CSE599i: Online and Adaptive Machine Learning **Winter 2018**

Lecture 10: Contextual Bandits

Lecturer: Lalit Jain *Scribes: Neeraja Abhyankar, Joshua Fan, Kunhui Zhang*

1 Introduction

A natural variation of the multi-armed bandit problem is obtained by associating "side information" (or a context) with each round of play. The rewards corresponding to the arms change at every round, **depending upon this context**. The **contextual bandit** problem has the learner receive the context (which is drawn from a set $\mathcal{C}$) at each time step. The learner pulls an arm according to an algorithm which takes this side-information into consideration, and receives a reward. Since this choice of arm is context-dependent, a notion of contextual regret is introduced where optimality is defined with respect to the best *policy* (i.e., a mapping from contexts to arms) rather than the best *arm*. The space of policies within which the optimum is sought is typically fixed, having some natural structure. A different viewpoint is obtained when the contexts are privately accessed by the policies (which are then appropriately called experts). In this case the contextual information is hidden from the forecaster, and arms must be chosen based only on the past estimated performance of the experts.

2 Example applications

A common application of contextual bandits is to improve user satisfaction by tailoring recommendations to specific user's needs (for example, news articles that the user is likely to be interested in). Traditional methods such as collaborative filtering often do not work well, because content on the web (along with its popularity) changes frequently, and many users have no historical record at all. In this case, the arms would be ads, movies, articles, or whatever is being recommended. The context would be user data and cookies, which can be utilized to predict their preferences. [1]

However, contextual bandits are not restricted to web applications. Suppose a doctor is trying to prescribe the best treatment for a particular patient. In this case, the arms would be the possible treatments, and the context would be the finite set of symptoms the user is experiencing.

Finally, another important application is in mobile health interventions, specifically "Just-In-Time Adaptive Interventions." Mobile health is being used to address behavior change domains including alcohol abuse, depression, obesity, and substance abuse. For many of these, there are a variety of possible intervention actions. For example, in a physical activity setting, the choices (arms) might be whether or not to send a message encouraging activity. The best choice of actions will be influenced by the user's context, including their GPS location, heartrate, and calendar busyness. [2]

3 Notation

The following notation will be used throughout:

K**:** Number of arms

T**:** Total number of time steps

$\mathcal{C}$: The set of contexts (this can be a finite set, or an infinite set such as a vector space $\mathbb{R}^d$). *For example, this could be information about a particular user, such as their browsing history, location, age, etc.*

$\pi : \mathcal{C} \to [K]$: A policy (map from contexts to arms). *For example, a policy could choose an article to recommend, for every possible set of user attributes.*

Π: The class of all of policies

I_t or a_t: The arm chosen at time t. *For example, this could be an article chosen for recommendation.*

r_t: The reward received at time t. *In our example, this is the rating given by user x_t for article I_t.*

$\ell_{i,t}$: The loss incurred by pulling arm i at time t. *The difference in reward w.r.t. the optimum policy.*

4 Problem setting

Assume we have a class of contexts: $\mathcal{C} \subseteq \mathbb{R}^d$ ($\mathcal{C}$ can also be a finite set), and a set of arms $[K]$. For $t = 1, \cdots, T$, each round t proceeds as follows:

- Observe context $x_t \in \mathcal{C}$ (this can be adversarially or randomly chosen).
- Choose arm $I_t \in [K]$. (I_t is the action that the algorithm takes, e.g., an article recommended to the user with features x_t)
- Receive a reward r_t (a function of both the context x_t and the action I_t). Or equivalently, receive loss $\ell_{i,t}$. *Assume that $\ell_{i,t}$ and r_t are bounded within* $[0, 1]$.
- Update the training data set with $\{(r_t, x_t, I_t)\}$ in order to improve future estimates.

In the stochastic bandit setting, $r_t \sim \mathcal{P}(\cdot|x_t, I_t)$ i.i.d for some distribution $\mathcal{P}$. In contrast, r_t may be chosen by an adversary at every round t in an adversarial setting.

5 Aside: Off-policy Evaluation

An important issue in contextual bandit algorithms is evaluating the effectiveness of any particular policy π, even when our data was not generated according to that policy. This problem arises both in practical settings as well as while theoretically evaluating performance guarantees.

- Often, we only have access to existing, logged data – which may be chosen according to an arbitrary policy. This problem naturally occurs where it is not feasible to run experiments for our target policy of interest. In formal terms, the logged data has a fixed distribution $\mathcal{P}_t = \mathcal{P}$ over contexts and rewards (i.e. $x_t \sim \mathcal{P}, r_t \sim \mathcal{P}(\cdot|x_t, a_t)$), but the arms a_t were chosen according to some different (randomized) logging policy p (i.e. $a_t \sim p(\cdot|x_t)$). We wish to estimate what *would have happened* with the presented data, had we used our policy π instead. For this, we of course need to assume *coverage in logging* – $\forall x, a$: if $\pi(a|x) > 0$, then $p(a|x) > 0$.
- In a lot of our theoretical proofs, we want to evaluate the optimality of a policy π_t chosen at time t, over data gathered in the previous rounds, where arms are chosen by other policies. In such scenarios too, we use the technique discussed in this section.

Given n i.i.d. samples $\{(x_s, a_s, r_s)\}_{s=1}^n$ where arms a_s are **chosen according to policy** p, our goal is to estimate the expected reward over the n rounds, **had we played arm** $\pi(x_s)$ at each round. i.e., to estimate $R^\star(x, \pi(x)) = \frac{1}{n}\sum_{s=1}^n \mathbb{E}_{r\sim\mathcal{P}(\cdot|x_s,\pi(x_s))}[r]$. This can be trivially achieved if we could estimate for any fixed context x and for any fixed arm a, the expected reward for that round

$$\mathbb{E}_{r\sim\mathcal{P}(\cdot|a,x)}[r] = R^\star(x, a) \tag{1}$$

The following trick provides an unbiased estimator of this expected reward (given a context and an arm) from our previous observations. Thus, it is commonly used in contextual bandit algorithms:

Theorem 1. *The following Inverse Propensity Score estimator is an unbiased estimator of $R^\star$:*

$$\hat{R}(x_s, a) = r_s \frac{\mathbb{1}_{a_s=a}}{p(a_s|x_s)} \tag{2}$$

(Note that the assumption about coverage under logging ensures that the denominator $p(a_s|x_s)$ is positive.)

Proof. The empirical estimate

$$\begin{aligned}
\mathbb{E}[\hat{R}(x_s,a)|x_s,a] &= \mathbb{E}_{r_s\sim\mathcal{P}}\left[\mathbb{E}_{a_s\sim p(x_s)}\left[r_s\frac{\mathbb{1}_{a_s=a}}{p(a_s|x_s)}\right]\right] && (3)\\
&= \mathbb{E}_{r_s\sim\mathcal{P}}\big[r_s\frac{\mathbb{P}(\pi(x_s)=a_s)}{p(a_s|x_s)}\big] \qquad (\because \text{ expectation of indicator} = \text{probability}) && (4)\\
&= R^\star(x_s,a) \qquad (\text{which by definition has } a_s\sim p, r_s\sim\mathcal{P}) && (5)
\end{aligned}$$

is thus unbiased. □

Intuitively, if we would have actually pulled the arm that policy π would have pulled ($a_s = \pi(x_s)$), we weight the resulting reward by the inverse of $p(a_s|x_s)$, which is the probability that the arm was chosen at time t. If the arm probably would have been chosen anyways, $p(a_s|x_s)$ is large, which downweighs the reward. This makes sense, because our policy should not get much credit for recommending an arm that was likely to be chosen anyways. However, if the policy recommended an arm that was unlikely to be chosen and yet resulted in high reward, then it is more likely to be a good policy.

