COMS E6998.001: Bandits and Reinforcement Learning **Fall 2017**

Introduction to Contextual Bandits

Lecturer: Alekh Agarwal *Lecture 6, October 11*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

6.1 Formal Setting and Examples

The contextual bandit (CB) setting is best described as a repeated interaction which proceeds over T rounds.

At each round $t = 1, 2, \ldots, T$:

1. Environment reveals a context $x_t \in \mathcal{X}$.
2. Learner chooses an action $a_t \in \mathcal{A}$.
3. Environment reveals a reward $r_t \in [0, 1]$.

Figure 6.1: The repeated interaction between the environment and a learning agent in the contextual bandit problem

The goal of the learner is to choose actions which maximize its cumulative reward, that is $\sum_{t=1}^{T} r_t$. Throughout, we will assume that there is an (unknown) joint distribution over contexts as well as the rewards of all the actions, denoted as D_t at time t. With some abuse of notation, we will also use D_t to denote the marginal distribution over contexts and conditional distribution over the rewards given a context. With this, the above interaction involves environment choosing $x_t \sim D_t$ and $r_t \sim D_t(\cdot|x_t, a_t)$. The learner is equipped with a set of policies $\Pi \subseteq \{\mathcal{X} \to \mathcal{A}\}$, and aims to find a policy $\pi \in \Pi$ which attains a large reward. The (expected) average reward of a policy π, often also called the value of π is defined as

$$V(\pi) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{x \sim D_t} \left[\mathbb{E}[r_t | x, \pi(x)] \right]. \tag{6.1}$$

We will often use the shorthand $\mathbb{E}_{D_t}[r(\pi(x))]$ to refer to the above expectation concisely.

We will focus on a setting where the set Π is either extremely large, or infinite. Similarly, we will typically consider the set $\mathcal{X}$ to be infinite, so that learner has to *generalize* across contexts. The set of actions $\mathcal{A}$, on the other hand will be expected to have a reasonable cardinality, with $|\mathcal{A}|$ denoted by K. Indeed, we will often identify the actions set with $[K]$, the set of integers $\{1, 2, \ldots, K\}$.

With the formal notation in place, we now flesh out a couple of examples.

Example 1 (Online news recommendation) *Let us consider a news website, which has a pool of candidate stories at any given point of time. When a user visits the website, it wants to decide which stories to display to the user. To simplify the discussion, let us assume for now that only one story out of the pool will be actually presented to the user. In this case, the context x can consist of any information about the user, such as their location, browser, any reading*

habits as gleaned from prior visits to the website, etc. The context might further contain features describing the content of the various news stories in the pool. The action set $\mathcal{A}$ consists of the pool of news stories, with the tacit assumption that this set is not too large, say at most 30. Rewards should measure user engagement with the chosen news story, which can be measured in a number of ways. We could use clicks, meaning a reward of 1 if the chosen story is clicked on by the user. More fine-grained rewards could further encode the time spent by the user reading the story, whether they explicitly like or share the story on social media etc. Policies in this setting might extract some features from the available information, and predict a real number for every action describing its quality. Concretely, if the features are described as a vector in $x \in \mathbb{R}^d$, then a policy might maintain a weight vector $w_i \in \mathbb{R}^d$ for each action i, and the action picked by the policy can be $\arg\max_{i=1}^{K} w_i^T x$ with ties broken in favor of smaller i. More generally, we can use any other multiclass classifier, such as based on tree ensembles, neural nets, etc. and this forms the most natural policy sets in practice.

Notice that we picked reward measures which can be clearly associated with a particular action, such as a click or time spent reading the selected story above. Often, longer term measures such as the number of visits by a user in a month might be more accurate measures of user engagement with the website. However, maximizing such metrics falls outside the scope of the contextual bandit problem, and results in harder reinforcement learning problems which we will visit in the later part of the course. Often the best practical approach is to find a well-defined short-term metric such as time spent on a story, which might be well correlated with desired longer term measures.

