CMSC 858G: Bandits, Experts and Games 11/07/16

Lecture 10 : Contextual Bandits

Instructor: Alex Slivkins Scribed by: Guowei Sun and Cheng Jie

1 Problem statement and examples

In this lecture, we will be talking about a generalization of bandits called *contextual bandits*. Here, each round t proceeds as follows:

- algorithm observes a "context" x_t,
- algorithm picks an arm a_t,
- reward $r_t \in [0,1]$ is realized.

The reward r_t depends both on the context x_t and the chosen action a_t. Formally, make the IID assumption: r_t is drawn independently from some distribution that depends on the (x_t, a_t) pair but not on t. The expected reward of action a given context x is denoted $\mu(a|x)$. This setting allows a limited amount of "change over time", but this change is completely "explained" by the observable contexts. Throughout, we assume contexts $x_1, x_2, \ldots$ are chosen by an oblivious adversary.

The main motivation is that a user with a known "user profile" arrives in each round, and the context is the user profile. Some of the natural scenarios include: choosing which news articles to display to a given reader, which products to recommend to a given customer, and which interface options (*e.g.,* font sizes and colors, page layout, etc.) to use for a given website visitor. Rewards are often determined by user clicks, possibly in conjunction with other observable signals that correlate with revenue and/or user satisfaction. Naturally, rewards for the same action may be different for different users.

Contexts can include other things apart from (and instead of) the user profiles. First, known features of the environment, such as day of the week, time of the day, season (*e.g.,* Summer, pre-Christmas shopping season), proximity to a major event (*e.g.,* Olympics, elections). Second, the set of feasible actions: only a subset of actions may be allowed in a given round and/or for a given user. Third, actions can come with features of their own, and it may be convenient to include this information into the context, esp. if the action features change over time.

For ease of exposition, we assume a fixed and known time horizon T. The set of actions is denoted by $\mathcal{A}$, and $K = |\mathcal{A}|$ is the number of actions. $\mathcal{X}$ denotes the set of all contexts. For brevity, we will use `CB` instead of "contextual bandits".

The reward of an algorithm `ALG` is $\texttt{REW}(\texttt{ALG}) = \sum_{t=1}^{T} r_t$, so that the expected reward is

$$\mathbb{E}[\texttt{REW}(\texttt{ALG})] = \sum_{t=1}^{T} \mu(a_t|x_t).$$

One natural goal is to compete with the best response:

$$\pi^*(x) = \max_{a \in \mathcal{A}} \mu(a|x) \tag{1}$$

Then regret is defined as

$$R(T) = \texttt{REW}(\pi^*) - \texttt{REW}(\texttt{ALG}). \tag{2}$$

2 Small number of contexts

One straightforward approach for CB is to apply a known bandit algorithm such as UCB1: namely, run a separate copy of this algorithm for each context.

Algorithm 1: Bandit algorithm ALG for each context

Initialization: For each context x, create an instance $\texttt{ALG}_x$ of algorithm ALG
for *each round* t **do**
  invoke algorithm $\texttt{ALG}_x$ with $x = x_t$
  "play" action a_t chosen by $\texttt{ALG}_x$, return reward r_t to $\texttt{ALG}_x$.
end

Let n_x be the number of rounds in which context x arrives. Regret accumulated in such rounds is $\mathbb{E}[R_x(T)] = O(\sqrt{K n_x \ln T})$. The total regret (from all contexts) is

$$\mathbb{E}[R(T)] = \sum_{x \in \mathcal{X}} \mathbb{E}[R_x(T)] = \sum_{x \in \mathcal{X}} O(\sqrt{K n_x \ln T}) \leq O(\sqrt{KT\,|\mathcal{X}|\,\ln T}).$$

Theorem 2.1. *Algorithm 1 has regret* $\mathbb{E}[R(T)] = O(\sqrt{KT\,|\mathcal{X}|\,\ln T})$*, provided that the bandit algorithm* ALG *has regret* $\mathbb{E}[R_{\texttt{ALG}}(T)] = O(\sqrt{KT \log T})$.

