CSCI 659 Lecture 8

Spring 2026

Instructor: Haipeng Luo

1 Contextual Bandits

In the last lecture, we discussed stochastic linear bandits, which assumes a linear model on the loss of each action. The goal of this lecture is to relax this assumption by considering a more general function class or even allowing arbitrarily losses without any structures. To this end, consider the following general *contextual bandit* problem: at each round $t = 1, \ldots, T$,

1. the environment first decides a *context* $x_t \in \mathcal{X}$ for an arbitrary context space $\mathcal{X}$ and a loss vector $\ell_t \in [0,1]^K$ specifying the loss of K actions;
2. the learner observes the context x_t and then selects an action $a_t \in [K]$;
3. the learner suffers and observes $\ell_t(a_t)$.

Clearly, compared to the standard MAB problem, the only extra element here is the contexts. To explain what the role of these contexts is, we consider the following two settings.

Realizable Setting. One natural role of the context at each round is that it directly determines the loss (or at least its mean) of each action for this round, through some fixed function from a regressor class. This is known as the realizable setting since the loss is realized by some fixed regressor. Formally, let $\mathcal{F} \subseteq (\mathcal{X} \times [K]) \to [0,1]$ be a regressor class known to the learner, and assume that there exists a regressor $f^\star \in \mathcal{F}$, unknown to the learner, such that $\mathbb{E}[\ell_t(a)|x_t] = f^\star(x_t, a)$ for all $t \in [T]$ and $a \in [K]$, where the expectation is with respect to the randomness of the environment when generating ℓ_t. In other words, there is a ground truth $f^\star$ that perfectly predicts the expected loss of each action given the context.

The stochastic linear bandit model discussed last time is essentially a special case of this realized setting, where the context x_t consists of the features $x_t^1, \ldots, x_t^K \in \mathbb{R}^d$ of the actions, that is, $x_t = (x_t^1, \ldots, x_t^K)$, and the regressor class is a set of linear functions: $\mathcal{F} = \{(x,a) \to \langle \theta, x^a \rangle \,|\, \theta \in \mathbb{R}^d, \|\theta\|_2 \le 1\}$. One slight difference is that now for simplicity we consider a fixed number of actions. However, note that this does not mean the action set itself is fixed — each round we can have completely different K actions, encoded via different features (in the recommendation system example, this means that we can still have different items to recommend at each round). In other words, unlike standard MAB, here the index of each action $a \in [K]$ is not meaningful, in the sense that it does not matter if we arbitrarily shuffle the indices at each round, as long as their features are also shuffled in the same way.

By allowing an arbitrary regressor class beyond the linear model, this setting is much more powerful and practical. The realizability assumption also becomes less restricted when we use a richer regressor class such as a set of neural nets. To further relax the realizability assumption, sometimes we also allow a certain level of *misspecification* in the model by changing the assumption to: there exists a fixed $\epsilon \in [0,1]$ and a regressor $f^\star \in \mathcal{F}$ such that $|f^\star(x_t, a) - \mathbb{E}[\ell_t(a)|x_t]| \le \epsilon$ for all $t \in [T]$ and $a \in [K]$. The no-misspecification case discussed earlier is clearly just a special case with $\epsilon = 0$.

How should we measure the learner's performance in this problem? Note that each regressor $f \in \mathcal{F}$ naturally defines a *policy* $\pi_f : \mathcal{X} \to [K]$ that selects the action with the lowest expect loss for the

given context, that is, $\pi_f(x) = \operatorname{argmin}_{a\in[K]} f(x,a)$. It is then natural to measure the learner's performance against the best policy, via the following pseudo-regret:

$$\bar{\mathcal{R}}_T = \max_{\pi\in\Pi} \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(a_t) - \sum_{t=1}^{T} \ell_t(\pi(x_t))\right], \tag{1}$$

where $\Pi = \{\pi_f : f \in \mathcal{F}\}$ is the set of all policies induced by $\mathcal{F}$. When there is no misspecification, it is clear that pseudo-regret becomes

$$\bar{\mathcal{R}}_T = \max_{\pi\in\Pi} \mathbb{E}\left[\sum_{t=1}^{T} f^\star(x_t,a_t) - \sum_{t=1}^{T} f^\star(x_t,\pi(x_t))\right] = \mathbb{E}\left[\sum_{t=1}^{T} f^\star(x_t,a_t) - \sum_{t=1}^{T} f^\star(x_t,\pi^\star(x_t))\right], \tag{2}$$

where we use $\pi^\star$ as a shorthand for $\pi_{f^\star}$.

Agnostic Setting. Note that when the misspecification level ϵ is as bad as 1, that is, when there are no reasonable regressors at all, it does not mean there is nothing we can learn at all, because a policy can be bad at predicting the exact value of the losses but at the same time somehow still good at identifying the good actions most of the time. In this case, the corresponding regressor that a policy uses to make it decisions is not that important anymore, and what really matters is only the policy itself, that is, what action it takes under each context. Motivated by this, we consider the *agnostic setting* where there is no explicit relation between the contexts and the losses at all, and the learner's goal is, given a fixed policy class $\Pi \subseteq \mathcal{X} \to [K]$, to compete with what the best policy from this class can do, that is, to minimize the same pseudo-regret defined in Eq. (1) (with Π now being an arbitrary given policy class).