Example 2 (Mobile health advice) *As our second example, we consider a mobile health application which is designed to prompt people suffering from a chronic condition such as diabeters or obesity to take a short exercise break. Let us assume, for simplicity, that the app displays these messages at 3 fixed times in the day so that the actions do not involve choosing the time of messaging. Suppose we have a small set of candidate messages, but not all phrasings and not all exercise options succeed equally for all users. Furthermore, the ideal choice might be different during the day when a person is in their office, and in the evening when they are at home. In this case, the context consists of broad health information about a person such as gender, age, height, weight etc., as well as some coarse summary of their previous interactions with the app (such as what type of workouts they tend to prefer). Actions are the different messages we can display and policies can be designed along similar lines as the previous example. A reward might be a binary indicator of whether the suggestion was taken or not, either indicated by the user explicitly, or measured using a wearable device.*

Note that in both the above examples, part of the context is assumed to be based on the previous interactions between the agent and its environment, through past reading habits or the exercise suggestions which were followed. This is typical of many contextual bandit problems, where even in a non-adversarial setting, the future contexts are affected by the chosen actions. However, this effect is often relatively small over short and medium time-spans and hence we choose to ignore the different distributions that some other policy might have generated when we evaluate it.

We conclude this section with one final example.

Example 3 (Faulty VM handling) *In this example, consider the problem of dealing with faulty virtual machines (VMs) in a large cloud service. The context consists of various telemetry readings about the workload on the VM when it stopped responding, hardware and software profile of the physical machine and the VM as well as broad indicators of the general health of the data center or the rack where the VM is located. Actions can be discrete waiting times, say 1 second apart between a smallest and largest reasonable duration, before which the VM should be either rebooted or migrated. If the machine recovers during the chosen waiting period, then the reward can be that chosen period. If not, the reward can be the chosen waiting time plus the time cost of a reboot or a migration.*

Finally, we describe the objective of learning in contextual bandit problems more concretely. A typical learning objective is to minimize regret, which is defined as

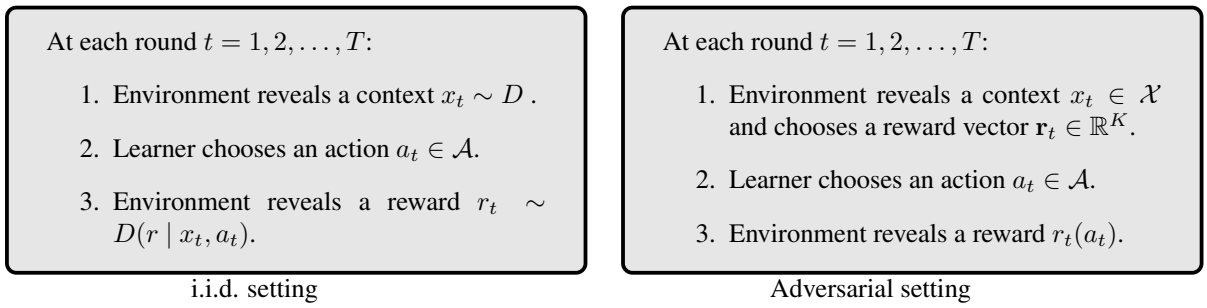

Figure 6.2: The repeated interaction between the environment and a learning agent for stochastic and adversarial CB

$$\text{Regret}_T = \max_{\pi \in \Pi} \sum_{t=1}^{T} \mathbb{E}_{D_t}[r(\pi(x))] - \sum_{t=1}^{T} \mathbb{E}_{D_t}[r(a_t)]. \tag{6.2}$$