Remark 2.2. The square-root dependence on $|\mathcal{X}|$ is slightly non-trivial, because a completely naive solution would give linear dependence. However, this regret bound is still very high if $|\mathcal{X}|$ is large, *e.g.,* if contexts are feature vectors with a large number of features. To handle CB with a large number of contexts, we usually need to assume some structure, as we shall see in the rest of this lecture.

3 Lipshitz Contextual Bandits

As a simple example of how structure allows to handle CB with a large number of contexts, let us consider (a simple example of) CB with Lipschtzness. Namely, we assume that contexts map into the $[0, 1]$ interval (*i.e.,* $\mathcal{X} \subset [0, 1]$) so that the expected rewards are lipshitz w.r.t. the contexts:

$$|\mu(a|x) - \mu(a|x')| \leq L|x - x'| \quad \text{for any arms } a, a' \text{ and contexts } x, x', \tag{3}$$

where L is the Lipschitz constant.

One simple solution for this problem is given by uniform discretization of the context space. The approach is very similar to what we've seen for Lipschitz bandits and dynamic pricing; however, we need to be a little careful with some details: particularly, watch out for "discretized best response".

Let S be the ϵ-*uniform mesh* on $[0, 1]$, *i.e.,* the set of all points in $[0, 1]$ that are integer multiples of ϵ. We take $\epsilon = 1/(d-1)$, where the integer d is the number of points in S, to be adjusted later

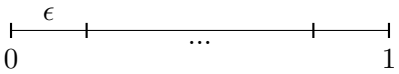

Figure 1: Discretization of Context Space

in the analysis.

We will use the CB algorithm from Section 2, applied to context space S; denote this algorithm as $\texttt{ALG}_S$. Let $f_S(x)$ be a mapping from context x to the closest point in S:

$$f_S(x) = \min(\operatorname*{argmin}_{x' \in S} |x - x'|)$$

(the min is added just to break ties). The overall algorithm proceeds as follows: in each round t, we "pre-process" the context x_t by replacing it with $f_S(x_t)$, and call $\texttt{ALG}_S$.

The regret bound will have two summands: regret bound for $\texttt{ALG}_S$ and (a suitable notion of) discretization error. Formally, let us define the "discretized best response" $\pi^*_S : \mathcal{X} \to \mathcal{A}$:

$$\pi^*_S(x) = \pi^*(f_S(x)) \quad \text{for each context } x \in \mathcal{X}.$$

Then regret of $\texttt{ALG}_S$ and discretization error are defined as, resp.,

$$\begin{aligned} R_S(T) &= \texttt{REW}(\pi^*_S) - \texttt{REW}(\texttt{ALG}_S) \\ \texttt{DE}(S) &= \texttt{REW}(\pi^*) - \texttt{REW}(\pi^*_S). \end{aligned}$$

It follows that the "overall" regret is the sum $R(T) = R_S(T) + \texttt{DE}(S)$, as claimed. We have $\mathbb{E}[R_S(T)] = O(\sqrt{KT|S|\ln T})$ from Lemma 2.1, so it remains to upper-bound the discretization error and adjust the discretization step ϵ.

Claim 3.1. $\mathbb{E}[\texttt{DE}(S)] \leq \epsilon LT$.

Proof. For each round t and the respective context $x = x_t$,

$$\begin{aligned} \mu(\pi^*_S(x) \mid f_S(x)) &\geq \mu(\pi^*(x) \mid f_S(x)) && \text{(by optimality of } \pi^*_S) \\ &\geq \mu(\pi^*(x) \mid x) - \epsilon L && \text{(by Lipschitzness).} \end{aligned}$$

Summing this up over all rounds t, we obtain

$$\mathbb{E}[\texttt{REW}(\pi^*_S)] \geq \texttt{REW}[\pi^*] - \epsilon LT. \qquad \square$$

Thus, regret is

$$\mathbb{E}[R(T)] \leq \epsilon LT + O(\sqrt{\tfrac{1}{\epsilon} KT \ln T}) = O(T^{2/3}(LK \ln T)^{1/3}).$$

where for the last inequality we optimized the choice of ϵ.

