned}
$$

where $\Delta = \mu_1 - \mu_0$

Problem 5.5

Conclude that to achieve a test that accurately determines whether the sample of size n came from P_0 or P_1 with a probability of error less than δ, we necessarily have $n \geq 2\Delta^{-2}\log(1/4\delta)$ where $\Delta = \mu_1 - \mu_0$.

Proof. From the previous problems, we get the following lower bound on the probability of error

$\delta = \inf_\phi \max \mathbb{P}_0(\phi = 1), \mathbb{P}_1(\phi = 0)$

$$\begin{aligned} \delta &\geq \frac{1}{4}\exp\left\{-\int_{x\in\mathbb{R}^n} \log\left(\frac{p_1(x)}{p_0(x)}\right) p_1(x)\,dx\right\} \\ \implies \delta &\geq \frac{1}{4}\exp\left\{-\frac{n\Delta^2}{2}\right\} && \text{from Problem 5.4} \\ \implies \exp\left\{\frac{n\Delta^2}{2}\right\} &\geq \frac{1}{4\delta} \\ \implies n &\geq \frac{2}{\Delta^2}\log(1/4\delta) \end{aligned}$$

□

Code for section 4

main.py

```python
from simulation import Simulation
from utils import plot
import math

if __name__=="__main__":
    n = 40
    means = [1]
    means.extend([1-1/math.sqrt(i-1) for i in range(2, n+1)])
    T = 30000
    n_sims = 100

    sim_types = [
        ('ECT', "ECT ; m = 100", 100),
        ('ECT', "ECT ; m = 500", 200),
        ('ECT', "ECT ; m = 1000", 300),
        ('UCB', "UCB", None),
        ('TS', "Thompson Sampling", None),
    ]

    mean_aggregate, var_aggregate, labels = [[0 for i in range(len(sim_types))] \
        for _ in range(3)]
```

```
    simulation = Simulation(n, T, means, n_sims)
    for i, (key, label, m) in enumerate(sim_types):
        labels[i] = label
        if key == 'ECT': mean_aggregate[i], var_aggregate[i] = \
            simulation.simulate_ect(m)
        elif key == 'UCB': mean_aggregate[i], var_aggregate[i] = \
            simulation.simulate_ucb()
        elif key == 'TS': mean_aggregate[i], var_aggregate[i] = \
            simulation.simulate_thompson_sampling()
        else:
            print('Invalid Key type')

    plot(mean_aggregate, var_aggregate, labels, T)
```

simulation.py

```
from bandit import Bandit
from algorithms import UCB, ETC, ThompsonSampling
from utils import aggregate_regrets, finalize_regrets

class Simulation(object):
    def __init__(self, n, T, means, n_sims):
        self.n = n
        self.T = T
        self.means = means
        self.n_sims = n_sims

    def simulate_ucb(self):
        count_aggregate, mean_aggregate, M2_aggregate = \
            [[0 for i in range(self.T)] for _ in range(3)]
        for n_sim in range(self.n_sims):
            if n_sim%(self.n_sims/10)==0:
                print(f"Running simulation: {n_sim}/{self.n_sims}")
            bandit = Bandit(self.means)
            ucb_algo = UCB()
            regrets, arm_played, ucb, T_i = ucb_algo.play(self.T, self.means, \
                bandit, 1)
            count_aggregate, mean_aggregate, M2_aggregate = aggregate_regrets(\
                regrets, count_aggregate, mean_aggregate, M2_aggregate)
        mean_aggregate, var_aggregate = finalize_regrets(count_aggregate, \
```

```
            mean_aggregate, M2_aggregate)
        return mean_aggregate, var_aggregate

    def simulate_thompson_sampling(self):
        count_aggregate, mean_aggregate, M2_aggregate = [[0 for i in \
            range(self.T)] for _ in range(3)]
        for n_sim in range(self.n_sims):
            if n_sim%(self.n_sims/10)==0:
                print(f"Running simulation: {n_sim}/{self.n_sims}")
            bandit = Bandit(self.means)
            prior = [(0, 1) for i in range(self.n)]
            ts_algo = ThompsonSampling()
            regrets, arm_played, theta_hat_avgs, T_i = ts_algo.play(\
                self.T, bandit, prior)
            count_aggregate, mean_aggregate, M2_aggregate = aggregate_regrets(\
                regrets, count_aggregate, mean_aggregate, M2_aggregate)
        mean_aggregate, var_aggregate = finalize_regrets(count_aggregate, \
            mean_aggregate, M2_aggregate)
        return mean_aggregate, var_aggregate

    def simulate_ect(self, m):
        count_aggregate, mean_aggregate, M2_aggregate = [[0 for i in \
            range(self.T)] for _ in range(3)]
        for n_sim in range(self.n_sims):
            if n_sim%(self.n_sims/10)==0:
                print(f"Running simulation: {n_sim}/{self.n_sims}")
            bandit = Bandit(self.means)
            etc_algo = ETC()
            regrets, arm_played, ucb, T_i = etc_algo.play(\
                self.T, m, self.means, bandit)
            count_aggregate, mean_aggregate, M2_aggregate = aggregate_regrets(\
                regrets, count_aggregate, mean_aggregate, M2_aggregate)
        mean_aggregate, var_aggregate = finalize_regrets(count_aggregate, \
            mean_aggregate, M2_aggregate)
        return mean_aggregate, var_aggregate
```