6 A naive approach: One bandit per context (Finite Context Set)

Compared to a typical multi-armed bandit problem, the only aspect the contextual bandit problem adds is that the learner receives a context at the beginning of the round, before it needs to select its action. When we have **a finite context set** $\mathcal{C}$, one simple approach is to use a separate off-the-shelf bandit algorithm *for each context*, such as Exp3 (Exponential-weight algorithm for Exploration and Exploitation). [3] This results in the following algorithm:

Algorithm 1 $\mathcal{C}$-Exp3 Algorithm

Parameter: a non-increasing sequence of real numbers $(\eta_t)_{t\in\mathbb{N}}$.
For each context $x \in \mathcal{C}$, initialize $p_1^{(x)}$ (context x's distribution over the choice of arms $\{1,\cdots,K\}$ at $t=1$) to be a uniform distribution.
For each round $t = 1,\cdots,T$:

- Receive context $x_t \in \mathcal{C}$.
- **Sample** an action I_t from a previously computed distribution $p_{i,t}^{(x_t)}$, where $p_{i,t}^{(x_t)}$ is the probability (under context x_t) of choosing action $i = 1,\cdots,K$ at time t.
- **For** each arm $i = 1,\cdots,K$, compute the estimated loss $\tilde{\ell}_{i,t}^{(x_t)} = \frac{\ell_{i,t}}{p_{i,t}^{(x_t)}}\mathbb{1}_{I_t=i}$
- **Compute** the new probability distribution over arms, $p_{t+1}^{(x_t)} = (p_{1,t+1}^{(x_t)}, p_{2,t+1}^{(x_t)} \dots p_{K,t+1}^{(x_t)})$, where:

$$p_{i,t+1}^{(x_t)} = \frac{\exp(-\eta_t\sum_{s=1}^t \tilde{\ell}_{i,s}^{(x_t)})}{\sum_{k=1}^K \exp(-\eta_t\sum_{s=1}^t \tilde{\ell}_{k,s}^{(x_t)})}$$

The $\mathcal{C}$-Exp3 algorithm simply applies an instance of Exp3 on each context $x \in \mathcal{C}$. To analyze this algorithm, we define the **pseudo-regret** for the contextual bandit problem as

$$\bar{R}_T = \max_{\pi:\mathcal{C}\to\{1,\cdots,K\}} \mathbb{E}\left[\sum_{t=1}^{T} \ell_{I_t,t} - \sum_{t=1}^{T} \ell_{\pi(x_t),t}\right],$$

where $x_t \in \mathcal{C}$ denotes the context marking the t-th game round. Recall that $\ell_{I_t,t}$ is the loss we receive from pulling arm I_t at time t, and $\ell_{\pi(x_t),t}$ is the loss we receive from pulling the arm that our policy recommends for the current context. Thus, the regret is the difference between the actual loss we incur, and the loss incurred by the best possible mapping from contexts to arms.

Define T_x to be the number of rounds in which the context was x. To derive a bound on the pseudo-regret, we start off by using the pseudo-regret bound for the vanilla Exp3 algorithm for a single context [4]:

Theorem 2. ***(Pseudo-regret of Exp3).*** *If Exp3 is run for every single context x with $\eta_t = \sqrt{\frac{2\ln K}{T_x K}}$, and T_x is the number of rounds in which the context was x, then*

$$\bar{R}_{T_x} \le \sqrt{2T_x K \ln K}$$

Now, we use the regret bound of Exp3 to prove a regret bound for $\mathcal{C}$-Exp3.

Theorem 3. *The pseudo-regret of the $\mathcal{C}$-Exp3 algorithm satisfies*

$$\bar{R}_T \le \sqrt{2T|\mathcal{C}|K\ln K}$$

for any set $\mathcal{C}$ of contexts.

Proof. The $\mathcal{C}$-Exp3 algorithm runs an instance of Exp3 on each context $x_t \in \mathcal{C}$. First, notice that the problem of finding the best map is equivalent to finding the best arm for each context, so we split the pseudo-regret summation over each context.

$$\max_{\pi:\mathcal{C}\to\{1,\cdots,K\}} \mathbb{E}\left[\sum_{t=1}^{T} \ell_{I_t,t} - \sum_{t=1}^{T} \ell_{\pi(x_t),t}\right] = \sum_{x\in\mathcal{C}} \max_{k=1,\cdots,K} \mathbb{E}\left[\sum_{t:x_t=x} (\ell_{I_t,t} - \ell_{k,t})\right]$$

Notice that each term in the outer summation is simply the regret of a single instance of Exp3, for one particular context x. Define T_x to be the number of rounds in which the context was x. Then, we can use the regret bound established in Theorem 2 for each context's instance of Exp3, which had T_x rounds:

$$\begin{aligned}\sum_{x\in\mathcal{C}} \max_{k=1,\cdots,K} \mathbb{E}\left[\sum_{t:x_t=x} (\ell_{I_t,t} - \ell_{k,t})\right] &\le \sum_{x\in\mathcal{C}} \sqrt{2T_x K\ln K}\\ &= |\mathcal{C}|\sqrt{2K\ln K}\sum_{x\in\mathcal{C}} \frac{1}{|\mathcal{C}|}\sqrt{T_x}\\ &\le |\mathcal{C}|\sqrt{2K\ln K}\sqrt{\sum_{x\in\mathcal{C}} \frac{1}{|\mathcal{C}|}T_x}\\ &= \sqrt{|\mathcal{C}|\cdot 2K\ln K\sum_{x\in\mathcal{C}} T_x}\\ &= \sqrt{2T|\mathcal{C}|K\ln K}\end{aligned}$$

Between the second and third line we used Jensen's inequality [5], and on the last line, we used the fact that $T = \sum_x T_x$.

□

At first, this regret seems unsatisfying, since if we simply ignored the context altogether and just ran a single instance of Exp3, by Theorem 2, we incur a regret of $\sqrt{2TK \ln K}$, which – since $|\mathcal{C}| \geq 1$ trivially – seems much better than Theorem 3's result. However, notice that the definition of regret used in both cases is different. In the contextual bandit problem, our goal is to learn the best mapping from contexts to arms. In the case where we ignored the context completely, our set of potential policies is much smaller, namely policies where one arm is *always* selected. However, it is likely that *all* of those policies perform poorly, since they always play the same arm regardless of the context. The "regret" was only low because we were comparing against an extremely poor benchmark. Thus, the $\sqrt{2T|\mathcal{C}|K \ln K}$ regret bound with respect to the set of all general policies is more meaningful.

Finally, since our policy set consists of all mappings from contexts to arms, $|\Pi| = K^{|\mathcal{C}|}$. Note that by the properties of logarithms, Theorem 3's regret bound is equal to $\mathcal{O}(\sqrt{TK \ln(K^{|\mathcal{C}|})}) = \mathcal{O}(\sqrt{TK \ln |\Pi|})$. Therefore, we might conjecture:

Conjecture 1. *If the policy set Π is any finite set, we can achieve $\mathcal{O}(\sqrt{TK \ln |\Pi|})$ regret in a general adversarial setting.*

The Exp4 algorithm in the next section achieves this.

7 The Expert (Adversarial) Case

In the previous section, we viewed each context as an independent bandit problem. However, if the set of possible contexts is large, the $\mathcal{C}$-Exp3 algorithm will not work very well because there will be too little data for each context, and the algorithm treats every context as being completely independent from each other.

Instead, a different approach is to run Exp3 over **policies**. Consider the case where there is a finite set of $|\Pi|$ randomized **policies** (maps from contexts to arms). Consider the setting where we impose only weak constraints on the process generating the data (we allow it to be adversarial), but constrain the comparison class by making it small, say $|\Pi|$. We can even ignore the context X_t, and rely only on the 'expert advice' provided by functions in the comparison class.