Theorem 3.2. *Consider the Lispchitz* CB *problem with contexts in* $[0, 1]$. *The uniform discretization approach yields regret* $\mathbb{E}[R(T)] = O(T^{2/3}(LK \ln T)^{1/3})$.

Further remarks. We can have a more general Lipschitz condition in which the right-hand side in (3) is replaced with $\mathcal{D}_{\mathcal{X}}(x, x')$, an arbitrary metric on contexts. Further, we can have a Lipshitiz condition on both contexts and arms:

$$|\mu(a|x) - \mu(a'|x')| \leq D_{\mathcal{X}}(x, x') + D_{\mathcal{A}}(a, a') \quad \text{for any arms } a, a' \text{ and contexts } x, x', \tag{4}$$

where $D_{\mathcal{X}}, D_{\mathcal{A}}$ are arbitrary metrics on contexts and arms, respectively.

Uniform discretization approach easily extends to this setting. Choose an ϵ-mesh $S_{\mathcal{X}}$ for $(\mathcal{X}, \mathcal{D}_{\mathcal{X}})$ and an ϵ-mesh $S_{\mathcal{A}}$ for $(\mathcal{A}, \mathcal{D}_{\mathcal{A}})$,[1] Fix a bandit algorithm ALG, and run a separate copy $\texttt{ALG}_x$ of this algorithm for each context $x \in S_{\mathcal{X}}$, with $S_{\mathcal{A}}$ as a set of arms. In each round t, "round up" each context x_t to the nearest point $x \in S_{\mathcal{X}}$, and call the corresponding algorithm $\texttt{ALG}_x$.

Further, adaptive discretization approach can improves over uniform discretization, just as it does for (non-contextual) bandits. Here, it is useful to think about $\mathcal{X} \times \mathcal{A}$, the space of context-arms pairs. An algorithm in Slivkins (2014) implements adaptive discretization (and improves over the regret bounds for uniform discretization) by discretizing $\mathcal{X} \times \mathcal{A}$, rather than $\mathcal{X}$ and/or $\mathcal{A}$ separately. In fact, the algorithm and analysis extends to more general setting when the right-hand side of (4) is $\mathcal{D}((x, a),\ (x', a'))$, an arbitrary metric on context-arm pairs.

Consider the special case when the context x_t is simply the time t and the Lipschitz condition

$$|\mu(a|t) - \mu(a|t')| \leq \mathcal{D}_a(t, t'),$$

where $\mathcal{D}_a$ is a metric on $[T]$, possibly parameterized by arm a. This describes a bandit problem with adversarial rewards and restriction that expected rewards can only change slowly. The paradigmatic special cases are $\mathcal{D}_a(t, t') = \sigma_a|t - t'|$ (slow change per round), and $\mathcal{D}_a(t, t') = \sigma_a\sqrt{|t - t'|}$ (*e.g.,* mean reward of each arm a evolves as a random walk with step σ_a.) Interestingly, Slivkins (2014) solves these problems using a general algorithm for Lipschitz CB, and achieve near-optimal dynamic regret (see Section 7 in Lecture 8 for definition and discussion).

4 Linear Contextual Bandits

Let us recap the setting of *linear bandits*. One natural formulation is that each arm a is characterized by a feature vector $x_a \in [0, 1]^d$, and the expected reward is linear in this vector: $\mu(a) = x_a \cdot \theta$, for some fixed but unknown vector $\theta \in [0, 1]^d$. One can think of the tuple

$$x = (x_a \in \{0, 1\}^d : a \in \mathcal{A}) \tag{5}$$

as a "static context" (*i.e.,* a context that does not change from one round to another).