The definition looks rather complicated, and it is worthwhile unpacking it a little bit for two versions of the problem, i.i.d. and adversarial as shown in Figure 6.1. In the i.i.d. setting, the distributions D_t are identical for all rounds t. In the adversarial setting, the rewards and contexts at round t can depend on all the previous actions of the the player. However, it turns out that the worst-case adversary does not need to choose contexts and rewards from a distribution. That is, the worst-case strategy for the adversary uses degenerate distributions D_t, since the rewards are only revealed to the player after the action a_t has been taken. Consequently, we can also specify the notions of regret for these settings as

$$\text{Regret}_T = \max_{\pi \in \Pi} \sum_{t=1}^{T} \mathbb{E}_{D}[r(\pi(x))] - \sum_{t=1}^{T} \mathbb{E}_{D}[r(a_t)]. \qquad \text{i.i.d. setting} \tag{6.3}$$

$$\text{Regret}_T = \max_{\pi \in \Pi} \sum_{t=1}^{T} r_t(\pi(x_t)) - \sum_{t=1}^{T} r_t. \qquad \text{adversarial setting} \tag{6.4}$$

6.2 Connections to other learning settings

In this section, how we will review how contextual bandits relate to multi-armed bandits, supervised learning and reinforcement learning.

Connections to multi-armed bandits: The similarities and differences between multi-armed and contextual bandits are most apparent from the notion of regret (6.2). While we also consider regret in multi-armed bandits (MAB), the algorithm's cumulative reward is compared to that of the *best fixed action in hindsight*. This corresponds to an extremely simple policy class of ignoring the context completely. Clearly this makes MAB a special case of CB. However, this special case can be overly restricted in most practical applications. Considering our earlier examples, applying MAB to news recommendation, mobile health and faulty VM handling corresponds to making the same choice, irrespective of who the user is or the characteristics of the VM in question. Contextual bandits with an expressive policy class can allow us to obtain a much higher reward in most of these settings.

Connections to reinforcement learning: At the other extreme, reinforcement learning (RL) is a strict generalization of CB. Loosely stated, while the agent's actions influence the future state of the world in RL, this is not the case in

CB. More formally, we again examine the notion of regret in adversarial CB problems (6.4). In this case, the contexts and rewards chosen by the adversary at time t depend on the actions of the algorithm at all the past rounds. That is, the algorithms choices does affect the future data it experiences. Crucially though, the algorithm and any other policy π in the second term of regret are both evaluated under the same contexts and rewards. We do not account for the fact that if the algorithm had indeed followed the policy π, then the adversary would have responded with different contexts and rewards. In RL, we would indeed consider this more complex benchmark, where each policy is evaluated on the data distribution that would be generated in response to its actions. Thus, CB makes a simplifying assumption compared to RL.

It turns out this assumption is already partly violated even in the examples we discussed earlier. In news or health advice recommendation tasks, it is common to use features which rely on the previous history of a user's interaction, such as the number of times they took a particular suggestion. Of course, this depends on the policy making these recommendations and formally violate the CB assumption. As we will see in the sequel, this simplification is critical for the development of practical algorithms, and still ends up being a reasonable approximation for many problems of interest. We also notice that the simplification is automatically correct in the i.i.d. version of the problem.

Connections to supervised learning: In practice, many CB problems are often cast as supervised learning problems instead. For instance, suppose we are interested in news recommendation. Then for every user, we record the chosen article and whether they read it or not. We now cast this as a conditional probability prediction problem, where given a (user, article) pair, our goal is to predict the probability of the user clicking on the article. Based on the labels collected in our system logs, we can easily run a logistic or least squares regression and obtain a model for predicting these probabilities. Seems simple enough, why use CB?

The problem with the above approach is that our logistic regression model is only guaranteed to be accurate on the distribution of data which it is trained on. Suppose we have an existing system A that is used to recommend articles to users. Then we know that for the distribution of (user, article) pairs generated by system A, we have a good estimate of click probabilities. Once we have such a model though, we want to use it for selecting the articles that are displayed, presumably based on the largest predicted click probability. Furthermore, this results in the choice of very different articles than the ones picked by system A. In an extreme version, imagine that system A just picks the same article for every user. Then we have no statistical evidence for or against the quality of any other article, and our logistic regression model is completely blind to their quality! Consequently, the training and test errors we will compute based on the data collected under system A will have little connection with the number of clicks we will see online when we deploy our logistic regression model.