This is similar to instances of "prediction with expert advice", so we use the word "expert" to denote a policy. We do not make any assumptions on how the policies (experts) compute their predictions. We will allow the comparison class to be distributions over the K arms. And we will allow the choice of advice to be adversarial.

7.1 One arm per policy

A very simple approach is to simply treat each possible policy (expert) as an arm, and run a single instance of Exp3 over the policies. This would give a regret bound of $\sqrt{T|\Pi| \log |\Pi|}$, by a similar argument as the proof to Theorem 3. However, this regret bound can be poor because the size of the policy set is often exponential in the number of arms. For example, if our policy set is all possible mappings from contexts to arms, there are $|\Pi| = K^{|\mathcal{C}|}$ possible policies. We would like to get rid of the $\sqrt{|\Pi|}$ dependence on the number of policies in the regret bound. Based on our analysis of the $\mathcal{C}$-Exp3 algorithm, we conjectured in Conjecture 1 that we might be able to achieve $\mathcal{O}(\sqrt{TK \ln |\Pi|})$ regret. The Exp4 algorithm, described below, achieves this. [6]

7.2 Exp4

The "Exp3 over policies" approach from the previous section maintains a probability distribution p_t over arms. Similarly, Exp4 ($\underline{\text{Exp}}$onential weights algorithm for $\underline{\text{Exp}}$loration and $\underline{\text{Exp}}$loitation with $\underline{\text{Exp}}$erts), introduced in [4], also maintains a probability distribution q_t over the class of policies/experts Π. As a result, in Exp4, at time t, each expert k's advice is a probability distribution over arms ($\xi_{k,t}$), instead of a single arm. ($\xi_{k,t}$ can be any probability distribution over the K arms.) While "Exp3 over policies" first selects an expert k, and then draws an arm $I_t \in [K]$ from that expert's distribution over arms ($\xi_{k,t}$), Exp4 first mixes the arm predictions of each expert $\xi_{k,t}$ with q_t (the distribution over experts), and then predicts I_t according to the resulting composite distribution over arms, p_t. For each expert k, we use p_t to compute $\tilde{y}_{k,t}$, which is an unbiased estimator of the loss $y_{k,t}$. Then, each expert's contribution is reweighted by the loss incurred by its predictions. The Exp4 algorithm is listed below:

Algorithm 2 Exp4 Algorithm (Exponential weights algorithm for Exploration and Exploitation with Experts)

Input: Set of K arms, set of experts Π.
Parameter: real number η
Let q_1 be the uniform distribution over the experts (policies), $\{1, \cdots, |\Pi|\}$.
For each round $t = 1, \cdots, T$:

- **Receive** expert advice $\xi_{k,t}$ for each expert $k \in \Pi$, where each $\xi_{k,t}$ is a probability distribution over arms.
- **Draw** an arm I_t from the probability distribution $p_t = (p_{1,t}, \cdots, p_{K,t})$, where $p_t = \mathbb{E}_{k \sim q_t} \xi_{k,t}$.
- **Observe** loss $\ell_{I_t,t}$. For each arm i, compute $\tilde{\ell}_{i,t}$, using the Inverse Propensity Score trick in Theorem 1 to obtain an unbiased estimator for the loss of arm i:

$$\tilde{\ell}_{i,t} = \frac{\ell_{i,t}}{p_{i,t}} \mathbb{1}_{I_t = i} \qquad i = 1, \cdots, K$$

- **Compute** the estimated loss for each expert, by taking the expected loss over the expert's predictions.

$$\tilde{y}_{k,t} = \mathbb{E}_{i \sim \xi_{k,t}} \tilde{\ell}_{i,t} = \sum_{i=1}^{K} \xi_{k,t}(i) \tilde{\ell}_{i,t} \qquad k = 1, \cdots, |\Pi|$$

- **Compute** the new probability distribution over the experts $q_{t+1} = (q_{1,t+1}, \cdots, q_{N,t+1})$, where

$$q_{j,t+1} = \frac{\exp(-\eta \sum_{s=1}^{t} \tilde{y}_{k,s})}{\sum_{k=1}^{|\Pi|} \exp(-\eta \sum_{s=1}^{t} \tilde{y}_{k,s})}$$

Theorem 4. ***(Pseudo-regret of Exp4).*** *The regret of Exp4 with* $\eta_t = \sqrt{\frac{\ln |\Pi|}{TK}}$ *is bounded by:*

$$\bar{R}_n \leq \sqrt{2TK \ln |\Pi|}$$

.

Proof. To complete the proof, we need to prove a few equalities. First, the expectation (over our choice of **arms**) of $\tilde{y}_{k,t}$ (our estimate of the loss of expert k), is just the true expected loss for that expert (over the expert's arm distribution), $\mathbb{E}_{i \sim \xi_t^k} \ell_{i,t}$:

Lemma 1. $\mathbb{E}_{I_t \sim p_t} \tilde{y}_{k,t} = \mathbb{E}_{i \sim \xi_{k,t}} \ell_{i,t}$

Proof.

$$\mathbb{E}_{I_t \sim p_t} \tilde{y}_{k,t} = \mathbb{E}_{I_t \sim p_t} \left[\mathbb{E}_{i \sim \xi_{k,t}} \left[\tilde{\ell}_{i,t} \right] \right] = \sum_{j=1}^{K} p_{j,t} \cdot \mathbb{E}_{i \sim \xi_{k,t}} \left[\frac{\ell_{i,t}}{p_{i,t}} \mathbb{1}_{j=i} \right]$$

$$= \mathbb{E}_{i \sim \xi_{k,t}} \left[\sum_{j=1}^{K} p_{j,t} \cdot \frac{\ell_{i,t}}{p_{i,t}} \mathbb{1}_{j=i} \right] = \mathbb{E}_{i \sim \xi_{k,t}} \ell_{i,t}$$

□

The last line follows because of the indicator variable; the only nonzero term in the sum is when $j = i$. Next, we need another equality:

Lemma 2. $\mathbb{E}_{k\sim q_t}\tilde{y}_{k,t}^2 = \frac{\ell_{I_t,t}^2}{p_{I_t,t}}$

Proof.

$$\mathbb{E}_{k\sim q_t}\tilde{y}_{k,t}^2 = \mathbb{E}_{k\sim q_t}\left[\mathbb{E}_{i\sim \xi_{k,t}}\left[\tilde{\ell}_{i,t}\right]^2\right] \le \mathbb{E}_{k\sim q_t}\left[\mathbb{E}_{i\sim \xi_{k,t}}\left[\tilde{\ell}_{i,t}^2\right]\right] = \sum_{k=1}^{|\Pi|} q_{k,t}\sum_{i=1}^{K}(\xi_{k,t})_i \cdot \tilde{\ell}_{i,t}^2$$
$$= \sum_{i=1}^{K}\tilde{\ell}_{i,t}^2\left(\sum_{k=1}^{|\Pi|} q_{k,t}(\xi_{k,t})_i\right) = \sum_{i=1}^{K}\tilde{\ell}_{i,t}^2\mathbb{E}_{k\sim q_t}(\xi_{k,t})_i = \sum_{i=1}^{K}\left(\frac{\ell_{i,t}}{p_{i,t}}\mathbb{1}(I_t = i)\right)^2 \cdot p_{i,t} = \frac{\ell_{I_t,t}^2}{p_{I_t,t}}$$

We used Jensen's inequality in the second step. Again, for the last step, note that the only nonzero term is when $i = I_t$.