In linear *contextual* bandits, contexts are of the form (5), and the expected rewards are linear:

$$\mu(a|x) = x_a \cdot \theta_a \quad \text{for all arms } a \text{ and contexts } x, \tag{6}$$

for some fixed but unknown vector $\theta = (\theta_a \in \mathcal{R}^d : a \in \mathcal{A})$.[2]

This problem can be solved by a version of the UCB technique. Instead of constructing confidence bounds on the mean rewards of each arm, we do that for the θ vector. Namely, in each

[1] As in Lecture 6, an ϵ-mesh for a metric space $(Y, \mathcal{D})$ is a subset $S \subset Y$ such that for each point $y \in Y$ there is a point $y' \in S$ with $\mathcal{D}(y, y') \leq y$.

[2] We allow the θ_a in (6) to depend on the arm a. The special case when all θ_a's are the same is also interesting.

round t we construct a "confidence region" $C_t \subset \Theta$ such that $\theta \in C_t$ with high probability. (Here Θ be the set of all possible θ vectors.) Then we use C_t to construct an UCB on the mean reward of each arm given context x_t, and play an arm with the highest UCB. This algorithm is known as `LinUCB` (see Algorithm 2).

Algorithm 2: `LinUCB`: UCB-based algorithm for linear contextual bandits

for *round* t **do**
  Form a confidence region $C_t \in \Theta$ s.t. $\theta \in C_t$ with high probability
  For each arm a, compute $\texttt{UCB}_t(a|x_t) = \sup_{\theta \in C_t} x_a \cdot \theta_a$
  pick arm a which maximizes $\texttt{UCB}_t(a|x_t)$.
end

To completely specify the algorithm, one needs to specify what the confidence region is, and how to compute the UCBs. This is somewhat subtle, and there are multiple ways to do that. Full specification and analysis of LinUCB is a topic for another lecture.

Suitably specified versions of `LinUCB` allow for rigorous regret bounds, and work well in experiments. The best known regret bound is of the form $\mathbb{E}[R(T)] = \tilde{O}(\sqrt{dT})$, and there is a nearly matching lower bound $\mathbb{E}[R(T)] \geq \Omega(\sqrt{dT})$. Interestingly, the algorithm is known to work well in practice even for scenarios without linearity.

Bibliography note. `LinUCB` has been introduced in Li et al. (2010), and analyzed in Chu et al. (2011).

5 Contextual Bandits with a Known Policy Class

We now consider a more general contextual bandit problem where we do not make any assumptions on the mean rewards. Instead, we make the problem tractable by making restricting the benchmark in the definition of regret (*i.e.,* the first term in (2)). Specifically, we define a *policy* as a mapping from contexts to actions, and posit a known class of policies Π. Informally, algorithms only need to compete with the best policy in π. A big benefit of this approach is that it allows to make a clear connection to the "traditional" machine learning, and re-use some of its powerful tools.

For ease of presentation, we assume that contexts arrive as independent samples from some fixed distribution $\mathcal{D}$ over contexts, which is known to the algorithm. For a given policy π, the expected reward is defined as

$$\mu(\pi) = \mathop{\mathbb{E}}_{x \in \mathcal{D}}[\mu(\pi(x))|x] \tag{7}$$

The regret of an algorithm `ALG` w.r.t. policy class Π is defined as

$$R_\Pi(T) = T \max_{\pi \in \Pi} \mu(\pi) - \texttt{REW}(\texttt{ALG}). \tag{8}$$

Note that the definition (2) can be seen a special case when Π is the class of all policies.

A simple but slow solution. One simple solution is to use a powerful algorithm `Exp4` that we've seen before, treating policies $\pi \in \Pi$ as "experts".

Theorem 5.1. *Algorithm* `Exp4` *with expert set* Π *yields regret* $\mathbb{E}[R_\Pi(T)] = O(\sqrt{KT \ln |\Pi|})$. *However, the running time per round is linear in* $|\Pi|$.