It is clear that the realizable setting is a special case where $\Pi = \{\pi_f : f \in \mathcal{F}\}$ and there is an additional assumption on how the contexts and losses are connected via a ground truth $f^\star$. The standard MAB is also a special case where Π contains only K policies, the a-th of which always selects action a regardless of the context.

Naive Approaches. Note that the most interesting scenario is when the context space $\mathcal{X}$ is large (and potentially high-dimensional), and so is the regressor class $\mathcal{F}$ or the policy class Π; just consider again the example where $\mathcal{F}$ is a set of neural nets. We will thus not make any restriction on the size of $\mathcal{X}$ (could even be infinity), and consider the case where the size of $\mathcal{F}$ or Π, denoted by N and assumed to be finite in most discussions for simplicity, is huge so that only $\text{poly}(\ln N)$ dependence is acceptable for the regret guarantees (and also the computational complexity; more on this to follow). This makes the following two naive approaches uninteresting.

First, one can see contextual bandits as a combination of many independent K-armed bandit problems, one for each distinct context. In other words, for each possible $x \in \mathcal{X}$, we maintain a K-armed bandit algorithm, and at time t, we use the algorithm corresponding to context x_t to make the decision. This is clearly not a good approach since it is possible that we do not even see the same context twice over T rounds, in which case there is no learning at all in this approach as each K-armed bandit algorithm is run for at most one round.

Another approach is to treat each policy as an arm, making this an N-armed bandit problem. Indeed, at each time t, deciding which arm/policy π to pull naturally tells us which action a_t to play by following the suggestion of this policy given the current context, that is, $a_t = \pi(x_t)$. Then, we can treat the observation $\ell(a_t)$ as the loss for this arm. The issue is of course that this leads to regret of order $\sqrt{TN}$, which is unacceptable due to the polynomial dependence on N.

2 Adversarial Contextual Bandits

To get a sense of what a good regret guarantee looks like for contextual bandits, we start by analyzing a simple (but inefficient) algorithm for the most general case: agnostic setting with adversarial contexts and losses.

The algorithm is in fact very close to the first naive approach mentioned in the last section. At time t, we compute a distribution $P_t \in \Delta(N)$ over the policies using Hedge and the past loss estimators

$\widehat{\ell}_1, \ldots, \widehat{\ell}_{t-1} \in \mathbb{R}_+^N$: $P_t(\pi) \propto \exp\left(-\eta \sum_{s<t} \widehat{\ell}_s(\pi)\right)$. Then, we sample a policy $\pi_t \sim P_t$ and play action $a_t = \pi_t(x_t)$, which is equivalent to sampling action a_t based on distribution $p_t \in \Delta(K)$ where $p_t(a) = \sum_{\pi \in \Pi: \pi(x)=a} P_t(\pi)$.

The key difference is how we construct estimator $\widehat{\ell}_t$ based on the feedback $\ell_t(a_t)$. What the naive approach does is $\widehat{\ell}_t(\pi) = \frac{\ell_t(\pi(x_t))}{P_t(\pi)} \mathbf{1}\{\pi = \pi_t\}$, completely following Exp3's importance weighted estimator by treating each policy as an arm. This unbiased estimator is non-zero only for the selected policy, and has a large variance: $\mathbb{E}_t[\widehat{\ell}_t(\pi)^2] = \frac{\ell_t(\pi(x_t))^2}{P_t(\pi)} \leq \frac{1}{P_t(\pi)}$.

However, note that the observation $\ell_t(a_t)$ tells us not only the loss for the selected policy, but also the loss for all other policies that selects the same action a_t. In other words, the missing information is really just the $K-1$ unobserved coordinates of ℓ_t, instead of $N-1$ values for the unselected policies. If we consider the problem in this way, then it is more natural to construct a K-dimensional estimator for ℓ_t, which we denote by the same notation $\widehat{\ell}_t \in \mathbb{R}_+^K$ with a slight abuse of notation (thus, whether $\widehat{\ell}_t$ is N dimensional or K dimensional depends on whether its input is a policy π or an action a). The construction is also by the same idea of inverse importance weighting: $\widehat{\ell}_t(a) = \frac{\ell_t(a)}{p_t(a)} \mathbf{1}\{a = a_t\}$. Then we set the estimator $\widehat{\ell}_t(\pi)$ for policy π naturally as $\widehat{\ell}_t(\pi(x_t))$. Compared to the earlier naive estimator, this new one is also unbiased ($\mathbb{E}_t[\widehat{\ell}_t(\pi)] = \ell_t(\pi(x_t))$), but with a smaller variance:

$$\mathbb{E}_t[\widehat{\ell}_t(\pi)^2] = p_t(\pi(x_t)) \times \frac{\ell_t(\pi(x_t))^2}{p_t(\pi(x_t))^2} + (1 - p_t(\pi(x_t))) \times 0 = \frac{\ell_t(\pi(x_t))^2}{p_t(\pi(x_t))} \leq \frac{1}{p_t(\pi(x_t))}, \quad (3)$$

since $p_t(\pi(x_t))$ is by definition at least $P_t(\pi)$ (in fact, the former is usually much larger).

The resulting algorithm is called Exp4 [Auer et al., 2002], short for *Exponential-weight algorithm for Exploration and Exploitation using Expert advice* (since the role of each policy is similar to an expert as in the expert problem). We summarize the algorithm in the pseudocode below, followed by its formal regret guarantee.