□

Now, recall that we proved this regret bound for Exp3 (see page 4 of the Exp3 lecture notes [3]):

Lemma 3. *Suppose that the Exp3 algorithm is run on K arms, and at each time step t, the algorithm's distribution over arms is p_t. Additionally, $\tilde{\ell}_{i,t}$ is the estimated loss, which is defined as $\frac{\mathbb{1}_{I_t=i}}{p_{i,t}}\ell_{i,t}$. Then, the following inequality holds:*

$$\sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\tilde{\ell}_{i,t} - \min_{i\in[K]}\sum_{t=1}^{T}\tilde{\ell}_{i,t} \le \frac{\ln(K)}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\tilde{\ell}_{i,t}^2$$

First plug in the definition of $\tilde{\ell}_{i,t}$, which is $\frac{\mathbb{1}_{I_t=i}}{p_{i,t}}\ell_{i,t}$. Also, note that if the inequality holds for the arm $i \in [K]$ that minimizes the second term on the left side, it must hold for **all** arms i.

$$\sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\frac{\mathbb{1}_{I_t=i}}{p_{i,t}}\ell_{i,t} - \sum_{t=1}^{T}\tilde{\ell}_{i,t} \le \frac{\ln(K)}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\tilde{\ell}_{i,t}^2$$

Again, the only nonzero term in the inner sum in the first term is when $i = I_t$. Also, apply the definition of expectation on the right side.

$$\sum_{t=1}^{T}\ell_{I_t,t} - \sum_{t=1}^{T}\tilde{\ell}_{i,t} \le \frac{\ln(K)}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}_{i\sim p_t}\tilde{\ell}_{i,t}^2$$

To apply this equation to Exp4, note that Exp4 is essentially running Exp3 over experts/policies, where the number of arms (K) is equal to the number of policies ($|\Pi|$), and the distribution over arms/experts (formerly $p_{i,t}$) is now $q_{i,t}$. The estimated loss $\tilde{\ell}_{i,t}$ is now $\tilde{y}_{i,t}$, which is the estimated loss of each expert in the Exp4 algorithm. For **any** fixed expert k, this gives the inequality:

$$\sum_{t=1}^{T}\ell_{I_t,t} - \sum_{t=1}^{T}\tilde{y}_{k,t} \le \frac{\ln|\Pi|}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}_{k\sim q_t}\tilde{y}_{k,t}^2$$

Since the only stochasticity is the arm we play, if we take expectation over the draw of arms $I_1, \cdots, I_T$, and apply Lemmas 1 and 2, then we have

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} \ell_{I_t,t}\right] - \sum_{t=1}^{T} \mathbb{E}_{i\sim\xi_{k,t}}\left[\ell_{i,t}\right] &\leq \frac{\ln|\Pi|}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}_{i\sim p_t}\left[\frac{\ell_{i,t}^2}{p_{i,t}}\right] \\
&= \frac{\ln|\Pi|}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\frac{\ell_{i,t}^2}{p_{i,t}} \\
&\leq \frac{\ln|\Pi|}{\eta} + \frac{\eta}{2}TK
\end{aligned}$$

The last inequality holds because we bounded $\ell_{i,t}$ to be between 0 and 1. Now, we need to find a suitable value of η. Note that the sum of the two terms is minimized if we set $\eta = \sqrt{\frac{2\ln|\Pi|}{TK}}$, so plugging in that value of η completes the proof:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} \ell_{I_t,t}\right] - \sum_{t=1}^{T} \mathbb{E}_{i\sim\xi_{k,t}}\left[\ell_{i,t}\right] &\leq \frac{\ln|\Pi|}{\sqrt{\frac{2\ln|\Pi|}{TK}}} + \frac{\sqrt{\frac{2\ln|\Pi|}{TK}}}{2}TK \\
&\leq \sqrt{2KT\ln|\Pi|}
\end{aligned}$$

□

This bound shows the power of EXP4 when $K \ll |\Pi|$, and is still good even when the number of experts is exponentially growing with T. Also, the last term is K, not $|\Pi|$, because we can bound the second moment more tightly with this mixing.

7.3 Limitations of Exp4

One downside of Exp4 is that since it works with an arbitrary hypothesis set, the algorithm may run in time exponential in the feature dimension. Consider a case when $|\Pi|$ goes to infinity. For example, if $\Pi = [0,1]^d$ is discretized into ϵ-sized boxes along each dimension with $|\Pi| = \left(\frac{1}{\epsilon}\right)^d$, then EXP4 has regret $\mathcal{O}(\sqrt{dTK\ln\frac{1}{\epsilon}})$. However, Exp4 is often computationally infeasible to run, since in the EXP4 algorithm, we need to maintain a distribution over all possible policies, which can be exponential in the number of features.

8 LinUCB: Linear Stochastic Contextual Bandits

As discussed, Exp4 is computationally infeasible if the set of policies is large. This is because Exp4 ignores the structure of the contexts, even though in reality, the contexts often have structure. For example, we expect that people who like classical music may also share common interests with people who like classical literature, and vice versa. In the LinUCB algorithm, we assume that the reward given a context follows a linear structure with noise. This is a strong assumption, but it makes the problem more computationally tractable, and works well in practice. [1] Note that for LinUCB, the contexts do not need to be i.i.d. - they can be arbitrary. [2]

8.1 LinUCB Algorithm

For each arm $a \in [K]$ (where the set of arms $[K]$ is arbitrary) we have an associated coefficient vector θ_a^*. At each round t, our context for each arm is a vector, $\mathbf{x}_{a,t}$. This contrasts with the non-contextual Linear Bandit problem where we only have one scalar per arm.

For the LinUCB algorithm, we assume that the reward is a linear function of its d-dimensional feature $\mathbf{x}_{a,t}$ with unknown coefficients θ^*_a, plus mean-zero and independent noise ϵ_t.

$$r_{a,t} = \mathbf{x}_{a,t}^T \theta^*_a + \epsilon_t$$

Then, the expected reward given a particular context is simply a linear function of the context:

$$\mathbb{E}[r_{a,t}|\mathbf{x}_{a,t}] = \mathbb{E}[\mathbf{x}_{a,t}^T \theta^*_a + \epsilon_t] = \mathbf{x}_{a,t}^T \theta^*_a + \mathbb{E}[\epsilon_t] = \mathbf{x}_{a,t}^T \theta^*_a$$

Suppose that m is the number of times a has been pulled so far. We define $\mathbf{X}_{a,t}$ to be a matrix with m rows corresponding to the m contexts previously observed for arm a, at the times when arm a was pulled.

$$\mathbf{X}_{a,t} = \begin{bmatrix} \mathbf{x}_{a,1}^T \\ \vdots \\ \mathbf{x}_{a,m}^T \end{bmatrix},$$

where m is the number of times a was pulled. Then we also have a vector of observed rewards from pulling arm a:

$$\mathbf{\Gamma}_{a,t} = \begin{bmatrix} r_{a,1} \\ \vdots \\ r_{a,m} \end{bmatrix},$$

We define vector $\mathbf{b}_{a,t}$ to be

$$\mathbf{b}_{a,t} = \mathbf{X}_{a,t}^T \Gamma_{a,t}$$

Note that the parameters are not shared across arms.