Once again, we can contrast with the CB model by examining the notion of regret. In regret, we compare the cumulative reward of our choices, with that of choices made according to any other policy. Consequently, a small regret directly informs us that no other policy in our class can allow us to obtain a substantially higher reward, unlike the test error of logistic regression model above. In the next lecture, we will also see a suited of techniques that set up a more formal training/test style paradigm for contextual bandit problems as an alternative to regret.

6.3 Solution strategies

It turns out that the EXP4 algorithm discussed in the previous lecture already solves the adversarial contextual bandit problem, at least in some cases. We begin with this result, discuss its limitations and then some aspects of the i.i.d. contextual bandit problem.

6.3.1 Adversarial contextual bandits and the EXP4 algorithm

Consider a contextual bandit problem where the policy class consists of only a finite, but very large number of policies. While this might appear limiting—all examples of policy sets we discussed so far are infinite—it typically provides preserves the key elements of the problem. In particular, if the policy is sufficiently large, we can think of the finite set as just a carefully selected *covering* of the bigger, infinite space so that our set contains at least one policy close to any other policy of interest in the infinite space. We refer the reader to [Langford and Zhang, 2008, Syrgkanis et al., 2016] for more details.

Limiting to this setting of a finite policy set now, we create one expert for each policy π. At round t, the expert for π receives x_t and predicts the action $\pi(x_t)$. We can now translate the regret guarantee of EXP4 as saying

$$\max_{\pi\in\Pi}\sum_{t=1}^{T} r_t(\pi(x)) - \sum_{t=1}^{T} r_t = \max_{e}\sum_{t=1}^{T} r_t(e) - \sum_{t=1}^{T} r_t = \mathcal{O}(\sqrt{KT\ln|\Pi|}).$$

Thus the regret bound we know already yields a regret guarantee for contextual bandits under our earlier definition 6.4, and that too in the most general adversarial setting.

The regret guarantee can be further shown to be unimprovable for contextual bandits [Agarwal et al., 2012]. However, there is a serious weakness in the EXP4 algorithm, which has been the key driver behind further research on contextual bandits. Note that the algorithm explicitly maintains a distribution over the policies, and needs to update the probability for each policy on every round. Thus, the computational and storage complexities of EXP4 are $\Omega(|\Pi|)$, in contrast with the logarithmic dependence in regret. This is severely limiting since we motivated the finite policy set setting by thinking of finite, but very large sets. However, the EXP4 algorithm can only handle such sets in the regret bound, with the algorithm being computationally and memory-wise inefficient in such settings.

In the algorithms that follow, we will typically look to obtain similar regret bounds as EXP4, but while keeping the storage and computation also logarithmic in $|\Pi|$ so that large policy sets can be algorithmically handled.

6.3.2 i.i.d. contextual bandits

In order to gain further intuition about the possible computation approaches to the contextual bandit problem, let us start from a simpler i.i.d. version. In this case, there is a fixed, but unknown joint distribution over contexts and rewards. We will use D to denote this joint distribution, as well as the marginal distribution of contexts. We further simplify the problem to a *full-information* version, where the player's interaction happens as described in Figure 6.3.