This is a powerful result: it works for an arbitrary policy class Π, and the logarithmic dependence on $|\Pi|$ makes the problem (reasonably) tractable, as far as regret bounds are concerned, even if the number of possible contexts is huge. Indeed, while there are $K^{|\mathcal{X}|}$ possible policies, for many important special cases $|\Pi| = K^c$, where c depends on the problem parameters but not on $|\mathcal{X}|$. However, `Exp4` has a very bad running time which scales as $|\Pi|$ rather than $\log |\Pi|$, which makes the algorithm prohibitively slow in practice.

Connection to a classification problem. We would like to achieve similar regret rates — (poly)logarithmic in $|\Pi|$ — but with a faster algorithm. To this end, we make a connection to a well-studied classification problem in "traditional" machine learning. This connection will also motivate the choices for the policy class Π.

To build up the motivation, let us consider "`CB` with predictions": a contextual analog of the "bandits with prediction" problem that we've seen before. In "`CB` with predictions", in the end of each round t the algorithm additionally needs to predict a policy $\pi_t \in \Pi$. The prediction is the algorithm's current guess for the best policy in Π: a policy $\pi = \pi^*_\Pi$ which maximizes $\mu(\pi)$.

Let's focus on a much simpler sub-problem: how to predict the best policy when so much data is collected that the mean rewards μ are essentially known. More formally:

$$\text{Given } \mu(a|x) \text{ for all arms } a \text{ and contexts } x \in \mathcal{X}, \text{ find policy } \pi \in \Pi \text{ so as to maximize } \mu(\pi). \quad (9)$$

Intuitively, any good solution for "`CB` with predictions" would need to solve (9) as a by-product. A very productive approach for designing `CB` algorithms goes in the opposite direction: essentially, it assumes that we already have a good algorithm for (9), and uses it as a subroutine for `CB`.

Remark 5.2. We note in passing that (9) is also well-motivated as a stand-alone problem. Of course, if $\mu(a|x)$ is known for all arms and all contexts, then an algorithm can simply compute the best response (1), so computing the best policy seems redundant. The interpretation that makes (9) interesting is that $\mathcal{X}$ is the set of all contexts already seen by the algorithm, and the goal is to choose an action for a new contexts that may arrive in the future.

For our purposes, it suffices to consider a special case of (9) with finitely many contexts and uniform distribution $\mathcal{D}$. Restating in a more flexible notation, the problem is as follows:

- Input: N data points $(x_i, c_i(a) \in \mathcal{A})$, $i = 1, \ldots, N$, where $x_i \in \mathcal{X}$ are distinct contexts, and $c_i(a)$ is a "fake cost" of action a for context x_i.
- Output: policy $\pi \in \Pi$ to approximately minimizes the empirical policy cost

$$c(\pi) = \frac{1}{N} \sum_{i=1}^{N} c_i(\pi(x_i)). \quad (10)$$

To recap, this is (still) a simple sub-problem of "`CB` with predictions", which

This happens to be a well-studied problem called "cost-sensitive multi-class classification" for policy class Π. An algorithm for this problem will henceforth be called a *classification oracle* for Π. While the exact optimization problem is NP-hard in the worst case for many natural policy classes, practically efficient algorithms have been designed for several important policy classes such as linear classifiers, decision trees and neural nets.

Remark 5.3. W.l.o.g., one can start with non-distinct contexts $x'_1, \ldots, x'_N \in \mathcal{X}$. Then we can define distinct contexts $x'_i = (x_i, i)$, $i \in [N]$. Each policy π is extended to the x'_i's by writing $\pi(x'_i) = \pi(x_i)$.

A simple oracle-based algorithm. We will use a classification oracle $\mathcal{O}$ as a subroutine for designing computationally efficient `CB` algorithms. The running time is then expressed in terms of the number of oracle calls (the implicit assumption being that each oracle call is reasonably fast). Crucially, `CB` algorithms designed with this approach can use any available classification oracle; then the relevant policy class Π is simply the policy class that the oracle optimizes over.