At time t, we apply a ridge regression estimator to estimate the coefficients of each arm a: $\hat{\theta}_{a,t} = (\mathbf{X}_a^T \mathbf{X}_a + \mathbf{I})^{-1} \mathbf{b}_{a,t}$. We need to assume that the noise (how far the true reward differs from the value predicted by the linear function) is conditionally R-sub-Gaussian, where R is a fixed constant [7]. The definition of this is:

$$\forall \lambda \in \mathbb{R} : \quad \mathbb{E}[\exp(\lambda \epsilon_t)|\mathbf{X}_{a,t}, \epsilon_1, \ldots, \epsilon_{t-1}] \leq \exp(\frac{\lambda^2 R^2}{2})$$

Now, let $\lambda > 0$ be a regularization parameter, $V = I\lambda$, define $Y_t = X_t \theta^*_a + \epsilon_t$, and assume that $||\theta^*_a||_2 \leq S$. Define $\mathbf{A}_{a,t} = V + \sum_{s=1}^t \mathbf{x}_{a,s} \mathbf{x}_{a,s}^T$. Also, we define the weighted 2-norm of a vector $\mathbf{x} \in \mathbb{R}^d$ with respect to a positive definite matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$:

$$||\mathbf{x}||_{\mathbf{A}} = \sqrt{\mathbf{x}^T \mathbf{A} \mathbf{x}}$$

It can be shown [7] that, for any $\delta > 0$, with probability at least $1 - \delta$, for all $t \geq 0$, the true coefficient vector for action a, θ^*_a, lies within the following confidence set:

$$C_{a,t} = \left\{ \theta^*_a \in \mathbb{R}^d : \left|\left| \hat{\theta}_{a,t} - \theta^*_a \right|\right|_{\mathbf{A}_{a,t}} \leq R \sqrt{2 \log \Big(\frac{\det(\mathbf{A}_{a,t})^{1/2} \det(\lambda \mathbf{I})^{-1/2}}{\delta} \Big)} + \lambda^{1/2} S \right\}$$

This provides a confidence bound on how far our estimated coefficients (based on our learned $\hat{\theta}_a$) are from the true coefficients. The proof of these confidence sets can be found in [7]. Now, we apply an approach similar to UCB. For each arm, we calculate an upper bound on the expected reward, by computing the reward produced by the best θ_a within that arm's confidence set.

$$u_{a,t} = \max_{\theta_a \in C_{a,t-1}} \mathbf{x}_{a,t}^T \theta_a$$

Then, choose the arm a_t with the highest confidence bound on its reward.

$$a_t = \arg\max_{a \in \mathcal{A}_t} u_{a,t}$$

Algorithm 3 presents the full algorithm.

Algorithm 3 LinUCB with Contextual Bandits

```
Input: $R \in \mathbb{R}^+$, regularization parameter $\lambda$
for $t = 1, 2, \dots, T$ do
  Observe feature vectors of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{a,t} \in \mathbb{R}^d$
  for all $a \in \mathcal{A}_t$ do
    if $a$ is new then
      $\mathbf{A}_a \leftarrow \lambda \mathbf{I}_d$ ($d$-dimensional identity matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ ($d$-dimensional zero vector)
    end if
    $\hat{\theta}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
    $C_{a,t} \leftarrow \left\{ \theta_a^* \in \mathbb{R}^d : \left|\left|\hat{\theta}_{a,t} - \theta_a^*\right|\right|_{\mathbf{A}_a} \le R\sqrt{2\log\left(\frac{\det(\mathbf{A}_a)^{1/2}\det(\lambda I)^{-1/2}}{\delta}\right)} + \lambda^{1/2}S \right\}$
    $p_{a,t} \leftarrow \arg\max_{\hat{\theta}_a \in C_{a,t}} \mathbf{x}_{a,t}^T\hat{\theta}_a$
  end for
  Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{a,t}$ with ties broken arbitrarily, and observe payoff $r_t$
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{a_t,t}\mathbf{x}_{a_t,t}^T$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{a_t,t}$
end for
```

8.2 Regret Analysis

Although LinUCB works well in practice, there are some technical difficulties in analyzing it, because the actions in future rounds are not independent of previous rounds (previous rounds influence our prediction of θ and thus future actions). To prove the correctness of the bounds, martingale techniques are needed. The approach is similar to the non-contextual case. [7]

The regret analysis of LinUCB hinges on three conceptual steps [8]:

- If we have valid confidence intervals, then the regret analysis applies as normal UCB, and the regret can be bounded by the size of the confidence intervals used at each time step.
- Theorem 2 in [7] provides general conditions on when ellipsoid confidence intervals are valid, which we elaborate on below.
- Lemma 11 in [7] provides a bound on the sum of confidence intervals $\sum_{t=1}^{T} \hat{c}_{a_t,t}$, when they are ellipsoid confidence intervals. That is used in to prove the regret bound stated as Theorem 3 in [7]. We elaborate on this below as well.

Using a similar proof to the Linear Bandits lecture [9], we can obtain a regret bound of

$$R_n = O\left(RS\lambda^{1/2}dK\ln(T/\delta)\sqrt{T}\right)$$

where R is the width of the tail of the randomness of the feedback, $S \ge\| \theta \|$ is an upper bound on the norm of the true model, λ is the regularization parameter for estimating the mean and is required to be $\lambda \ge \max_x \| x \|^2$, and with probability at least $1-\delta$, the above inequality is satisfied.

LinUCB's regret bound has a worse dependence on the number of arms and dimensions than Exp4's corresponding regret bound of $\mathcal{O}(\sqrt{dTK\ln\frac{1}{\epsilon}})$ (see section 7.3). However, LinUCB is much more computationally efficient than Exp4, as its runtime does not have an exponential dependence on the feature dimension.

8.3 Extension to non-linear case

If the actual rewards do not follow a linear function of the contexts, it may still be possible to use LinUCB by embedding the contexts into a high-dimensional space using a feature mapping ϕ. This is very similar to the approach used by kernel methods and support vector machines. [2]

9 Epsilon-greedy methods

While still in the stochastic setting, we come back to looking at the general i.i.d. contextual bandit problem – i.e. where rewards are *not necessarily linearly* dependent on the context. Recall that using Exp4 for this problem involves maintaining a probability distribution over all policies and updating them at every round. This operation has a storage complexity of the order of $|\Pi|$, which can be very expensive for a large policy class. In this section, we discuss a strategy which combines random exploration with greedy exploitation to achieve a regret $\sim \mathcal{O}(T^{2/3})$. Even though this is significantly inferior to the Exp4 regret of $\sim \mathcal{O}(\sqrt{T})$, the ϵ-Greedy method is frequently used for the huge advantage it provides in storage complexity.

Formalizing the problem, we have our usual (i.e. not necessarily linear) bandit setting with a joint distribution $\mathcal{P}$ over contexts, and rewards of actions (given a context). For the purpose of our discussion, we are only considering the case where $\mathcal{P}$ is **time invariant**. Arms are chosen according to a selected policy π. The "value" of this policy is defined as the expected average reward by choosing π:

$$V(\pi) = \mathbb{E}_{x\sim\mathcal{P}(x)}\left[\mathbb{E}_{r\sim\mathcal{P}(\cdot|\pi(x),x)}[r]\right] \tag{6}$$

9.1 Greedy with full information

A simple baseline for comparison is a contextual bandit setting where the rewards of **all** arms are revealed to the learner at every round, not just that of the selected arm. Here, there is no need for exploration, since we can go on picking the best strategy and updating our estimates at every time t. This will be referred to as the (full-information) Greedy algorithm. In formal terms, it is stated in Algorithm 4.

Algorithm 4 Algorithm for Greedy with full information

Initialize Training set $S = \emptyset$
For each round $t = 1, \cdots, T$:

- **Receive** context $x_t \in \mathcal{C}$.
- **Determine** policy, based on training data; ties being broken arbitrarily.

$$\pi_\tau = \arg\max_{\pi\in\Pi} \sum_{(a,r)\in S} r_{a,t}\mathbb{1}_{\pi(x)=a}$$

- **Play** arm $a_t = \pi_\tau(x_t)$ according to determined policy.
- **Observe** rewards $r_{a,t}$ *for each arm* a.
- Update training set with collected samples $S \leftarrow S \cup \{(x_t, a_t, r_t)\}$.