6.3.2.1 The GREEDY algorithm using full-information

The player's task in the full-information is considerably simpler, as there is no need to explore. While they still only gain the reward for the chosen action, the rewards of all the other actions are also revealed. This allows the algorithm to assess what might have happened if it chose a different action. The Hedge algorithm which was previously introduced, is for instance an admissible algorithm in this setting and can be used to obtain a good regret guarantee (just like EXP4 with partial information). However, due to the i.i.d. nature of the problem, a considerably simpler strategy also works. Let us define the following GREEDY strategy:

1. At time t, let π_t be any policy which satisfies:

$$\pi_t = \arg\max_{\pi\in\Pi}\sum_{s=1}^{t-1} r_s(\pi(x_s)) \tag{6.5}$$

At each round $t = 1, 2, \ldots, T$:

1. Environment reveals a context $x_t \in \mathcal{X}$ sampled from the distribution D.
2. Learner chooses an action $a_t \in \mathcal{A}$.
3. Learner gains reward $r_t(a_t)$, and additionally observes $r_t(a)$ for all $a \in \mathcal{A}$, where $\mathbf{r} \sim D(\cdot|a, x)$.

Figure 6.3: The full-information version of i.i.d. contextual bandit problem

2. Predict $\pi_t(x_t)$.

We begin by showing why this strategy obtains a low regret for the i.i.d. setting. In order prove the result, we will use the following form of Hoeffding's inequality, which can be obtained from the one you saw in the first lecture with some algebra.

Lemma 6.1 (Hoeffding's inequality) *Suppose $X_1, \ldots, X_n$ are independent and $a_i \le X_i \le b_i$. Let $\overline{X} = \sum_{i=1}^n X_i/n$. Then for any $\delta \in (0, 1]$, we have*

$$\left|\overline{X} - \mathbb{E}\left[\overline{X}\right]\right| \le \sqrt{\frac{\sum_{i=1}^n (b_i - a_i)^2}{2n^2} \log \frac{2}{\delta}},$$

with probability at least $1 - \delta$. In the special case where $b_i \equiv 1$ and $a_i \equiv 0$ for all i, the following simpler bound holds with probability at least $1 - \delta$,

$$\left|\overline{X} - \mathbb{E}\left[\overline{X}\right]\right| \le \sqrt{\frac{1}{2n} \log \frac{2}{\delta}}.$$

Using the inequality, we now analyze the *instantaneous regret* of the GREEDY strategy first. The overall regret bound follows as a simple corollary.

Theorem 6.2 *The policy π_t computed at round t of the* GREEDY *algorithm satisfies, with probability at least $1 - \delta$*

$$V(\pi_t) \ge \max_{\pi \in \Pi} V(\pi) - \sqrt{\frac{2}{t} \log \frac{2N}{\delta}}.$$

Proof: For a fixed policy π, the random variables $r_s(\pi(x_s))$ for $s = 1, \ldots, t$ are i.i.d. with mean $V(\pi)$, due to the i.i.d. assumption on the contexts and actions. Since the rewards are further assumed to be in $[0, 1]$ (recall Figure 6.1), we can invoke the simple form of Hoeffding's inequality from Lemma 6.1 to conclude that with probability at least $1 - \delta$,

$$\left|\frac{1}{t} \sum_{s=1}^{t} r_s(\pi(x_s)) - V(\pi)\right| \le \sqrt{\frac{1}{2t} \log \frac{2}{\delta}}. \tag{6.6}$$

Unfortunately, we cannot apply the inequality directly to π_t, which is a policy that is picked after looking at the contexts and rewards over these t rounds. So the random variables $r_s(\pi_t(x_s))$ are not i.i.d. with mean $V(\pi_t)$. A

standard trick to circumvent this difficulty is to establish a *uniform deviation* bound. That is, we show a bound similar to the one above, but simultaneously for all $\pi \in \Pi$. The idea is that if the sample averages and expectations are close for each $\pi \in \Pi$, then they will also be close for π_t in particular. To do this, let $\mathcal{E}(\pi)$ refer to the event that the bound in Equation 6.6 holds, and let $\mathcal{E}(\pi)^C$ be the complement of that event. Equation 6.6 then implies that for any fixed $\pi \in \Pi$, $\mathbb{P}(\mathcal{E}(\pi)^C) \leq \delta$. Observe that we are interested in the probability of $\mathcal{E}(\pi)$ holding for all $\pi \in \Pi$ simultaneously, which means that we want to control

$$\mathbb{P}\left(\cup_{\pi\in\Pi}\mathcal{E}(\pi)^C\right) \leq \sum_{\pi\in\Pi}\mathbb{P}(\mathcal{E}(\pi)^C) \leq N\delta.$$