Algorithm 1: Exp4

Input: learning rate $\eta > 0$

for $t = 1, \ldots, T$ **do**

- compute $P_t \in \Delta(N)$ such that $P_t(\pi) \propto \exp\left(-\eta \sum_{s<t} \widehat{\ell}_s(\pi(x_s))\right)$
- sample a_t from $p_t \in \Delta(K)$ where $p_t(a) = \sum_{\pi \in \Pi: \pi(x)=a} P_t(\pi)$
- observe $\ell_t(a_t)$ and construct $\widehat{\ell}_t \in \mathbb{R}_+^K$ such that $\widehat{\ell}_t(a) = \frac{\ell_t(a)}{p_t(a)} \mathbf{1}\{a = a_t\}$

Theorem 1. *With $\eta = \sqrt{\frac{\ln N}{TK}}$, Exp4 ensures $\bar{\mathcal{R}}_T = \mathcal{O}(\sqrt{TK \ln N})$.*

Proof. Similar to the Exp3 analysis, the proof starts from the following local-norm bound for Hedge, which holds due to the nonnegativity of loss estimator: for any $\pi^\star$,

$$\sum_{t=1}^T \left\langle P_t, \widehat{\ell}_t \right\rangle - \sum_{t=1}^T \widehat{\ell}_t(\pi^\star) \leq \frac{\ln N}{\eta} + \eta \sum_{t=1}^T \sum_{\pi \in \Pi} P_t(\pi) \widehat{\ell}_t(\pi)^2.$$

By the unbiasedness of $\widehat{\ell}_t$, we have

$$\mathbb{E}_t[\langle P_t, \widehat{\ell}_t \rangle] = \sum_{\pi \in \Pi} P_t(\pi) \ell_t(\pi(x_t)) = \sum_{a=1}^K \sum_{\pi: \pi(x_t)=a} P_t(\pi) \ell_t(a) = \sum_{a=1}^K p_t(a) \ell_t(a) = \mathbb{E}_t[\ell_t(a_t)],$$

and $\mathbb{E}_t[\widehat{\ell}_t(\pi^\star)] = \ell_t(\pi^\star(x_t))$. On the other hand, the expectation of the local-norm term can be bounded by K just as Exp3 (instead of N), thanks to the smaller variance of the estimator shown earlier in Eq. (3):

$$\mathbb{E}_t\left[\sum_{\pi \in \Pi} P_t(\pi) \widehat{\ell}_t(\pi)^2\right] \leq \sum_{\pi \in \Pi} \frac{P_t(\pi)}{p_t(\pi(x_t))} = \sum_{a=1}^K \sum_{\pi: \pi(x_t)=a} \frac{P_t(\pi)}{p_t(a)} = \sum_{a=1}^K \frac{p_t(a)}{p_t(a)} = K.$$

Therefore, taking expectation on both sides, we obtain $\bar{\mathcal{R}}_T \leq \frac{\ln N}{\eta} + \eta TK$, which is $\mathcal{O}(\sqrt{TK \ln N})$ after picking the optimal η. □

This proof shows again the importance of getting a right combination of local-norm and loss estimators. The regret bound $\mathcal{O}(\sqrt{TK \ln N})$ has only logarithmic dependence on N, same as Hedge for the expert problem, and $\sqrt{K}$ dependence on the number of actions, same as Exp3 for MAB. In fact, this is known to be near optimal for adversarial contextual bandits; the optimal bound for this problem turns out to be $\mathcal{O}\left(\sqrt{TK \ln \frac{N}{K}}\right)$, recently shown by Chase et al. [2025].

Question 1. *What happens if we instead use FTRL with Tsallis entropy to determine P_t?*

While the regret bound depends on N only logarithmically, one obvious issue of Exp4 is that its time/space complexity is linear in N. Just like the expert problem, generally there is no way to get pass this barrier unfortunately. To resolve this issue, we need to assume some *oracle access* to the regressor class $\mathcal{F}$ or the policy class Π, and derive algorithms that never search over all policies explicitly and instead only make use of the regressor/policy class through such oracles. These oracles themselves might not be efficiently implementable exactly, but are often considered as algorithmic primitives that are widely and approximately implemented in practice anyway using highly efficient heuristics. The question then becomes, given such oracles, how to derive algorithms that make a small number of oracle calls while ensuring low regret (ideally comparable to the optimal bound $\mathcal{O}(\sqrt{TK \ln N})$). This turns out to be a highly nontrivial problem, and in the remaining of this lecture we will discuss some examples from recent research.

3 Realizable Setting with an Online Regression Oracle

Consider the realizable setting with a regressor class $\mathcal{F}$, and assume that the learner is given access to an *online regression oracle* $\mathcal{O}_{sq}$ that solves the following online regression problem:

1. the environment decides and reveals an instance $z_t \in \mathcal{X} \times [K]$;
2. the oracle $\mathcal{O}_{sq}$ predicts a value $\widehat{y}_t \in [0, 1]$;
3. the environment reveals the true value $y_t \in [0, 1]$.