Theorem 5. *The regret of* Greedy with full information *is bounded as*

$$R_{\text{Greedy}} = TV(\pi^*) - \sum_{t=1}^{T} V(\pi_t) \le \sqrt{8T\log\frac{2N}{\delta}}$$

where $\pi^* = \arg\max_{\pi\in\Pi} V(\pi)$*, is the optimal policy.*

Proof. To prove this, we start by bounding $V(\pi^*) - V(\pi_t)$, the *instantaneous regret* at a single round t. For any fixed policy π, $r_s(\pi(x_s))$, where $s = 1, 2, ..., t$ are i.i.d. random variables with expectation $V(\pi)$

(by definition (6)). As earlier, we assume that the rewards $r_s(\pi(x_s))$ are bounded between $[0,1]$. Using **Hoeffding's inequality** for these variables, we have w.p. at least $1-\delta$,

$$\left|\frac{1}{t}\sum_{s=1}^{t} r_s(\pi(x_s)) - V(\pi)\right| \le \sqrt{\frac{1}{2t}\log\frac{2}{\delta}} \tag{7}$$

There is a catch here – the inequality holds only for *any single* π, but we want it to hold for *simultaneously for every* π, in particular, for the chosen policies over all rounds t

$$\pi_t = \arg\max_{\pi\in\Pi}\sum_{s=1}^{t} r_s(\pi(x_s)) \tag{8}$$

This can be done using the **union bound**. Let $\mathcal{E}(\pi)$ be the event that Equation (7) holds for π.

$$\mathbb{P}(\mathcal{E}(\pi)^{\mathsf{c}}) \le \delta \qquad (\because (7))$$

$$\mathbb{P}\left(\bigcup_{\pi\in\Pi}\mathcal{E}(\pi)^{\mathsf{c}}\right) \le \sum_{\pi\in\Pi}\mathbb{P}(\mathcal{E}(\pi)^{\mathsf{c}}) \le N\delta \qquad \text{where } N = |\Pi|$$

which guarantees that Equation (7) holds *for all policies* π with probability at least $1-N\delta$.

Invoking Equation (7) for both π_t and π^*, we have

$$V(\pi_t) \ge \sum_{s=1}^{t} r_s(\pi_t(x_s)) - \sqrt{\frac{1}{2t}\log\frac{2N}{\delta}},$$
$$V(\pi^*) \le \sum_{s=1}^{t} r_s(\pi^*(x_s)) + \sqrt{\frac{1}{2t}\log\frac{2N}{\delta}}$$

also, we have by Equation (8) $\quad \displaystyle\sum_{s=1}^{t} r_s(\pi_t(x_s)) \ge \sum_{s=1}^{t} r_s(\pi^*(x_s))$

Thus,

$$\begin{aligned} V(\pi_t) &\ge \sum_{s=1}^{t} r_s(\pi_t(x_s)) - \sqrt{\frac{1}{2t}\log\frac{2N}{\delta}} \\ &\ge \sum_{s=1}^{t} r_s(\pi^*(x_s)) - \sqrt{\frac{1}{2t}\log\frac{2N}{\delta}} \\ &\ge V(\pi^*) - 2\sqrt{\frac{1}{2t}\log\frac{2N}{\delta}} \end{aligned} \tag{9}$$

which gives us the suboptimality at round t. For summing over $t = 1,2,...,T$, we use the inequality

$$\sum_{t=1}^{T}\frac{1}{\sqrt{t}} < \sum_{t=1}^{T}\frac{2}{\sqrt{t}+\sqrt{t-1}} = \sum_{t=1}^{T} 2(\sqrt{t}-\sqrt{t-1}) = 2\sqrt{T}$$

Substituting in Equation (9), we have the bound

$$T\max_{\pi\in\Pi} V(\pi) - \sum_{t=1}^{T} V(\textsc{Greedy}\pi) \le \sqrt{8T\log\frac{2N}{\delta}}$$

holding w.p. $1-\delta$. $\square$

9.2 Partial Exploration

We now tackle the bandit feedback setting where we do not get to see the reward for all arms, and exploration is necessary. An obvious approach (called the the ϵ-greedy algorithm) is to have a preliminary exploration phase (for $\tau = \epsilon T$ trials – which is a fraction of the total number of trials), and reap its benefits in a pure exploitation phase. To fairly evaluate the performance achieved by the pure exploration in τ trials, we restrict the algorithm to learn only from the exploration phase rewards, and not from the exploitation phase rewards, for this naive case. Note that we assume the knowledge of the time horizon T here. Formally, the process is stated in Algorithm 5.

Algorithm 5 ϵ-Greedy Algorithm

```
Parameter $\tau$ dependent on stopping time $T$
Initialize Training set $S = \emptyset$
For each round $t = 1, \cdots, \tau$ (exploration):
    • Receive context $x_t \in \mathcal{C}$.
    • Sample an action $a_t \in \mathcal{A}$ uniformly at random.
    • Observe reward $r_t$. Update training set $S \leftarrow S \cup \{(x_t, a_t, r_t)\}$
For each round $t = \tau + 1, \cdots, T$ (exploitation):
    • Determine policy, based on training data
          $\pi_\tau = \arg\max_{\pi \in \Pi} \sum_{(a,r) \in S} r \mathbb{1}_{\pi(x)=a}$     (10)
    • Receive context $x_t \in \mathcal{C}$.
    • Play action $a_t = \pi_\tau(x_t)$ according to determined policy.
    • Observe reward $r_t$.
```

Theorem 6. *If the number of exploration steps $\tau \propto T^{2/3}$, then the regret of ϵ-Greedy is at most of order $\mathcal{O}(T^{2/3})$*

Proof. For an illustrable comparison with Equation (8), we reformulate Equation (10) as

$$\begin{aligned}\pi_\tau &= \arg\max_{\pi \in \Pi} \sum_{s=1}^{\tau} r_s \mathbb{1}_{\pi(x_s)=a_s} \\ &= \arg\max_{\pi \in \Pi} \hat{R}_\tau(\pi) \qquad \text{(here, we define } \hat{R}_\tau(\pi) \text{ as the past reward had we used policy } \pi) \qquad (11)\end{aligned}$$

Let us consider the expected value of a single reward –

$$\begin{aligned}\mathbb{E}_{\mathcal{P}(r,x)}[r_s \mathbb{1}_{\pi(x_s)=a_s}] &= \mathbb{E}_{x \sim \mathcal{P}(x)}\left[\mathbb{E}_{r_s \sim \mathcal{P}(\cdot|x,\pi(x))}[r_s \mathbb{1}_{\pi(x_s)=a_s}|x]\right] \\ &= \mathbb{E}_{x \sim \mathcal{P}(x)}\left[\mathbb{E}_{r \sim \mathcal{P}(\cdot|x,\pi(x))} \frac{[r(\pi(x_s))|x]}{K}\right] \qquad (\because a_s \text{ is being randomly chosen}) \\ &= \frac{V(\pi)}{K} \qquad \text{(by Equation (6))} \qquad (12)\end{aligned}$$

Note that the quantity $r_s \mathbb{1}_{\pi(x_s)=a_s}$ appearing in the LHS above, when divided by the probability of choosing arm a_s, would be an unbiased estimator of $V(\pi)$. This agrees with the inverse propensity scoring method discussed in Section 5, where the arm sampling is uniform, with probability $1/K$.