That is, with probability $1 - N\delta$, Equation 6.6 holds simultaneously for all π, and for π_t in particular. Now, let $\pi^\star$ refer to the best policy in hindsight, that is $\pi^\star = \arg\max_{\pi\in\Pi} V(\pi)$.[1] Then we will invoke Equation 6.6 twice, once with π_t and once with $\pi^\star$. That is, with probability $1 - N\delta$, we have

$$\frac{1}{t}\sum_{s=1}^{t} r_s(\pi_t(x_s)) \geq V(\pi_t) - \sqrt{\frac{1}{2t}\log\frac{2}{\delta}}.$$

$$V(\pi^\star) \geq \frac{1}{t}\sum_{s=1}^{t} r_s(\pi^\star(x_s)) - \sqrt{\frac{1}{2t}\log\frac{2}{\delta}}.$$

Adding the two inequalities, and noticing that the empirical reward of π_t is larger than that of $\pi^\star$ by definition (recall Equation 6.5), we obtain that with probability at least $1 - N\delta$

$$V(\pi_t) \geq \max_{\pi\in\Pi} V(\pi) - \sqrt{\frac{2}{t}\log\frac{2}{\delta}}.$$

Redefining δ/N as δ completes the proof. ■

As mentioned earlier, the regret bound for the *greedy algorithm* follows as a corollary.

Corollary 6.3 *With probability at least $1 - \delta$, the regret of the* GREEDY *algorithm is at most* $\sqrt{8T\log\frac{2NT}{\delta}}$.

The corollary follows by noting that the regret at round t is simply the suboptimality of the policy π_t which is controlled in Theorem 6.2. We can add the terms across rounds, and use the fact that $\sum_{t=1}^{T} 1/\sqrt{t} \leq 2\sqrt{T}$ to obtain the corollary.

We will show in the subsequent lectures how this simple strategy achieves low-regret in the full-information setting. We will also describe how this algorithm can be implemented efficiently, under a particular notion of efficiency.

6.3.2.2 The τ-GREEDY algorithm for bandit feedback

The simplicity of the GREEDY algorithm provokes a natural question. If something so simple works in the full-information setting, is there a natural counterpart in the partial information setting where we only observe the reward of the chosen action. We now describe a relatively direct extension of the GREEDY strategy, called τ-GREEDY which can be executed in a contextual bandit setting. We first introduce the notation $\mathbf{1}(\cdot)$ to be the $0-1$ indicator for whether the argument of it is true or not. The τ-greedy algorithm for contextual bandits is presented in Algorithm 1.

The τ-GREEDY strategy is effectively trying to mimic the GREEDY strategy. It deals with the partial feedback by doing uniformly random exploration on an initial subset of the rounds, and uses the data collected on these rounds to

[1] Note that we can pick any policy that attains this largest value, if it is not unique.

Algorithm 1 The τ-GREEDY algorithm for contextual bandits

Require: Exploration parameter $\tau \in (0, T]$.
  Initialize $S = \emptyset$ to be the empty set.
  for $t = 1, 2, \ldots, \tau$ **do**
    Play a_t uniformly at random from $1, \ldots, K$.
    Update $S = S \cup \{(x_t, a_t, r_t)\}$.
  end for
  Let π_τ be any policy which satisfies:

$$\pi_\tau = \arg\max_{\pi \in \Pi} \sum_{(x,a,r) \in S} r\mathbf{1}(\pi(x) = a). \tag{6.7}$$

  for $t = \tau + 1, \ldots, T$ **do**
    Play $a_t = \pi_\tau(x_t)$.
  end for

assess the quality of each policy. The greedy policy based on this data is then used for the remaining rounds, leading to a clear demarcation between the *explore* and *exploit* phase.