By "solving", we mean that $\mathcal{O}_{sq}$ satisfies the following regret guarantee in terms of square loss: for any (possibly adaptively chosen) sequence $z_{1:T}$ and $y_{1:T}$,

$$\sum_{t=1}^{T} (\widehat{y}_t - y_t)^2 - \min_{f \in \mathcal{F}} \sum_{t=1}^{T} (f(z_t) - y_t)^2 \leq B(T) \tag{4}$$

for some regret bound $B(T) = o(T)$. Below are some examples (in fact, in all these examples, the oracle satisfies Eq. (4) even without the realizable assumption):

- (**Finite Class**) When $\mathcal{F}$ is finite, Hedge is one such oracle with regret $B(T) = \mathcal{O}(\sqrt{T \ln N})$. In fact, because of the nice properties of square loss (in particular its exp-concavity), it can be shown that Hedge with a constant learning rate achieves only $\mathcal{O}(\ln N)$ regret in this case.
- (**Linear Class**) When $\mathcal{F}$ is the linear model mentioned earlier: $\mathcal{F} = \{z = (x, a) \to \langle \theta, x^a \rangle \,|\theta \in \mathbb{R}^d, \|\theta\|_2 \leq 1\}$, the online regression problem is just an instance of OCO, and thus OGD is one such oracle with regret $B(T) = \mathcal{O}(\sqrt{T})$ as long as the L_2 norm of the contexts is bounded (recall that there is no d dependence at all). Again, thanks to the exp-concavity of square loss, there exist other algorithms (such as Online Newton Step [Hazan et al., 2007]) with $B(T) = \mathcal{O}(d \ln T)$ regret in this case. Importantly, these algorithms are all efficiently implementable (with poly(d) dependence per round).

While there are not too many examples beyond the linear model where such an oracle can be implemented efficiently, the point is really that regression is so basic in machine learning practice and people have been successfully using complicated regressor class such as neural nets (by simply running gradient descent) to "solve" this problem already, without worrying about whether it truly enjoys low regret or not. Therefore, assuming such an oracle is practically reasonable. Doing so also allows any future advances in regression to be immediately translated to solving contextual bandits.

Algorithm Design. To solve contextual bandits using such an oracle, consider the following natural algorithm framework. At time t, given context x_t, ask the oracle $\mathcal{O}_{sq}$ to predict the loss for each of the K actions, denoted by $\widehat{y}_t(1), \ldots, \widehat{y}_t(K)$. Then, based on these predictions, come up with a distribution $p_t \in \Delta(K)$, sample $a_t \sim p_t$, observe $\ell_t(a_t)$, and finally feed the instance (x_t, a_t) along with its true value $\ell_t(a_t)$ to the oracle for it to perform some internal updates. It remains to figure out the construction of p_t.

To this end, first observe that under this framework, in expectation the square loss regret of the oracle against the ground truth $f^\star$ can be written as:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{T}(\widehat{y}_t(a_t) - \ell_t(a_t))^2 - \sum_{t=1}^{T}(f^\star(x_t, a_t) - \ell_t(a_t))^2\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{T}(\widehat{y}_t(a_t) - f^\star(x_t, a_t))(\widehat{y}_t(a_t) + f^\star(x_t, a_t) - 2\ell_t(a_t))\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{T}(\widehat{y}_t(a_t) - f^\star(x_t, a_t))^2\right] = \mathbb{E}\left[\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)(\widehat{y}_t(a) - f^\star(x_t, a))^2\right] \qquad (5)
\end{aligned}$$

where the second step is because $\ell_t(a_t)$ is in expectation equal to $f^\star(x_t, a_t)$ due to the realizable assumption. On the other hand, in light of Eq. (2), the pseudo-regret of the learner is $\mathbb{E}[\sum_{t=1}^{T}(\sum_{a=1}^{K} p_t(a) f^\star(x_t, a) - f^\star(x_t, \pi^\star(x_t)))]$. Therefore, one natural goal is to design p_t so that $\sum_{a=1}^{K} p_t(a) f^\star(x_t, a) - f^\star(x_t, \pi^\star(x_t))$ is connected to $\sum_{a=1}^{K} p_t(a)(\widehat{y}_t(a) - f^\star(x_t, a))^2$, for example, in the form that the former is bounded by some fixed factor times the latter plus a small $o(1)$ term. Let this factor be $\gamma/4$ for some parameter $\gamma > 0$ to be specified later (the constant 4 is only for convenience as it will become clear). To understand the best way to design p_t and what the $o(1)$ small term is, we study the following optimization problem for any given $\widehat{y} \in \mathbb{R}^K$:

$$\mathrm{OPT}(\widehat{y}) = \min_{p \in \Delta(K)} \max_{a^\star \in [K]} \max_{\mu \in \mathbb{R}^K} \langle p - e_{a^\star}, \mu \rangle - \frac{\gamma}{4} \|\widehat{y} - \mu\|^2_{\mathrm{diag}(p)} .$$