Now, we can apply Hoeffding's inequality to the i.i.d. random variables $\{r_s \mathbb{1}_{\pi(x_s)=a_s}\}$, using Equations (11) and (12). This is legitimate according to the same argument used for justifying equation 7, where we need to use the union bound and replace δ by $N\delta$ to guarantee the bound for all of the N policies simultaneously. Thus, w.p. at least $1-\delta$,

$$\left| \frac{1}{\tau} \hat{R}_\tau(\pi) - \frac{V(\pi)}{K} \right| \le \sqrt{\frac{1}{2\tau} \log \frac{2N}{\delta}} \tag{13}$$

Invoking Equation (13) for both π_t and π^*, we have

$$\begin{aligned} \frac{V(\pi_t)}{K} &\ge \frac{1}{\tau} \hat{R}_\tau(\pi_t) - \sqrt{\frac{1}{2\tau} \log \frac{2N}{\delta}} \\ \frac{V(\pi^*)}{K} &\ge \frac{1}{\tau} \hat{R}_\tau(\pi^*) + \sqrt{\frac{1}{2\tau} \log \frac{2N}{\delta}} \end{aligned}$$

Thus,

$$\begin{aligned} V(\pi_t) &\ge \frac{K}{\tau} \hat{R}_\tau(\pi_t) - K\sqrt{\frac{1}{2\tau} \log \frac{2N}{\delta}} \\ &\ge \frac{K}{\tau} \hat{R}_\tau(\pi^*) - K\sqrt{\frac{1}{2\tau} \log \frac{2N}{\delta}} \\ &\ge V(\pi^*) - K\sqrt{\frac{2}{\tau} \log \frac{2N}{\delta}} \end{aligned}$$

gives us the suboptimality at round t; we sum this over $t = \tau+1, ..., T$. Note that the first τ rounds, being random pulls, can have an arbitrary regret.

Thus, w.p. $1-\delta$, we have the bound

$$T \max_{\pi \in \Pi} V(\pi) - \sum_{t=1}^{T} V(\epsilon\text{-}\textsc{Greedy}\ \pi) \le \text{ (constant) } \cdot \tau + TK\sqrt{\frac{2}{\tau} \log \frac{2N}{\delta}}$$

To find the τ which minimizes the above expression, taking the derivative gives optimum for $\tau \propto T^{2/3}$, and the entire expression for the regret $\propto T^{2/3}$. As opposed to the Greedy regret of $\mathcal{O}(\sqrt{T})$, ϵ-Greedy has a regret $\sim \mathcal{O}(T^{2/3})$.

□

Epoch-greedy methods
For the ϵ-Greedy algorithm, we assume that the learner has knowledge of the time horizon T. The number of exploration steps τ was determined as a function of T. If T is unknown, the same strategy can be used in *epochs*. Each epoch has one exploration round and as many exploitation rounds as we can fit in, while keeping the regret bound comparable to the known time horizon case – namely – $\mathcal{O}(T^{2/3})$. If we expect a regret of ε_t after epoch t, we are allowed to run exploitation for $\lceil 1/\varepsilon_t \rceil$ rounds. The details of this approach, including some comparisons to Exp4, are elaborated in [10].

10 ILOVETOCONBANDITS

The ϵ-greedy and epoch-greedy methods choose arms uniformly at random during the exploration phase, and therefore incur a suboptimal $\mathcal{O}(T^{2/3})$ regret. We should be able to improve on this if we use our previously-observed rewards to influence our exploration strategy. For example, if a policy has poor empirical regret on our existing observations, and the policy's empirical regret is an accurate estimate of its true regret, we should stop exploring that policy. Ultimately, we would like to achieve the optimal $\mathcal{O}(\sqrt{T})$ regret. The Exp4 algorithm does achieve a similar $\mathcal{O}(\sqrt{TK \ln |\Pi|})$ regret, but the computational complexity is linear in $|\Pi|$, since it maintains a **dense** distribution over all policies, and updates the probability of each policy at every step. This makes it computationally infeasible for large policy sets.

Recently, the ILOVETOCONBANDITS algorithm [11] was developed, which uses an adaptive exploration strategy to achieve the statistically-optimal $\mathcal{O}(\sqrt{T})$ regret, while still being computationally feasible to run. In order to avoid a $\Omega(|\Pi|)$ time complexity, we restrict ourselves to algorithms that only access the policy class through an optimization oracle, which returns the policy with the highest empirical reward given the observations so far. The algorithm can maintain a distribution over policies, but it must be **sparse** (i.e. most policies must have zero probability) in order to avoid computing probabilities for each policy. The following discussion is based on Alekh Agarwal's lecture notes. [12]

10.1 ArgMax Oracle

The algorithm assumes access to an oracle that can find the policy in Π which maximizes the empirical reward over previous observations, and makes $\tilde{O}(\sqrt{KT})$ oracle calls across all T rounds. We define this as the ArgMax Oracle:

Definition 1. *For a set of policies Π, the **ArgMax Oracle ($\mathcal{AMO}$)** is an algorithm, which for any sequence of context and reward vectors, $(x_1, r_1), (x_2, r_2), \ldots, (x_t, r_t) \in X \times \mathbb{R}_+^A$, returns $\arg\max_{\pi \in \Pi} \sum_{\tau=1}^{t} r_\tau(\pi(x_\tau))$.*

In other words, the oracle returns the policy (within the pre-defined policy set Π) that maximizes the reward over the first t contexts. If we have this ArgMax oracle, we can implement the ϵ-GREEDY method from the previous section by calling the oracle once to compute the GREEDY policy after τ rounds. Unfortunately, the regret bound for that scales as $T^{2/3}$, instead of the optimal $\sqrt{T}$. The algorithm below demonstrates how to improve the regret bound by making our exploration strategy adaptive to previously-observed rewards.

10.2 Notation

$V(\pi)$ denotes the expected reward of policy π. After the contextual bandit algorithm runs for t rounds, we have an empirical dataset of the form $(x_s, a_s, p_s, r_s)_{s=1}^t$, where p_s is the probability that arm a_s was chosen at round s. To estimate the expected reward of policy π, $V(\pi)$, we use an Inverse Propensity Score estimator; we showed in Theorem 1 that this is an unbiased estimator of the reward:

$$\hat{V}_t(\pi) = \frac{1}{t} \sum_{s=1}^{t} r_t \frac{\mathbb{1}_{a_t = \pi(x_t)}}{p_t} \tag{14}$$

We now define the expected and empirical regret of a single policy π as follows:

$$\bar{R}(\pi) = \max_{\pi' \in \Pi} V(\pi') - V(\pi) \tag{15}$$

$$\widehat{R}_t(\pi) = \max_{\pi' \in \Pi} \hat{V}_t(\pi') - \hat{V}_t(\pi)$$

The algorithm maintains a probability distribution over policies, which we denote by q_t; the probability of policy π is $q_{\pi,t}$. This induces a distribution over arms, given the context x, $q_t(a|x)$:

$$q_t(a|x) = \sum_{\pi \in \Pi : \pi(x) = a} q_{\pi,t}$$

In other words, $q_t(a|x)$ is the sum of the probabilities of all policies which recommended arm a under the current context x. Finally, since we want to ensure that our reward estimates have low variance, we should prevent the probability of any action from being too small. To accomplish this, we assign a minimum probability for each action (γ). Then, there is $(1 - K\gamma)$ probability left, which we split among the arms according to the $q_t(a|x)$ distribution. We use the shorthand $q_t^\gamma(a|x)$ to be the smoothed mixture of q_t with the uniform distribution over arms:

$$q_t^\gamma(a|x) = (1 - K\gamma) \sum_{\pi\in\Pi:\pi(x)=a} (q_{\pi,t} + \gamma)$$

10.3 Finding a good exploration distribution

The key idea of the ILOVETOCONBANDITS algorithm is that it maintains a sparse distribution over policies, $q_{i,t}$, to sample from. This distribution should be chosen to guarantee a good balance between exploration and exploitation. In particular, the distribution should favor policies with low empirical regret, while making sure that for each policy, that the empirical regret is a low-variance estimator of the true regret (so that we do not inaccurately neglect policies simply because of poor reward estimates).