Let us consider a simple explore-then-exploit `CB` algorithm based on this approach. First, we explore uniformly for the first N rounds, where N is a parameter. Each round $t = 1, \ldots, N$ of exploration gives a data point $(x_t, c_t(a) \in \mathcal{A})$ for the classification oracle, where the "fake costs" are given by inverse propensity scoring:

$$c_t(a) = \begin{cases} -r_t K & \text{if } a = a_t \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Finally, we call the oracle and use the policy returned by the oracle in the remaining rounds; see Algorithm 3.

Algorithm 3: Explore-then-exploit with a classification oracle

Parameter: exploration duration N, classification oracle $\mathcal{O}$

1. Explore uniformly for the first N rounds: in each round, pick an arm u.a.r.
2. Call the classification oracle with data points $(x_t, c_t(a) \in \mathcal{A})$, $t \in [N]$ as per (11).
3. Exploitation: in each subsequent round, use the policy π_0 returned by the oracle.

Remark 5.4. Algorithm 3 is modular in two ways: it can take an arbitrary classification oracle, and, in principle, use any other unbiased estimator instead of (11). In particular, the proof below only uses the fact that (11) is an unbiased estimator with $c_t(a) \leq K$.

For a simple analysis, assume that the rewards are in $[0, 1]$ and that the oracle $\mathcal{O}$ is *exact*, in the sense that it returns a policy $\pi \in \Pi$ that exactly maximizes $\mu(\pi)$.

Theorem 5.5. *Assume rewards lie in* $[0, 1]$. *Let* $\mathcal{O}$ *be an exact classification oracle for some policy class* Π. *Algorithm 3 parameterized with oracle* $\mathcal{O}$ *and* $N = T^{2/3}(K \log(|\Pi|T))^{\frac{1}{3}}$ *has regret*

$$\mathbb{E}[R_\Pi(T)] = O(T^{2/3})(K \log(|\Pi|T))^{\frac{1}{3}}.$$

Proof. Let us consider an arbitrary N for now. For a given policy π, we estimate its expected reward $\mu(\pi)$ using the empirical policy costs from (10), where the action costs $c_t(\cdot)$ are from (11).

Let us prove that $-c(\pi)$ is an unbiased estimate for $\mu(\pi)$:

$$\begin{aligned}
\mathbb{E}[c_t(a)|x_t] &= -\mu(a|x_t) \qquad \text{(for each action } a \in \mathcal{A}) \\
\mathbb{E}[c_t(\pi(x_t))|x_t] &= -\mu(\pi(x)|x_t) \qquad \text{(plug in } a = \pi(x_t)) \\
\mathop{\mathbb{E}}_{x_t \sim D}[c_t(\pi(x_t))] &= \mathop{\mathbb{E}}_{x_t \sim D}[\mu(\pi(x_t))|x_t] \qquad \text{(take expectation over both } c_t \text{ and } x_t) \\
&= -\mu(\pi),
\end{aligned}$$

which implies $\mathbb{E}[c(\pi)] = -\mu(\pi)$, as claimed.

Now, let us use this estimate to set up a "clean event":

$$\left\{ |c(\pi) - \mu(\pi)| \leq \texttt{conf} \text{ for all policies } \pi \in \Pi \right\},$$

where the "confidence radius" is

$$\texttt{conf}(N) = O\left(\sqrt{\frac{K \log(T|\Pi|)}{N}} \right).$$

We can prove that the clean event does indeed happen with probability at least $1 - \frac{1}{T}$, say, as an easy application of Chernoff Bounds. For intuition, the K is present in the confidence radius is because it upper-bounds the costs $c_t(\cdot)$. The $|\Pi|$ is there (inside the log) because we take a Union Bound across all policies. And the T is there because we need the "error probability" to be on the order of $\frac{1}{T}$.