Here, $\widehat{y}$ corresponds to the prediction vector of the oracle $\widehat{y}_t$, p corresponds to the distribution p_t we want to find, $a^\star$ corresponds to the best action $\pi^\star(x_t)$, $\mu \in \mathbb{R}^K$ corresponds to the true loss mean vector $f^\star(x_t, \cdot)$, $e_{a^\star} \in \mathbb{R}^K$ is the standard basis vector with the $a^\star$-th coordinate being 1, and $\mathrm{diag}(p)$ is a diagonal matrix with p on the diagonal. It is then clear that, if we let p_t to be the argmin of the problem defined in $\mathrm{OPT}(\widehat{y}_t)$, then

$$\sum_{a=1}^{K} p_t(a) f^\star(x_t, a) - f^\star(x_t, \pi^\star(x_t)) \le \frac{\gamma}{4}\sum_{a=1}^{K} p_t(a)(\widehat{y}_t(a) - f^\star(x_t, a))^2 + \mathrm{OPT}(\widehat{y}_t),$$

and thus the overall regret is connected to the square loss regret of the oracle as:

$$\bar{\mathcal{R}}_T \le \mathbb{E}\left[\frac{\gamma}{4}\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)(\widehat{y}_t(a) - f^\star(x_t, a))^2 + \sum_{t=1}^{T}\mathrm{OPT}(\widehat{y}_t)\right] \le \frac{\gamma}{4}B(T) + \mathbb{E}\left[\sum_{t=1}^{T}\mathrm{OPT}(\widehat{y}_t)\right], \qquad (6)$$

where the second step uses Eq. (5) and the assumption of the oracle Eq. (4). It now all boils down to solving $\mathrm{OPT}(\widehat{y})$. Interestingly, the solution turns out to be connected to FTRL with learning rate γ and a special regularizer known as *log barrier*.

Lemma 1. *For any $\widehat{y} \in \mathbb{R}^K$, we have $\mathrm{OPT}(\widehat{y}) = \frac{K-1}{\gamma}$, and the corresponding minimizer is $p^\star = \mathrm{argmin}_{p \in \Delta(K)} \langle p, \widehat{y} \rangle + \frac{1}{\gamma}\sum_{a=1}^{K} \ln \frac{1}{p(a)}$, or equivalently $p^\star(a) = \frac{1}{\gamma(\widehat{y}(a) + \lambda)}$ where λ is such that $p^\star$ is a distribution (and can be found efficiently using a binary search).*

Proof. Let's first fix p and $a^\star$, and work on the innermost problem: $\max_{\mu \in \mathbb{R}^K} \langle p - e_{a^\star}, \mu \rangle - \frac{\gamma}{4}\|\widehat{y} - \mu\|^2_{\mathrm{diag}(p)}$. This is simply a quadratic in μ, with maximizer $\mu^\star = \frac{2}{\gamma}\mathrm{diag}(p)^{-1}(p - e_{a^\star}) + \widehat{y}$ (verify it by setting the gradient to zero). The maximum is

$$\frac{1}{\gamma}\|p - e_{a^\star}\|^2_{\mathrm{diag}(p)^{-1}} + \langle p - e_{a^\star}, \widehat{y} \rangle = \frac{1}{\gamma}\sum_{a=1}^{K} \frac{(p(a) - \mathbf{1}\{a = a^\star\})^2}{p(a)} + \langle p - e_{a^\star}, \widehat{y} \rangle$$

$$= \frac{1}{\gamma}\sum_{a=1}^{K}\left(p(a) - 2\mathbf{1}\{a=a^\star\} + \frac{\mathbf{1}\{a=a^\star\}}{p(a)}\right) + \langle p - e_{a^\star}, \widehat{y}\rangle$$
$$= \frac{1}{\gamma}\left(\frac{1}{p(a^\star)} - 1\right) + \langle p - e_{a^\star}, \widehat{y}\rangle\,.$$

Then, we focus on solving the rest of the problem:

$$\min_{p\in\Delta(K)}\max_{a^\star\in[K]}\frac{1}{\gamma}\left(\frac{1}{p(a^\star)} - 1\right) + \langle p - e_{a^\star}, \widehat{y}\rangle\,. \tag{7}$$

We claim that the optimal p is such that the objective above is the same for all $a^\star$ (in which case the part $\max_{a^\star\in[K]}$ becomes trivial). To see this, simply note that

$$\max_{a^\star\in[K]}\frac{1}{\gamma}\left(\frac{1}{p(a^\star)} - 1\right) + \langle p - e_{a^\star}, \widehat{y}\rangle \geq \mathbb{E}_{a^\star\sim p}\left[\frac{1}{\gamma}\left(\frac{1}{p(a^\star)} - 1\right) + \langle p - e_{a^\star}, \widehat{y}\rangle\right]$$
$$= \frac{1}{\gamma}\sum_{a=1}^{K} p(a)\left(\frac{1}{p(a)} - 1\right) = \frac{K-1}{\gamma}, \qquad (\mathbb{E}_{a^\star\sim p}\left[\langle p - e_{a^\star}, \widehat{y}\rangle\right] = 0)$$

and the inequality becomes equality when $\frac{1}{\gamma}\left(\frac{1}{p(a^\star)} - 1\right) + \langle p - e_{a^\star}, \widehat{y}\rangle$ is the same for all $a^\star$. Ignoring all terms that are independent of $a^\star$, we see that this is the same as saying that $p^\star$ should make sure $\frac{1}{\gamma p(a^\star)} - \widehat{y}(a^\star) = \lambda$ for all $a^\star \in [K]$, where λ is such that $p^\star$ is a valid distribution. This is the same as $p^\star = \operatorname{argmin}_{p\in\Delta(K)}\langle p, \widehat{y}\rangle + \frac{1}{\gamma}\sum_{a=1}^{K}\ln\frac{1}{p(a)}$ since it is exactly equivalent to setting the gradient of the Lagrangian of this FTRL objective to zero. □

The FTRL form of the solution is quite intriguing — instead of using the cumulative estimated losses in FTRL (which does not really make sense for contextual bandits), here we use the predicted losses from the regression oracle, importantly with a special log-barrier regularizer. The analysis also becomes quite different as we have seen. In fact, another difference is that there is no need to solve FTRL very accurately — any approximate solution to $\mathrm{OPT}(\widehat{y})$ or equivalently Eq. (7) with a constant approximation factor suffices. We provide one such approximate solution below, which is only two times worse compared to the optimal one but importantly enjoys a closed form.