algorithms.py

```
import numpy as np
import math
```

```python
class UCB(object):
    def __init__(self):
        pass

    def play(self, T, means, bandit, alpha=1):
        def play_arm(i, t):
            theta_hat_i = bandit.pull_arm(i)
            theta_hat_sums[i] += theta_hat_i
            T_i[i] += 1
            ucb[i] = theta_hat_sums[i]/T_i[i] + \
                alpha*math.sqrt(2*math.log(2*n*T*T)/T_i[i])
            regrets[t] = bandit.get_regret()
            arm_played[t] = i

        n = len(means)
        theta_hat_sums = [0 for i in means]
        T_i = [0 for i in means]
        ucb = [float("inf") for i in means]
        regrets = [0 for t in range(T)]
        arm_played = [0 for t in range(T)]

        for i in range(len(means)):
            play_arm(i, i)

        for t in range(n, T):
            I_t = np.argmax(ucb)
            play_arm(I_t, t)

        return regrets, arm_played, ucb, T_i

class ETC(object):
    def __init__(self):
        pass

    def play(self, T, m, means, bandit):
        def play_arm(i, t):
            theta_hat_i = bandit.pull_arm(i)
```

```python
            theta_hat_sums[i] += theta_hat_i
            T_i[i] += 1
            theta_hat_avgs[i] = theta_hat_sums[i]/T_i[i]
            regrets[t] = bandit.get_regret()
            arm_played[t] = i

        n = len(means)
        theta_hat_sums = [0 for i in means]
        theta_hat_avgs = [0 for i in means]
        T_i = [0 for i in means]
        regrets = [0 for t in range(T)]
        arm_played = [0 for t in range(T)]

        for t in range(T):
            if t<m*n:
                i = t%n
                play_arm(i, t)
            else:
                if t==m*n: I_t = np.argmax(theta_hat_avgs)
                play_arm(I_t, t)

        return regrets, arm_played, theta_hat_avgs, T_i

class ThompsonSampling(object):
    def __init__(self):
        pass

    def sample_theta(self, distribution):
        sample = [np.random.normal(loc=mean, scale=math.sqrt(variance)) \
            for mean, variance in distribution]
        return sample

    def compute_posterior(self, X, mu0, var0):
        var = 1
        var_posterior = var*var0/(var+var0)
        mean_posterior = var_posterior*(mu0/var0 + X/var)
        return (mean_posterior, var_posterior)
```

```
    def play(self, T, bandit, prior):
        def play_arm(i, t):
            theta_hat_i = bandit.pull_arm(i)
            theta_hat_sums[i] += theta_hat_i
            T_i[i] += 1
            theta_hat_avgs[i] = theta_hat_sums[i]/T_i[i]
            regrets[t] = bandit.get_regret()
            arm_played[t] = i
            prior[i] = self.compute_posterior(theta_hat_i, prior[i][0], \
                prior[i][1])

        theta_hat_sums = [0 for i in prior]
        theta_hat_avgs = [0 for i in prior]
        T_i = [0 for i in prior]
        regrets = [0 for t in range(T)]
        arm_played = [0 for t in range(T)]

        for t in range(T):
            sample = self.sample_theta(prior)
            I_t = np.argmax(sample)
            play_arm(I_t, t)

        return regrets, arm_played, theta_hat_avgs, T_i
```

bandit.py

```
import numpy as np

class Bandit(object):
    """
    Implements a K armed Bandit
    """
    def __init__(self, means):
        self.means = means
        self.K = len(means)
        self.optimal_mean = max(means)
        self.regret = 0
        self.last_regret = 0
```

```
    def K(self):
        return self.K

    def pull_arm(self, i):
        X_i = np.random.normal(loc=self.means[i], scale=1)
        self.regret += self.optimal_mean - X_i
        self.last_regret = self.optimal_mean - X_i
        return X_i

    def get_regret(self):
        return self.regret

    def latest_regret(self):
        return self.last_regret
```

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)
```

```python
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], M2_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])
        plt.errorbar(x_error, y_error, e, linestyle='None', fmt='o', \
            color=colors[i], capsize=5, alpha=0.5, barsabove=True)
    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 mean")
    plt.savefig('plot.png')
    plt.show()
```