To enforce a bound on the variance of our regret estimate, we use the following fact (see page 6 of [12] for the proof):

Lemma 4. *For any fixed time step t and policy $\pi \in \Pi_t$:*

$$Var_t[\hat{V}_t(\pi)] \leq \mathbb{E}_t \mathbb{E}_{x_t\in\mathcal{C}} \left[\frac{1}{q_t^\gamma(\pi(x_t)|x_t)}\right]$$

where $\mathbb{E}_t[Y]$ is shorthand for $\mathbb{E}[Y|(x_s, a_s, p_s, r_s)_{s=1}^t]$, and $Var_t[Y]$ is shorthand for $Var[Y|(x_s, a_s, p_s, r_s)_{s=1}^t]$.

Assume that we are given samples $(x_s, a_s, p_s, r_s)_{s=1}^t$, and minimum probability γ. Define $b_t(\pi) = \frac{\widehat{R}_t(\pi)}{4(\epsilon-2)\gamma \ln T}$. To formalize our desired constraints, we want a distribution over policies $Q(\pi)$ that satisfies the following Equations 16 and 17. We refer to these two constraints collectively as the **Optimization Problem (OP)**.

$$\sum_{\pi\in\Pi} q_{\pi,t} b_t(\pi) \leq 2K \tag{16}$$

$$\forall \pi \in \Pi : \mathbb{E}_{x\sim D}\left[\frac{1}{q_t^\gamma(\pi(x)|x)}\right] \leq 2K + b_t(\pi) \tag{17}$$

Recall that $q_{\pi,t}$ is the distribution's probability of policy π, and $b_t(\pi)$ is proportional to the empirical regret of policy π. The first constraint (Eq. 16) imposes a bound on the weighted sum of the empirical regret across all policies (weighted by the probability of each policy, $q_{\pi,t}$). This forces q_t to assign higher probabilities to policies with low empirical regret. Then, the second constraint (Eq. 17) enforces a low-variance constraint for each policy, since according to Lemma 4, the variance of the empirical regret of policy π depends inversely on $q_t^\gamma(\pi(x)|x)$. Intuitively, for every single policy, the second constraint forces the $q_t(a|x)$ distribution to place enough weight on arms frequently selected by that policy, to ensure that our empirical regret estimate for that policy has low variance. Importantly, we allow for different bounds for each policy π. For policies that have poor empirical regret, we allow for larger estimation variance; even with a small number of samples, it is clear that the policy is unlikely to be good.

10.4 Algorithm

Now that we've defined the Optimization Problem, we can use the following algorithm to select actions [12]:

1. At time t, find a distribution q_t which satisfies OP using past samples, i.e. with $b_{t-1}(\pi)$.

2. Observe context x_t, and play an arm according to $q_t^\gamma(a|x_t)$

3. Observe reward r_t and incorporate (x_t, a_t, p_t, r_t) to the dataset.

The algorithm maintains a distribution over all policies, q_t. At all time steps, we must find a distribution q_t that satisfies OP's constraints, which can be done using a coordinate descent algorithm. [11] If OP's constraints are met at all times t, we can derive the following regret bound.

Theorem 7. *Suppose an algorithm plays according to a distribution which is a solution to OP at each round t. Then with probability at least $1-\delta$, the algorithm incurs a regret no more than $\mathcal{O}\left(\sqrt{KT\ln(\frac{TN}{\delta})} + K\ln\frac{TN}{\delta}\right)$.*

The proof [12] is omitted due to its complexity; it involves martingale techniques to handle the fact that the arm probabilities at each time step are heavily influenced by previous contexts, actions, and rewards. However, it is noteworthy that the regret scales with the optimal $\sqrt{T}$. Now, the main question is whether it is possible to efficiently find a distribution q_t satisfying the OP constraints.

10.5 Solving Optimization Problem With Coordinate Descent

In fact, there exists a coordinate descent algorithm [11] which is guaranteed to efficiently find a distribution q_t satisfying the OP constraints. While the details are omitted here, the algorithm is always able to find a distribution satisfying the OP constraints within $\frac{4\ln(1/(K\gamma))}{\gamma}$ iterations, where γ is the minimum probability of an action. [12] The algorithm starts by initializing the q_t vector to be sparse (such as with the zero vector). For each iteration, it manages to enforce the low-regret constraints while maintaining the sparsity of the distribution (in each iteration, at most one non-zero weight is added to the q_t distribution). Then, it uses the ArgMax oracle in a clever way to check if equation 17 (the low-variance constraint) is violated; if so, it readjusts the policy distribution to reduce the violation of the constraint.

10.6 Thoughts on ILOVETOCONBANDITS

In summary, the ILOVETOCONBANDITS algorithm uses an adaptive exploration strategy to achieve the statistically optimal regret guarantee for i.i.d. contextual bandit problems. It is also efficient, as it invokes the $\mathcal{AMO}$ at most $\tilde{O}(\sqrt{T})$ times over T rounds. In contrast, ϵ-Greedy requires exactly one $\mathcal{AMO}$ call but is suboptimal in regret, while EXP4 has no known implementation via the $\mathcal{AMO}$ and has a runtime which scales linearly in the number of policies. However, for practical implementation, ϵ-Greedy is often the most convenient and efficient to implement.

11 Summary: Comparison of algorithms

EXP4 uses exponential weighting to achieve $\tilde{O}(\sqrt{T})$ regret. It can handle arbitrary (but finite) policy sets as well as adversarial contexts. However, its computational complexity may be exponential in number of features. Epoch-Greedy is computationally efficient (given an oracle optimizer) but has a weaker regret guarantee of $\tilde{O}(T^{2/3})$. The new ILOVETOCONBANDITS algorithm achieves $\tilde{O}(\sqrt{T})$ regret while still being computationally feasible (with a sublinear number of oracle calls). While that approach is still under development, an online variant of that algorithm could provide a scalable solution to the contextual bandit problem.

For a broad overview of the various algorithms appropriate for the settings considered in this lecture, we break down our assumptions into four categories as in the below table 11.

	Small $\|\Pi\|$	Large $\|\Pi\|$
Stochastic	Exp4	LinUCB, variations of Greedy, ILOVETOCONBANDITS
Adversarial	Exp3, Exp4	Exp4

References

[1] Lihong Li and Wei Chu et al. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[2] Ambuj Tewari and Susan A. Murphy. From ads to interventions: Contextual bandits in mobile health. In *Mobile Health*. Springer, 2017.

[3] Kevin Jamieson. Non-stochastic bandits. `https://courses.cs.washington.edu/courses/cse599i/18wi/resources/lecture5/lecture5.pdf`, February 2018.

[4] Sebastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. In *Foundations and Trends in Machine Learning*, volume 5, pages 1–122, 2012.

[5] Eric W. Weisstein. Jensen's inequality. `http://mathworld.wolfram.com/JensensInequality.html`. Accessed: 2018-02-05.

[6] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–30, 2002. Copyright - Copyright] 2002 Society for Industrial and Applied Mathematics; Last updated - 2012-02-01.

[7] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2312–2320. Curran Associates, Inc., 2011.

[8] Yisong Yue. A note on linear bandits and confidence sets. 2016.

[9] Kevin Jamieson. Linear bandits. `https://courses.cs.washington.edu/courses/cse599i/18wi/resources/lecture9/lecture9pdf`, February 2018.

[10] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2008.

[11] Vasilis Syrgkanis, Haipeng Luo, Akshay Krishnamurthy, and Robert E Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3135–3143. Curran Associates, Inc., 2016.

[12] Alekh Agarwal. Exploration in contextual bandits. `http://alekhagarwal.net/bandits_and_rl/exploration.pdf`, October 2017.