Lemma 2. *For any $\widehat{y} \in \mathbb{R}^K$, let $b = \operatorname{argmin}_a \widehat{y}(a)$ and $\mathring{p} \in \Delta(K)$ be such that $\mathring{p}(a) = \frac{1}{\gamma(\widehat{y}(a)-\widehat{y}(b))+K}$ for $a \neq b$ (the definition of $\mathring{p}(b)$ is implicit via $1 - \sum_{a\neq b} p^\star(a)$). Then $\mathring{p}$ approximately solves Eq. (7) with an approximation factor of 2.*

Proof. This is by direct calculations. Fix any $a^\star$, let's first work on the part $\langle \mathring{p} - e_{a^\star}, \widehat{y}\rangle$:

$$\langle \mathring{p} - e_{a^\star}, \widehat{y}\rangle = \langle e_b - e_{a^\star}, \widehat{y}\rangle + \langle \mathring{p} - e_b, \widehat{y}\rangle$$
$$= \widehat{y}(b) - \widehat{y}(a^\star) + \sum_{a\neq b}\mathring{p}(a)(\widehat{y}(a) - \widehat{y}(b)) \leq \widehat{y}(b) - \widehat{y}(a^\star) + \frac{K-1}{\gamma},$$

where the last step uses the definition of $\mathring{p}(a)$ for $a \neq b$. Therefore, the objective in Eq. (7) is at most

$$\frac{1}{\gamma}\left(\frac{1}{\mathring{p}(a^\star)} - 1\right) + \widehat{y}(b) - \widehat{y}(a^\star) + \frac{K-1}{\gamma}.$$

When $a^\star = b$, this is at most $\frac{2(K-1)}{\gamma}$ since $\mathring{p}(b) \geq 1/K$; when $a^\star \neq b$, plugging in the definition of $\mathring{p}(a^\star) = \frac{1}{\gamma(\widehat{y}(a^\star)-\widehat{y}(b))+K}$, we get exactly $\frac{2(K-1)}{\gamma}$. Recalling that the optimal value is $\frac{(K-1)}{\gamma}$ according to Lemma 1 finishes the proof. □

Based on all these discussions, we summarize the final algorithm, called SquareCB [Foster and Rakhlin, 2020], in the pseudocode below, followed by its formal regret guarantees.

Algorithm 2: SquareCB

Input: online regression oracle $\mathcal{O}_{sq}$ and parameter $\gamma > 0$
for $t = 1, \dots, T$ **do**
receive context x_t
ask $\mathcal{O}_{sq}$ to predict the loss for each action, denoted by $\widehat{y}_t(1), \dots, \widehat{y}_t(K)$
compute $p_t \in \Delta(K)$ such that

$$\begin{cases} \text{Option I:} & p_t(a) = \frac{1}{\gamma(\widehat{y}_t(a)+\lambda)} \text{ for all } a, \text{ where } \lambda \text{ is found via a binary search} \\ \text{Option II:} & p_t(a) = \frac{1}{\gamma(\widehat{y}_t(a)-\min_b \widehat{y}_t(b))+K} \text{ for } a \neq \operatorname{argmin}_b \widehat{y}_t(b) \end{cases}$$

sample $a_t \sim p_t$, observe $\ell_t(a_t)$, and feed $((x_t, a_t), \ell_t(a_t))$ to $\mathcal{O}_{sq}$

Theorem 2. *Under assumption (4), SquareCB (either option) ensures* $\bar{\mathcal{R}}_T = \mathcal{O}(\gamma B(T) + \frac{TK}{\gamma})$, *which is* $\mathcal{O}(\sqrt{TB(T)K})$ *after picking the optimal* γ.

Proof. This is by directly combining Eq. (6) and Lemma 1 for Option I or Lemma 2 for Option II. □

Note that the bound $\mathcal{O}(\sqrt{TB(T)K})$ is always sublinear as $B(T) = o(T)$. Applying this result to the earlier examples, we get the following conclusions:

- (**Finite Class**) Recall that Hedge with a constant learning rate ensures $B(T) = \ln N$ for a finite regressor class. Thus, SquareCB with this oracle ensures $\bar{\mathcal{R}}_T = \mathcal{O}(\sqrt{TK \ln N})$, exactly matching that of Exp4 (but note that the former only works for the realizable setting).
- (**Linear Class**) For the linear regressor class, if we use OGD as the oracle, then $B(T) = \mathcal{O}(\sqrt{T})$ and SquareCB ensures $\bar{\mathcal{R}}_T = \mathcal{O}(T^{3/4}\sqrt{K})$. While this has a worse dependence on T compared to LinUCB's regret $\widetilde{\mathcal{O}}(d\sqrt{T})$, note that it has no dependence on d at all and is thus suitable for high-dimensional problems as long as the number of actions K is not too large. Another benefit is that the resulting algorithm is much more efficient than LinUCB since its per-round time complexity is linear in d while LinUCB requires at least d^2 time. If we instead use the aforementioned Online Newton Step as the oracle with $B(T) = \mathcal{O}(d \ln T)$, then SquareCB ensures $\widetilde{\mathcal{O}}(\sqrt{dTK})$, closer to what LinUCB achieves.