Let $\pi^* = \pi^*_\Pi$ be an optimal policy. Since we have an exact classification oracle, $c(\pi_0)$ is maximal among all policies $\pi \in \Pi$. In particular, $c(\pi) \geq c(\pi^*)$. If the clean event holds, then

$$\mu(\pi^\star) - \mu(\pi_0) \leq 2\,\texttt{conf}(N).$$

Thus, each round in exploitation contributes at most `conf` to expected regret. And each round of exploration contributes at most 1. It follows that $\mathbb{E}[R_\Pi(T)] \leq N + 2T\,\texttt{conf}(N)$. Choosing N so that $N = O(T\,\texttt{conf}(N))$, we obtain $N = T^{2/3}(K \log(|\Pi|T))^{\frac{1}{3}}$ and $\mathbb{E}[R_\Pi(T)] = O(N)$. □

Remark 5.6. Suppose classification oracle in Theorem 5.5 is only approximate: say, it returns a policy $\pi_0 \in \Pi$ which optimizes $c(\cdot)$ up to an additive factor of ϵ. It is easy to see that expected regret increases by an additive factor of ϵT. In practice, there may be a tradeoff between the approximation guarantee ϵ and the running time of the oracle.

Further discussion. The oracle-based approach to `CB` was pioneered in Langford and Zhang (2007), with a slightly more complicated "epsilon-greedy"-style algorithm, and a similar analysis.

The key features of Theorem 5.5 are: polylogarithmic dependence on $|\Pi|$, $T^{2/3}$ dependence on T (up to log factors), and a single oracle call per round. Follow-up work investigated the tradeoff between dependence on T and the number of oracle calls (keeping polylog($|\Pi|$) dependence as a must). One concrete goal here is to achieve the optimal $\sqrt{T}$ dependence of regret on T with only a "small" number of oracle calls. In particular, a recent break-through result of (Agarwal et al., 2014) achieved regret bound $O(\sqrt{KT \log(T|\Pi|)})$ with only $\tilde{O}(\sqrt{KT/\log|\Pi|})$ oracle calls across all T rounds. However, this result does not appear to extend to approximate oracles, and calls the oracle on a number of carefully constructed artificial problem instances (so if a given classification

oracle performs greatly in practice, this would not necessarily carry over to a great performance on these artificial problem instances). However, their algorithm appears to perform well in simulations.

Another recent break-through (Syrgkanis et al., 2016a; Rakhlin and Sridharan, 2016; Syrgkanis et al., 2016b) extends `CB` with classification oracles to adversarial rewards. The best current result in this line of work involves $T^{2/3}$ regret, so there may still be room for improvement.

6 Practical view

For now, please refer to Section 3 of Agarwal et al. (2016).

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 *31st Intl. Conf. on Machine Learning (ICML)*, 2014.

Alekh Agarwal, Sarah Bird, Markus Cozowicz, Miro Dudik, John Langford, Lihong Li, Luong Hoang, Dan Melamed, Siddhartha Sen, Robert Schapire, and Alex Slivkins. Multiworld testing: A system for experimentation, learning, and decision-making. 2016. A white paper, available at `https://github.com/Microsoft/mwt-ds/raw/master/images/MWT-WhitePaper.pdf`.

Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual Bandits with Linear Payoff Functions. In *14th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011.

John Langford and Tong Zhang. The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits. In *21st Advances in Neural Information Processing Systems (NIPS)*, 2007.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *19th Intl. World Wide Web Conf. (WWW)*, 2010.

Alexander Rakhlin and Karthik Sridharan. BISTRO: an efficient relaxation-based method for contextual bandits. In *33nd Intl. Conf. on Machine Learning (ICML)*, 2016.

Aleksandrs Slivkins. Contextual bandits with similarity information. *J. of Machine Learning Research (JMLR)*, 15(1):2533–2568, 2014. Preliminary version in *COLT 2011*.

Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert E. Schapire. Efficient algorithms for adversarial contextual learning. In *33nd Intl. Conf. on Machine Learning (ICML)*, 2016a.

Vasilis Syrgkanis, Haipeng Luo, Akshay Krishnamurthy, and Robert E. Schapire. Improved regret bounds for oracle-based adversarial contextual bandits. In *29th Advances in Neural Information Processing Systems (NIPS)*, 2016b.