We will now give a regret bound for this algorithm, which formalizes the above intuition.

Theorem 6.4 *The regret of the* τ-GREEDY *algorithm shown in Algorithm 1 is bounded with probability at least* $1 - \delta$ *as* $\mathcal{O}\left((TK)^{2/3}(\log(N/\delta))^{1/3}\right)$.

Proof: Note that the policy π_τ computed in Equation 6.7 is similar to the policy computed in Equation 6.5, except that we only compute it once and on τ examples only. We begin with a slight rewriting of the maximization criterion in Equation 6.7. Let us define the objective at time t as

$$\widehat{R_\tau}(\pi) = \sum_{s=1}^{\tau} r_s \mathbf{1}(\pi(x_s) = a_s).$$

Taking an individual summand, let us define $Y_s = r_s\mathbf{1}(\pi(x_s) = a_s)$. Then we can compute its expectation as

$$\begin{aligned}
\mathbb{E}[Y_s] &= \mathbb{E}\left[r_s\mathbf{1}(\pi(x_s) = a_s)\right] \\
&= \mathbb{E}\left[\mathbb{E}[r_s\mathbf{1}(\pi(x_s) = a_s) \mid x_s]\right] \\
&\overset{(a)}{=} \frac{1}{K}\,\mathbb{E}[\mathbb{E}[r(\pi(x_s)) \mid x_s]] \\
&= \frac{1}{K}V(\pi).
\end{aligned}$$

Here (a) follows since the identity of the chosen action in the explore events is independent of x_s as well as r_s. Hence, adding up the expectations, we see that

$$\mathbb{E}[\widehat{R_\tau}(\pi)] = \frac{\tau}{K}V(\pi),$$

and that it is indeed a sum of τ i.i.d. terms with this mean. That is, we are effectively working with τ/K samples instead of τ samples in defining the greedy policy. We can apply Hoeffdeing's inequality as in the proof of Theorem 6.2

to $\widehat{R_t}(\pi)$ now, as well as combine it with union bound, to see that with probability at least $1-\delta$, simultaneously for all $\pi \in \Pi$

$$\left|\frac{1}{\tau}\widehat{R_\tau}(\pi) - \frac{V(\pi)}{K}\right| \leq \sqrt{\frac{1}{2\tau}\log\frac{2N}{\delta}}.$$

Now we apply the inequality twice as before, once with π_τ and once with $\pi^\star$ to conclude that

$$V(\pi_\tau) \geq V(\pi^\star) - K\sqrt{\frac{2}{\tau}\log\frac{2N}{\delta}}.$$

Note that the regret of the algorithm is now written as the suboptimality of $\pi(\tau)$ on rounds $\tau+1,\ldots,T$ and can be arbitrarily large on the first τ rounds. Hence we bound the regret by

$$\tau + (T-\tau)(V(\pi_\tau) - V(\pi^\star)) \leq \tau + TK\sqrt{\frac{2}{\tau}\log\frac{2N}{\delta}}.$$

Minimizing over τ now yields the theorem statement. ■

This regret bound is significantly inferior than that of EXP4, in that it scales as $\mathcal{O}(T^{2/3})$ instead of $\mathcal{O}(\sqrt{T})$. Can we develop better algorithms which are also computationally efficient in the i.i.d. setting? The next lecture will take a slight digression and introduce some necessary concepts to help us answer this question. We will also see how algorithms like GREEDY and τ-GREEDY can be implemented in a computationally efficient manner in the next lecture.

References

Alekh Agarwal, Miroslav Dudík, Satyen Kale, John Langford, and Robert E Schapire. Contextual bandit learning with predictable rewards. In *International Conference on Artificial Intelligence and Statistics*, pages 19–26, 2012.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2008.

Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert E Schapire. Efficient algorithms for adversarial contextual learning. In *ICML*, 2016.