Robustness to Misspecification. While we have focused on the realizable setting, SquareCB turns out to be highly robust to misspecification. Recall that in this case, $|f^\star(x_t, a) - \mathbb{E}[\ell_t(a)|x_t]| \leq \epsilon$ holds for some misspecification level $\epsilon \in [0, 1]$. By just slightly adapting the analysis, we obtain:

Theorem 3. *In the* ϵ*-misspecification setting, SquareCB (with either option) ensures* $\bar{\mathcal{R}}_T = \mathcal{O}(\gamma(B(T) + \epsilon^2 T) + \frac{TK}{\gamma} + \epsilon T)$, *which is* $\mathcal{O}(\sqrt{TB(T)K} + \epsilon T\sqrt{K})$ *with the optimal* γ.

Proof. First, revisiting how we obtained Eq. (2) from Eq. (1), it is clear that with misspecification, the regret should become $\bar{\mathcal{R}}_T \leq \mathbb{E}\left[\sum_{t=1}^T f^\star(x_t, a_t) - \sum_{t=1}^T f^\star(x_t, \pi^\star(x_t))\right] + 2\epsilon T$. Then, we revise Eq. (5) as

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^T (\widehat{y}_t(a_t) - \ell_t(a_t))^2 - \sum_{t=1}^T (f^\star(x_t, a_t) - \ell_t(a_t))^2\right] \\
&= \mathbb{E}\left[\sum_{t=1}^T (\widehat{y}_t(a_t) - f^\star(x_t, a_t))(\widehat{y}_t(a_t) + f^\star(x_t, a_t) - 2\ell_t(a_t))\right] \\
&\geq \mathbb{E}\left[\sum_{t=1}^T (\widehat{y}_t(a_t) - f^\star(x_t, a_t))^2\right] - 2\epsilon\mathbb{E}\left[\sum_{t=1}^T |\widehat{y}_t(a_t) - f^\star(x_t, a_t)|\right] \\
&\geq \frac{1}{2}\mathbb{E}\left[\sum_{t=1}^T (\widehat{y}_t(a_t) - f^\star(x_t, a_t))^2\right] - 2\epsilon^2. \qquad (\alpha\beta \leq \alpha^2 + \tfrac{1}{4}\beta^2)
\end{aligned}$$

Finally, by Lemma 1 or Lemma 2, the way we select p_t still ensures:

$$\mathbb{E}\left[\sum_{t=1}^{T} f^\star(x_t, a_t) - \sum_{t=1}^{T} f^\star(x_t, \pi^\star(x_t))\right] \le \frac{\gamma}{4}\mathbb{E}\left[\sum_{t=1}^{T}(\widehat{y}_t(a_t) - f^\star(x_t, a_t))^2\right] + \mathcal{O}\left(\frac{TK}{\gamma}\right).$$

Combining all steps proves $\bar{\mathcal{R}}_T = \mathcal{O}(\gamma(B(T) + \epsilon^2 T) + \frac{TK}{\gamma} + \epsilon T)$. □

While paying additional $\mathcal{O}(\epsilon T)$ regret is very natural in the ϵ-misspecification setting, one should not take such robustness guarantee for granted. For example, it can be shown that LinUCB is not robust and could suffer linear regret for some $\epsilon = o(1)$; see Lattimore et al. [2020]. Also, note that the optimal tuning of γ requires the knowledge of the misspecification level ϵ, which is unrealistic in practice. Fortunately, this can be addressed by learning over multiple copies of the SquareCB algorithm, each with a different guess on ϵ; see Foster et al. [2020].

4 Offline Oracles

We briefly mention two other types of oracle and the corresponding approaches. While the online regression oracle is natural, it is after all solving a non-trivial online problem. An even more basic oracle would be one that only solves the corresponding *offline* problem. For example, an offline regression oracle solves the following problem

$$\operatorname*{argmin}_{f \in \mathcal{F}} \sum_{(x,a,y) \in \mathcal{S}} (f(x,a) - y)^2$$

for any given dataset $\mathcal{S} \subseteq \mathcal{X} \times [K] \times [0,1]$. In other words, it solves the problem defining the benchmark in Eq. (4), which is indeed a fairly standard offline regression problem. In light of SquareCB, a natural approach to solve contextual bandit using this oracle would be: at each time t, feed the previous observed data to the oracle, use the returned regressor to predict the loss of each action given context x_t, and finally construct a distribution $p_t \in \Delta(K)$ in the same way as SquareCB, from which a_t is sampled. This algorithm was introduced by Simchi-Levi and Xu [2021], and it enjoys $\widetilde{\mathcal{O}}(\sqrt{TK \ln N})$ regret for a finite $\mathcal{F}$, almost the same as SquareCB, except that the analysis relies on an additional assumption that the contexts $x_1, \dots, x_T$ are drawn from a fixed and unknown distribution (while SquareCB allows them to be even chosen by an adaptive adversary). It was also shown that there is no need to call the oracle at every round; in fact, only $\mathcal{O}(\ln(\ln T))$ total number of calls over T rounds is needed to achieve the same regret bound.

The oracle-based approaches discussed so far only work for the realizable setting. To handle the agnostic setting with an arbitrary policy class Π, another type of offline oracle was proposed, which solves the following *cost-sensitive classification* problem:

$$\operatorname*{argmin}_{\pi \in \Pi} \sum_{(x,\ell) \in \mathcal{S}} \ell(\pi(x))$$

for any given dataset $\mathcal{S} \subset \mathcal{X} \times \mathbb{R}^K$. This is a cost-sensitive classification problem since each policy π is essentially predicting the "class" for each context x out of K possibilities, and the cost for predicting each class is different as specified by the cost vector ℓ. This is also just the problem defining the benchmark in the regret definition of Eq. (1).

To solve contextual bandits using this classification oracle, earlier works consider a stochastic setting where $(x_1, \ell_1), \dots, (x_T, \ell_T)$ are independent samples of a fixed joint distribution over $\mathcal{X} \times [0,1]^K$. To construct a dataset fed to the oracle by only observing x_t and $\ell_t(a_t)$ at time t, it is natural to construct the same importance weighted estimator $\widehat{\ell}_t \in \mathbb{R}_+^K$ as in Exp3/Exp4. Suppose that at time t, we feed $(x_1, \widehat{\ell}_1), \dots, (x_{t-1}, \widehat{\ell}_{t-1})$ to the oracle and obtain π_t as the returned policy. Simply picking $a_t = \pi_t(x_t)$ is a bad idea since there is no exploration at all (also note that randomness is required for constructing importance weighted estimators). Instead, one can set aside a small probability to uniformly explore all actions (and pick $\pi_t(x_t)$ with the rest of the probability). Because of this simple exploration scheme, however, it is not difficult to show that this approach at best achieves $\widetilde{\mathcal{O}}(T^{2/3}(K \ln N)^{1/3})$ regret.

How to improve the regret to $\widetilde{\mathcal{O}}(\sqrt{TK\ln N})$ was answered by the work [Agarwal et al., 2014]. Interestingly, the solution is again related to FTRL with the log-barrier regularizer (although not explicitly mentioned in the original work). Specifically, at time t, we find a distribution P_t over the policies via:

$$P_t = \operatorname*{argmin}_{P\in\Delta(N)} \left\langle P, \sum_{s<t} \widehat{\ell}_s \right\rangle + \frac{1}{\gamma} \sum_{s<t} \sum_{a=1}^{K} \ln \frac{1}{P(a|x_s)}$$

where we recall the overloading of the notation $\widehat{\ell}_s \in \mathbb{R}_+^N$ used earlier for Exp4 (so that $\widehat{\ell}_s(\pi) = \widehat{\ell}_s(\pi(x_s))$), and use $P(a|x)$ to denote $\sum_{\pi:\pi(x)=a} P(\pi)$. With this distribution P_t, we simply sample a_t from $P_t(\cdot|x_t)$. The only difference between Exp4 and this algorithm is in the regularizer — Exp4 uses the entropy regularizer in the policy space: $\sum_\pi P(\pi)\ln P(\pi)$, while this algorithm uses a log-barrier regularizer in the action space induced by the previous observed contexts: $\sum_{s<t}\sum_{a=1}^{K} \ln \frac{1}{P(a|x_s)}$. It is this important difference that makes this algorithm efficiently implementable, even though it is written in the space of the policy class and a naive implementation would require at least $\mathcal{O}(N)$ time. Specifically, it turns out that one can find an approximate and, more importantly, *sparse* solution to the FTRL using a only few number of calls to the classification oracle (sparse in the sense that the size of the support of P_t only depends on N logarithmically).

[Agarwal et al., 2014] shows that the regret of this algorithm is $\widetilde{\mathcal{O}}(\sqrt{TK\ln N})$ (again, comparable to Exp4 except that Exp4 works for the adversarial setting). The analysis is yet again very different from the standard FTRL analysis covered in this course; see Luo [2017] for a short summary. The fact that the log-barrier regularizer plays an important (but somewhat different) role in both the setting with a regression oracle and that with a classification oracle is intriguing.

References

Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

Zachary Chase, Shinji Ito, and Idan Mehalel. A tight lower bound for non-stochastic multi-armed bandits with expert advice. *arXiv preprint arXiv:2511.00257*, 2025.

Dylan Foster and Alexander Rakhlin. Beyond ucb: Optimal and efficient contextual bandits with regression oracles. In *International Conference on Machine Learning*, pages 3199–3210. PMLR, 2020.

Dylan J Foster, Claudio Gentile, Mehryar Mohri, and Julian Zimmert. Adapting to misspecification in contextual bandits. *Advances in Neural Information Processing Systems*, 33:11478–11489, 2020.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, pages 5662–5670. PMLR, 2020.

Haipeng Luo. Lecture notes 21, introduction to online learning, 2017. URL `https://haipeng-luo.net/courses/CSCI699/lecture21.pdf`.

David Simchi-Levi and Yunzong Xu. Bypassing the monster: A faster and simpler optimal algorithm for contextual bandits under realizability. *Mathematics of Operations Research*, 2021.